INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THÈSE

pour obtenir le grade de DOCTEUR de l'INPG

Spécialité : Imagerie, Vision, Robotique

Préparée au sein du projet SHARP,
commun au Laboratoire GRAphique, VIsion, Robotique et à l'INRIA Rhône-Alpes,
ZIRST, 655 av. de l'Europe, 38330 Montbonnot St. Martin – France

dans le cadre de l'Ecole Doctorale Mathématiques, Sciences et Technologie de l'Information, Informatique

présentée et soutenue publiquement par

Diego d'Aulignac

le 3 décembre 2001

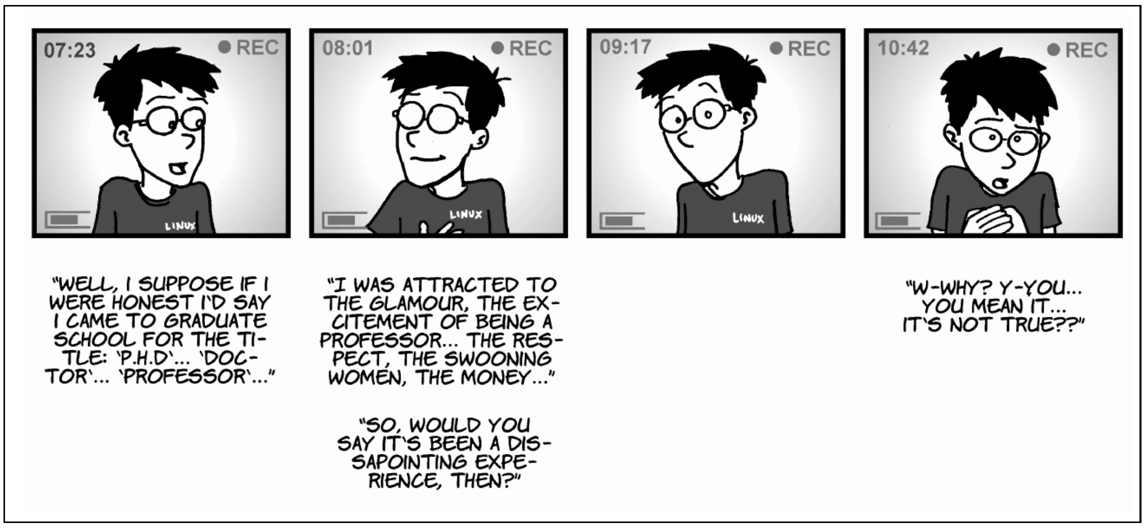
Modélisation de l'interaction avec objets déformables en temps-réel pour des simulateurs médicaux

Directeur de thèse : Christian Laugier

Composition du jury :

Présidente :	Mme. Marie-Paule Cani
Rapporteurs :	M. Christophe Chaillou
	M. Hervé Delingette
Examinateurs :	M. Oussama Khatib
	M. Yohan Payan

À mon père...


07:23
● REC
LINUX
"WELL, I SUPPOSE IF I WERE HONEST I'D SAY I CAME TO GRADUATE SCHOOL FOR THE TITLE: 'P.H.D'... 'DOCTOR'... 'PROFESSOR'..."
08:01
● REC
LINUX
"I WAS ATTRACTED TO THE GLAMOUR, THE EXCITEMENT OF BEING A PROFESSOR... THE RESPECT, THE SWOONING WOMEN, THE MONEY..."
"SO, WOULD YOU SAY IT'S BEEN A DISSAPOINTING EXPERIENCE, THEN?"
09:17
● REC
LINUX
10:42
● REC
"W-WHY? Y-YOU... YOU MEAN IT... IT'S NOT TRUE??"

"Piled Higher and Deeper" by Jorge Cham, © Stanford Daily.

Remerciements

First of all, I would like to thank my thesis committee members:

- Marie-Paule Cani, Professor at the INPG, for assuming the role of the president of the committee;
- Christophe Chaillou, Professor at the University of Lille, and Hervé Delingette, Researcher at INRIA, for reviewing my manuscript and their constructive comments that allowed me to improve the quality of the final version;
- Oussama Khatib, Professor at Stanford, and Yohan Payan, Researcher at TIMC-IMAG, for their insightful comments and questions at the defense;
- Christian Laugier, Head of Research at INRIA, who has supervised me over these years while leaving me enough freedom to pursue my research.

Much of the work on the modeling of the thigh has been done within the framework of the France-Berkeley cooperation; in particular I would like to thank Cenk Cavusoglu with whom I've had the pleasure to work with. I'd also like to thank Jocelyne Troccaz's group at the TIMC-IMAG laboratory for their help with the generation of echographic images. Further, I would like to thank everybody involved in the AISIM and Caesare projects for the interesting discussions that we had.

During these years the staff at INRIA have been both efficient and friendly. The librarians have helped me to find more than one article or book I would otherwise never have read. On the other hand, the secretaries have guided me through the often obscure bureaucratic procedures that perplexed me.

Of course, I would like to thank everybody in the SHARP group for the great research environment and relaxed atmosphere. In particular I would like to thank the following people (in order of appearance): Anton Deguet who never tired in explaining to me the basic concepts of physically based modeling as well as helping me to settle in Grenoble; François Boux de Casson with whom I've shared my office and many other pleasurable moments over these years; Sepanta Sekhavat for his eternal good humor; Kenneth Sundaraj with whom I can talk just as easily about collision detection as last Sunday's football results; the flying boys from Brazil, Remis Balaniuk and Ivan Costa, for their scientific rigor as well as their personal kindness; and Stephane Vieira who has been more than just an intern for his final year project.

Many thanks to all of those that have proof-read the early versions of my manuscript. They have allowed me to correct several errors in this document. Any remaining mistakes are, of course, entirely their responsibility. Moreover, all my respect to those that attended the presentations that allowed me to prepare my defense. I can't even begin to imagine the pain and agony that I've put them through. Nevertheless, their comments and suggestions have helped me to present my work in a more comprehensible manner. I don't know what I would have done without them.

It was a great comfort to see many of my friends at my defense and I'd like to thank them for being there. Also I appreciate the efforts of Veronika, Marta, Sep, and all those that helped to prepare the party afterwards.

Finally, thanks to Pauline for putting up with me during the sometimes stressful last few months of my thesis and helping me to keep calm and relax.

Contents

Synthèse

I Introduction

I.I Motivations

L'avènement de l'imagerie médicale a bouleversé les méthodes de travail des médecins. Les investigations de l'informatique pour faire avancer la médecine continuent, et un champs de travail, très actif actuellement, est celui du développement de simulateurs médicaux.

I.II Objectifs

Un des objectifs de ces travaux est de proposer aux médecins des simulateurs pour l'entraînement à des procédures chirurgicales, de la même manière qu'il existe des simulateurs de vol pour former les pilotes. Afin d'être réalistes, ces simulateurs doivent évidemment intégrer des modèles mécaniques de déformation des organes. Le défi est d'envergure, car la mécanique du "vivant" est loin d'être complètement connue, du fait même de la complexité des systèmes vivants.

I.III Simulateur existants

Aujourd'hui plusieurs simulateurs médicaux existent déjà. Par exemple on peut s'initier à la laparoscopie [Tendick et al., 2000] en utilisant un simulateur doté d'un dispositif retour d'efforts ou interagir avec un modèle mécanique de l'oeil [Sagar et al., 1994] basé sur les éléments finis. Aussi des modèles construits à partir de donnés du patient ont été proposés. Par contre, la performance en temps réel de la simulation est essentielle pour une simulation interactive.

I.IV Structure

Cette synthèse est divisée en cinq parties principales. Les différentes approches pour la modélisation des objets déformables seront abordées avant de nous pencher sur les problématiques des résolutions statiques et dynamiques des systèmes. Une fois l'objet déformable modélisé il faut encore pouvoir interagir avec lui: nous examinerons les détections et réponses aux collisions ainsi que le retour haptique. Finalement ces bases théoriques trouveront une application dans le cadre d'un simulateur échographique et un modèle interactif du foie humain.

II Modélisation des corps déformables

II.I Introduction

La majorité de l'anatomie humaine est composée de tissus mous. Pour décrire le phénomène de la déformation, le domaine de la mécanique a mis en place une base théorique pour les milieux continus; mais pour modéliser les objets de géométrie complexe on devra néanmoins les discrétiser en éléments pour appliquer cette théorie de la déformation.

II.II Mécanique des milieux continus

Le déplacement ou rotation d'un objet n'implique pas forcément une déformation. La mécanique des milieux continus nous donne les bases théoriques pour mesurer une déformation ainsi que la vitesse de déformation. Ces mesures seront directement liées à la contrainte de l'objet, soit par une relation linéaire ou en fonction des caractéristiques du matériau. La contrainte nous permettra de calculer les forces internes dans l'objet dues à la déformation (voir aussi Section 2.2).

II.III Modèles Discrets

La mécanique des milieux continus n'est pas applicable directement aux objets de géométrie complexe. Pour cela on divise l'objet en éléments. Plusieurs approches existent.

II.III.I Maillage Masse-Réssort

Des ressorts modélisent le matériau entre deux points. Pour cela il peut s'avérer difficile de modéliser un objet 3D avec ces éléments. Le comportement physique d'un modèle utilisant ces composants sera donc explicitement lié à la topologie du maillage (Figure 2.7).

II.III.II Méthodes des Éléments Finis

Les éléments finis volumétriques peuvent modéliser le comportement physique d'un objet 3D, qui en principe ne dépend pas de la discrétisation géométrique de l'objet. En plus les phénomènes des conservations de volume sont faciles à mettre en oeuvre. Malheureusement ce type d'élément demande plus de calcul qu'un ressort; on a trouvé que le temps de calcul des forces internes pour un objet modélisé avec des éléments finis est environ quatre fois supérieur au même objet modélisé avec des masses-ressorts (voir Tablau 2.4).

II.III.III Système des Particules

Un objet peut être modélisé par un ensemble de particules. Les forces d'attraction et de répulsion entre *toutes* ces particules sont normalement en fonction de la distance qui les sépare, mais il est tout à fait possible d'inclure d'autres critères comme par exemple la température. Cette approche se prête particulièrement bien à la modélisation des fluides ou de la fumée, mais il est problématique de simuler le comportement physique des objets à topologie fixe.

II.III.IV Contraintes

Au lieu de calculer la force sur un point, laquelle provoquera un déplacement, il est envisageable d'imposer directement un déplacement. Cette méthodologie a déjà été largement appliquée aux objets articulés, mais il est possible de l'utiliser aussi pour la modélisation d'objets déformables. Par exemple on peut imposer une distance maximale et minimale entre deux particules forcée par une contrainte de déplacement.

II.IV Conclusions

On a vu que plusieurs approches existent pour discrétiser un objet de géométrie complexe dans le but d'appliquer la théorie de déformation des milieux continus. Les ressorts sont en fait des éléments qui modélisent la matière entre deux particules; malheureusement le comportement physique d'un maillage masses-ressorts sera donc en fonction de sa topologie. Les éléments finis tetrahedriques ne présentent pas ce problème, mais ont une complexité de calcul supérieure.

III Résolution statique et dynamique

III.I Introduction

Dans la section précédente on a examiné différents modèles physiques qui fournissent les forces intérieures à l'objet. Cela nous permettra de calculer la déformation due à une force extérieure. On pourra choisir entre deux méthodes de résolution: statique ou dynamique. La résolution statique cherche à trouver directement une position d'équilibre, pendant que la résolution dynamique s'intéresse à l'évolution de la déformation dans le temps.

III.II Résolution statique linéaire

Étant donné que le déplacement de tous les points de l'objet modélisé est petit une résolution linéaire est suffisante. Pour cela on doit résoudre un système d'équations pour trouver une configuration où les forces intérieures à l'objet s'équilibrent parfaitement avec les forces extérieures (principe des travaux virtuels).

III.III Résolution statique non-linéaire

Si l'objet subit une grande déformation où rotation on doit prendre en compte les non-linéarités géométriques et matériels. Cela nous oblige à résoudre un système non-linéaire, par exemple, à l'aide de la méthode de Newton-Raphson.

La méthode de Newton-Raphson (NR) nous permet de résoudre un système non-linéaire par la résolution *itérative* d'un système d'équations. Cela implique que l'on doit recalculer les forces internes et la matrice de rigidité à chaque itération. Cette méthode est donc beaucoup plus coûteuse qu'une résolution linéaire (voir Figure 3.4).

La méthode de Newton-Raphson modifié (NRM) se distingue de la méthode de NR, uniquement par le fait que la matrice de rigidité n'est pas recalculé à chaque itération. Cela veut dire que pour des problèmes non-linéaires NRM devra prendre *plus* d'itérations que NR pour trouver la solution, mais chaque itération de NRM est *moins* coûteuse en termes de complexité de calcul (voir Figure 3.5).

Les méthodes de NR et NRM peuvent être troublées par des problèmes de convergence. Une solution possible pour ce problème est de multiplier le vecteur de changement d'état par un facteur inférieur à 1. Ce facteur dépend de la non-linéarité du système est peut être déterminé expérimentalement.

III.IV Résolution Dynamique

La résolution statique cherche directement une configuration d'équilibre si elle existe. Par contre s'il n'y a pas de configuration unique d'équilibre ou que l'on s'intéresse à l'évolution de la déformation dans le temps une analyse dynamique s'avère nécessaire.

L'intégration explicite utilise les dérivées à la configuration courante. Malheureusement ces dérivées ne sont pas constantes ce qui provoquera une solution erronée. Il est possible de prendre en compte des dérivées d'ordre supérieur pour obtenir une solution plus exacte, mais le pas de temps sera toujours limité par les paramètres de rigidité, viscosité, et masse pour une intégration stable (Equation 3.21).

A la différence de l'intégration explicite, l'intégration implicite cherche à trouver la configuration dans laquelle les dérivées vont nous ramener à l'état courant pour un pas de temps négatif. Cela veut dire qu'on pourra toujours retourner à l'état précédent, ce qui garantit que cette méthode d'intégration est absolument stable (mais pas forcément exacte) indifféremment des paramètres physiques du système. Malheureusement cette méthode nécessite la résolution d'un système non-linéaire à chaque pas de temps, ce qui la fait très coûteuse en termes de complexité de calcul (voir Section 3.4.2).

Un mode est dit normal lorsque toutes les valeurs d'état, changent de façon harmonique et à la même fréquence (mais pas forcément à la même amplitude et phase). Tout mouvement du système peut alors être décrit comme une superposition de modes normaux. Cela permet de réécrire le système comme une série d'équations indépendantes (Equation 3.35). La suppression des modes qui décrivent les hautes fréquences peut alors contribuer à une meilleure stabilité numérique, mais le prix à payer est la recherche des valeurs propres du système.

III.V Résolution des systèmes linéaires

Autant la résolution statique comme l'intégration implicite nécessitent la résolution d'un ou plusieurs systèmes d'équations linéaires. Une multitude d'approches existent pour ce problème.

III.V.I Méthode de Jacobi

La méthode de Jacobi essaye de résoudre un système linéaire d'équations en trouvant la solution pour chaque équation de façon indépendante. Si cela est fait plusieurs fois dans une boucle itérative il est possible de se rapprocher de la solution pour le système global. Il peut être prouvé que cette approche converge étant donné que la valeur propre maximale (où rayon du spectre) de la matrice est entre -1 et 1.

III.V.II Méthode de Gauss-Seidel

Cette méthode est très similaire à la méthode de Jacobi. La différence notable est que la solution trouvée pour une équation est utilisée immédiatement dans la même itération pour la résolution des équations suivantes. Cela augmente considérablement la vitesse de convergence, laquelle est forcément liée à l'ordre dans lequel les équations sont résolues.

III.V.III Méthode de sur-relaxation successive

Il est possible d'encore augmenter la vitesse de convergence par rapport à la méthode de Gauss-Seidel en introduisant un facteur de sur-correction à la résolution de chaque équation. La valeur optimale de ce facteur peut être calculée à partir du rayon du spectre, mais en pratique cette valeur est souvent évaluée de manière expérimentale.

III.V.IV Méthode du gradient conjugué

Les méthodes précédentes pour la résolution d'un système d'équations linéaires répètent les mêmes opérations à chaque itération; elles sont aussi appelées méthodes stationnaires. La méthode des moindres carrés ou des gradients conjugués changent à chaque itération leur direction de recherche pour mieux minimiser l'erreur de *toutes* les équations du système.

III.VI Réseaux de neurones

On a vu que la résolution d'un système non-linéaire peut être accomplie par la solution itérative de plusieurs systèmes linéaires. Cela est malheureusement un processus assez coûteux en termes de complexité de calcul. Par contre les réseaux de neurones sont capables d'approcher la solution d'un système non-linéaire pour un coût de calcul faible après apprentissage. Pour cette raison nous examinerons leur utilisation dans le contexte de la simulation physique.

Le perceptron multi-couche (Figure 3.14) est un des réseaux de neurones les plus utilisés. Grâce à une couche cachés et de neurones avec une fonction sigmoidale il est capable d'approcher une grande variété de problèmes non-linéaires.

III.VII Conclusions

Il est possible de trouver la déformation soit par une résolution statique où dynamique. Pour une résolution statique il faut qu'une configuration d'équilibre existe; pour des petites déformations on peut choisir un résolution linéaire, mais sinon une résolution non-linéaire plus coûteuse est exigée. Par contre si l'évolution de la déformation dans le temps nous intéresse, nous devrons procéder à une analyse dynamique. La stabilité de l'intégration explicite est dépendante des paramètres physiques du système. Cela n'est pas le cas avec une intégration implicite, mais on doit résoudre un système non-linéaire à chaque pas de temps.

IV Interaction

IV.I Introduction

Dans la conception simulateur médical il est essentiel que l'utilisateur puisse interagir avec le modèle physique. On devra donc d'abord détecter les collisions entre les outils contrôlés par l'utilisateur. Si une collision est détectée il s'avérera nécessaire de calculer une réponse à cette collision. Finalement, si on utilise un interface haptique on devra fournir des forces de contact à une très haute fréquence.

IV.II Détection des Collisions

La détection de la collision entre deux enveloppes polyédriques peut demander un temps de calcul assez élevé. Il est donc important d'optimiser cette procédure pour pouvoir envisager une simulation en temps-réel.

Il existent plusieurs approches pour détecter la collision ou calculer la distance entre deux enveloppes polyédriques convexes. On comparera ces approches en termes de complexité de calcul (Section 4.2.1).

Malheureusement pas toute l'anatomie humaine est composée d'organes convexes. Dans certains cas il est possible de diviser un objet concave en plusieurs objets convexes et appliquer les algorithmes présentes dans la section précédente. Sinon il faudra trouver explicitement les facettes en interaction; pour éviter une complexité quadratique différentes méthodes d'optimisation existent (Section 4.2.2).

IV.III Réponse à la Collision

Une fois que la collision entre deux objets a été détectée, une réponse physiquement réaliste doit être calculée. On pourra différencier entre les approches suivantes.

- **Contraintes** Une possibilité est de considérer les collisions comme des contraintes supplémentaires dans la déformation de l'objet. Malheureusement ce type d'approche introduit un système d'équations non-linéaires qu'il est difficile de résoudre en temps-réel.

- **Impulse** L'impulse traite la collision comme une interaction ponctuelle dans le temps et l'espace. La vitesse de l'objet change de façon instantanée. Ce point fait que cette méthode est très appropriée aux objets rigides, mais pas aux objets déformables car la collision ne peut pas être considérée comme instantanée.

- **Méthodes de Pénalité** La méthode utilise une mesure d'interpénétration des objets en collision pour calculer une force de pénalité, destinée à les séparer. Pour simuler le phénomène de viscosité lors d'une collision la vitesse d'interpénétration est utilisée. Malheureusement cette méthode souffre d'une discontinuité dans les forces; pour remédier à cela on introduit du domaine de la mécanique une méthode de pénalité non-linéaire.

IV.IV Interface Haptique

Une interface haptique est un dispositif de retour d'effort. Dans ce travail on utilisera le PHANToM distribué par Sensable Technologies (Figure 4.6). Cet engin fournit des positions dans un espace 3D, et on peut à travers lui envoyer des forces à l'utilisateur. Par contre les forces doivent être envoyées à des fréquences élevées pour garantir une bonne sensation haptique.

IV.IV.I Modèle local

Si les forces peuvent pas être calculées aux fréquences requises, on se propose d'utiliser une approximation locale de la collision. Pour cela la détection et la réponse à la collision sont calculées à partir d'un modèle géométrique simple, permettant le calcul des forces à haute fréquence.

IV.V Conclusions

La reponse à la collision est directement liée à la détection. Pour des objets de géométrie complexe, la détection de la collision peut être très coûteuse en calcul. Cela veut dire que les forces de pénalité sont évaluées à des fréquences trop basses pour un rendu haptique réaliste. Pour remédier à cette situation on introduit une approximation locale du contact.

V Simulateur Échographique

V.I Introduction

L'échographie est largement utilisée dans le milieu médical comme un moyen pour diagnostiquer différentes pathologies de façon non-invasive et peu coûteuse. Un exemple est la détection de thromboses veineuses. Le désavantage de cet examen est qu'il est difficile d'apprendre à diagnostiquer correctement les pathologies, car cela demande l'expérience acquise sur un nombre considérable de patients. Pour cette raison nous proposons le développent d'un simulateur échographique.

V.II Travaux précédents

La génération d'images échographiques a été proposée par différentes approches. D'un coté il est possible de générer une image directement du modèle physique par la simulation de la propagation du son. L'alternative est de construire une image échographique à partir d'un processus d'interpolation entre des données déjà existantes.

V.III Modélisation de la cuisse humaine

La modélisation de la déformation étant essentielle au développement d'un simulateur échographique, on a étudié la physique de la cuisse humaine basée sur des mesures physiques.

On a utilisé un bras articulé doté d'un capteur de force pour mesurer les forces présentes pour une succession de déplacements pour une série de points sur la cuisse humaine (Figure 5.3). Ces mesures ont été effectuées avec deux sondes de forme différente (Figure 5.4).

La caractéristique des courbes de force-déformation sont fortement non-linéaires. Cela nous a incité à utiliser un modèle masses-ressorts multi-couches incorporant des ressorts non-linéaires (Section 5.3.2).

Identification des paramètres Ayant choisi le modèle et sa topologie on s'est penché sur la question de l'identification de ses paramètres. On a utilisé une minimisation de moindre carré pour trouver les paramètres du modèle qui donnent le comportement le plus proche des données expérimentales.

La déformation de la cuisse est seulement simulée dans la zone où a lieu l'examen échographique. Le reste de la jambe et le pied est tout de même représenté graphiquement pour un meilleur repérage dans l'espace virtuel.

V.IV Résolution Dynamique

Le modèle physique nous permet de calculer les forces internes pour une déformation donnée. Ces forces vont décrire les changements de l'état du modèle par rapport au temps.

V.IV.I Intégration Explicite

Les paramètres de rigidité du modèle physique étant assez élevés le pas de temps de l'intégration doit être très petit. Alternativement, si la gravité n'est pas présente, il est possible d'utiliser des masses artificiellement grandes pour stabiliser le système. L'utilisation de méthodes d'intégration d'ordre supérieur augmente la taille du pas de temps tout en garantissant une simulation stable, mais n'accélère pas la simulation globale à cause de la complexité de calcul supérieure.

V.IV.II Intégration Implicite

Pour une intégration implicite il est nécessaire de résoudre un système non-linéaire à chaque pas de temps. Néanmoins, on a constaté que pour cette application une résolution *semi-implicite* est suffisante pour garantir une simulation stable. En plus, la Jacobienne n'est évaluée qu'une fois, réduisant considérablement la complexité algorithmique.

V.IV.III Résultats expérimentaux

Pour une simulation avec détection et réponse à la collision sur une Silicon Graphics équipée d'un processeur R10000, la fréquence de calcul est de $100Hz$. Cela permet d'envisager le modèle de la cuisse comme une partie intégrante d'un simulateur échographique.

V.V Résolution statique à l'aide des réseaux de neurones

Dans le cas idéal on voudrait connaître les déplacements du système pour une force extérieure appliquée. Malheureusement cela demande la résolution d'un système non-linéaire, très gourmand en temps de calcul. On se propose alors d'approcher le système non-linéaire par un réseau de neurones.

On utilisera un réseau multi-couche classique pour approcher les déplacements directement à partir des mesures expérimentales effectués. Cela veut dire qu'il y aura autant d'entrées et de sorties pour le réseau neuronal, que de points de mesure.

Après l'entraînement du réseaux en utilisant un algorithme de retro-propagation de l'erreur, l'erreur moyen carré est de 4.1384×10^{-2}. Cela signifie que l'erreur est rarement supérieur à 1 Newton, et le comportement non-linéaire est bien approché.

Cette approche nous a donné des résultats encourageants pour un calcul bien inférieur à la résolution d'un système non-linéaire. Néanmoins cette approche a des limitations : par exemple, un changement de la topologie demanderais un re-entraînment du réseau, qui n'est pas envisageable en temps-réel.

V.VI Interaction

Dans les sections précédentes la déformation de la cuisse a été modélisée. Dans cette section nous examinerons la problématique de l'interaction.

Comme le modèle géométrique de la sonde échographique est de forme simple, nous avons utilisé une approche de détection des collisions basée sur la carte graphique, permettant ainsi une accélération. Cette approche détecte les triangles représentant la surface de la cuisse, qui sont à l'intérieur de la sonde. En calculant leur distance par rapport au but de la sonde, on pourra appliquer une force de collision.

Notre simulation dynamique (avec détection de la collision) tourne à une fréquence de environ $100Hz$. Par contre, si l'on envoie les forces de collision à cette fréquence au dispositif haptique, les retour de forces à l'utilisateur comporteront des à-coups considérables. Pour remédier à ce problème on a choisi d'implémenter un modèle local. Le modèle local est un plan qui passe à travers le triangle le plus proche. Tout calcul de la force envoyé à l'interface haptique, est calculé en utilisant cette représentation simplifiée, permettant d'atteindre les hautes fréquences requises.

En comparant les forces envoyées à l'utilisateur avec et sans modèle local on s'aperçoit que la sensation haptique est largement amélioré grâce à cette approximation.

Le modèle local que nous avons mis en oeuvre est une approximation locale du contact. Par ceci nous voulons dire que la détection et la réponse de collision sont calculées en utilisant un modèle géométrique simplifié à une fréquence beaucoup plus élevée. Cependant, la déformation n'est pas modelée; les paramètres du modèle local sont constants pendant une étape de la simulation physique.

V.VII Génération d'images échographiques

Comme outil pédagogique notre simulateur décrit jusqu'ici est stérile. Des images correspondant à la déformation réelle doivent être produites pour enseigner au praticien la corrélation entre la déformation de la surface de la cuisse et l'image échographique résultante.

La déformation de l'image quand la pression est appliquée peut être obtenue par l'approche décrite par [Henry, 1997]. Donné une segmentation de l'image échographique pour trouver l'artère, veine, et des zones de tissu mou, leur déformation peuvent être calculées en tenant compte des paramètres tels que la pression artérielle et veineuse. Comme mentionné précédemment, en modifiant ces critères nous pouvons simuler un ensemble de pathologies.

La Figure 5.16 montre comment l'image échographique change pendant qu'une pression croissante est appliquée à la surface de la cuisse par les moyens de la sonde échographique virtuelle. La position de la sonde est donnée par le dispositif haptique (voir Figure 5.17).

V.VIII Conclusions

Nous avons mesuré les forces dues à la déformation de la cuisse à l'aide d'un capteur de forces. Basés sur ces données des connecteurs non linéaires ont été utilisés dans un modèle masses-ressorts. De plus, les paramètres des ressorts ont été identifiés pour approcher mieux les mesures. Alternativement, nous avons examiné la praticabilité d'employer un réseau des neurones pour approcher la déformation statique, basée directement sur les données mesurées, avec des résultats prometteurs. Pour une analyse dynamique concluez que, pour ce cas, il est avantageux d'utiliser l'intégration implicite due à la rigidité des ressorts. Ceci permet une intégration stable avec la détection et la réponse de collision à une fréquence de $100Hz$. Ainsi, pour fournir des cadences plus élevées pour le dispositif haptique, une approximation locale du contact a été mise en application. Les résultats expérimentaux prouvent qu'un meilleur retour d'effort peut être réalisé en utilisant cette approche. Pour finir, le modèle physique a été intégré avec un générateur d'images échographiques, ayant pour résultat un premier prototype rudimentaire d'un simulateur échographique de la cuisse humaine.

VI Modèle interactif du foie humain

VI.I Introduction

Nous présentons le modèle d'un foie basé sur un réseau de masses-ressorts. Les différentes propriétés anatomiques et les essais rhéologiques de l'organe sont discutés. Basé sur cette information un modèle hétérogène et non linéaire est proposé. De plus, nous comparons l'analyse statique et dynamique et discutons leurs mérites relatifs. En conclusion, nous offrons les perspectives possibles de ce modèle en tant qu'élément d'un simulateur laparoscopique.

VI.II Description du modèle

En raison des propriétés anatomiques et biomécaniques mentionnées ci-dessus, un modèle hétérogène est choisi pour modeler le comportement mécanique du foie. Dans la suite, nous montrons comment nous avons modelé le foie en utilisant deux composants principaux : un composant pour modeler la capsule de Glisson et un autre pour modeler le parenchyme. Chacun de ces modèles inclut un composant géométrique et physique.

VI.III Intégration de la dynamique

Nos résultats expérimentaux sont en excellente concordance avec la théorie linéaire de stabilité quand les déformations sont raisonnablement bornées, laissant estimer la stabilité de la simulation basée sur les paramètres des ressorts. Nous avons ainsi prouvé qu'il y a une limite intrinsèque à la rigidité et à l'atténuation des ressorts. De plus, les valeurs de la haute rigidité permettent seulement une intégration stable quand le système est critiquement atténué (voir Figure 6.8).

VI.IV Résolution statique

Nous aurions pu choisir d'employer l'intégration implicite pour éviter les problèmes de rigidité. Cependant, nous estimons que la réponse passagère d'une analyse dynamique est suffisamment petite pour justifier une résolution directe de l'équilibre, ou équilibre statique. De plus, nous croyons que les conditions de bornes imposés au foie dans la cavité abdominale sont suffisantes pour imposer l'existence d'une seule configuration d'équilibre, nécessaire à n'importe quelle technique statique de résolution. Cependant, les non-linéarités (particulièrement dues à la rotation de l'organe) exigent une approche algorithmique appropriée. Par conséquent nous avons des variantes différentes de Newton-Raphson basées sur la résolution itérative d'un système linéaire en utilisant l'itération de SOR. En affichant graphiquement la configuration aux itérations intermédiaires nous créons l'illusion du comportement *quasi-dynamique* du foie pour une complexité de calcul inférieure au processus dynamique.

VI.V Changement de la topologie

Nous avons discuté la mise en place possible d'un algorithme de découpage dans notre simulateur. C'est d'importance primordiale dans une opération laparoscopique, d'où la nécessité de modeler ce phénomène. Les résultats préliminaires ont été donnés sur un algorithme approprié de découpage, cependant, un modèle du réseau vasculaire, essentiel pour un prototype de simulation chirurgicale, reste à mettre en application.

VI.VI Conclusions

Nous présentons un modèle du foie basé sur les données rhéologiques disponibles dans la littérature. Cependant, les techniques explicites d'intégration se sont avérées instables pour les propriétés élastiques et visqueuses enregistrées du foie. Par conséquent, nous avons constaté qu'une résolution statique non linéaire peut fournir une alternative intéressante, étant donné que la réponse passagère à une déformation du foie est négligeable.

VII Conclusions

VII.I Resumé

Nous avons examiné différentes approches en ce qui concerne la modélisation de la déformation d'un objet avec une géométrie complexe. En particulier nous avons comparé l'utilisation des réseaux masses-ressorts et des éléments finis tétraèdriques. Nous avons constaté que le comportement des réseaux masses-ressorts dépend intrinsèquement de la configuration topologique des ressorts. Ce problème résulte du fait que nous essayons de modeler un objet tridimensionnel par les éléments unidimensionnels. Les éléments finis volumétriques, d'autre part, ne souffrent pas de cette limitation. De plus, ils peuvent facilement modeler des phénomènes tels que l'incompressibilité. Cependant, ils exigent également plus de calcul par élément. Nous avons trouvé expérimentalement qu'un objet modelé avec les éléments finis tétraèdriques est environ quatre fois plus lent pour calculer les forces internes que le réseau masses-ressorts pour le même objet.

Pour la déformation d'un objet nous pouvons choisir entre une charge statique et une analyse dynamique. Une résolution statique peut être envisagée quand une seule configuration d'équilibre pour une contrainte externe donnée existe. Pour des petites déformations une résolution linéaire suffira, cependant, pour de plus grandes déformations ou des rotations une résolution non linéaire doit être exécutée. Alternativement, si nous sommes intéressés par le changement de la déformation en fonction du temps, ou aucune position d'équilibre n'existe, une analyse dynamique est nécessaire. Pour résoudre les équations résultant de cette résolution, soit l'intégration explicite ou implicite peuvent être employées. La stabilité des méthodes explicites est directement liée à la rigidité et à des coefficients d'atténuation du système physique, imposant une limite au pas de temps qui peut être pris. Les méthodes implicites ne souffrent pas de cette limitation mais peuvent exiger la solution d'un système non-linéaire.

La détection de la collision entre les modèles polygonaux complexes est une tâche chère et longue. Cependant, des méthodes basées sur la pénalité doivent être équipées d'information de collision à une vitesse rapide pour une réponse réaliste de collision. De plus, si les forces de collision sont envoyées à un dispositif à retour d'effort la sensation haptique sera mauvaise si la fréquence est basse. Pour éviter ce problème une approximation locale du contact est présentée. Puisque la géométrie du modèle local est beaucoup plus simple, la détection et la réponse de collision peuvent être exécutées à une fréquence beaucoup plus élevée, ayant pour résultat un affichage haptique amélioré.

VII.II Notre contribution

Un problème de base pour la simulation médicale est de modéliser la déformation avec un degré de réalisme suffisant, mais permettant la résolution interactive. Pour cela on a comparé les éléments finis tétraèdriques avec des maillages masse-réssort en termes de complexité de calcul et réalisme physique.

Dans le passé une résolution statique linéaire avait été proposé pour l'animation en temps-réel [Cotin, 1997]. Par contre ce type de résolution se limite à des petits déplacements. Pour cette raison on propose une résolution statique non-linéaire qui peut prendre en compte les grands déplacements. À notre connaissance cette méthode est tout à fait novatrice dans le domaine de l'animation interactive.

Pour une résolution dynamique des méthodes d'intégration explicites ou implicites peuvent être utilisées. On a analysé la stabilité de ces deux approches en fonction des paramètres physiques dans le contexte de la simulation temps-réel.

Un problème majeur pour la simulation médicale est l'identification des paramètres du modèle à partir des mesures expérimentales. On a abordé ce problème dans le contexte d'un simulateur échographique.

VII.III Horizons

Clairement il y a encore des possibilités pour améliorer les algorithmes utilisés pour la résolution des systèmes linéaires et non linéaires. En particulier une résolution multigrille pourrait offrir une alternative intéressante. De plus, des procédures pour simuler les systèmes physiques peuvent être parallélisées, en particulier les résolutions linéaires et le calcul des forces internes.

Par contre, mon point de vue est que c'est la prochaine grande augmentation de puissance de calcul qui permettra de résoudre les systèmes physiques viendra par le matériel sur mesure [Bishop and Kelliher, 2001]. De la même manière que les cartes 3D graphiques ont révolutionné les possibilités pour afficher des scènes complexes se composant de millions de polygones, je pense que les cartes physiques de simulation nous permettront de simuler des objets composés de millions d'éléments en temps réel sur des architectures d'ordinateur qui ne coûtent pas des millions de francs.

Chapter 1

Introduction

1.1 Motivations

When a student graduates from medical school he is supposed to have acquired the necessary theoretical background for his practice. However, in order to learn how to apply this knowledge, he is obliged to undergo a period of residency at an approved medical institution. Here the trainee will first observe procedures being carried out, before being allowed to practice on actual patients under the supervision of an experienced doctor.

Danger to patients A problem with this system is that an unexperienced trainee is more likely to make a mistake. For critical skills such as surgery this represents an obvious danger to the patient, even if an experienced practitioner is present; in other cases a wrong diagnosis may equally compromise his health.

Cost The current apprenticeship system may not only be dangerous to a patient, it is also extremely expensive. The use of the operating room, physicians bills, anaesthesia, and instrument cost need all to be taken into account. This financial overhead restricts the exposure trainees have to procedures. Thus, even over an extended period, the resident may only have performed a handful of challenging operations, and without encountering all of the different pathological cases.

Certification For procedural specialists such as surgeons, interventional radiologists, interventional cardiologists, and endoscopists, no adequate, objective measure of expertise or dexterity is available except for complication records from cases performed during the residency and fellowship [Dawson and Kaufman, 1998][1]

1.2 Objectives

Flight simulators have been in existence for many decades now. It has been proven that undergoing a simulation of a particular situation will help the pilot to make the right decision (i.e. not crash the plane) when confronted to it in reality.

Highly realistic simulators for laparoscopic interventions already exist. Surgeons practice complex procedures on pigs. This prepares the surgeon for using a new technique on a human, therefore reducing the risk of mistakes being made. However, besides

[1]citation from this paper: You are on an operating table, counting backwards from 100. You are getting sleepy. As you lose consciousness, you hear someone say: “Oh cool, my first bypass”.

ethical problems arising from animal rights, this approach is also expensive: the cost of sacrificing the pig, instruments, and the operating room persists.

It is our aim to replace the apprenticeship on humans and animals by a computer simulation of the procedure. The advantage of such a simulation system is that it could provide hands-on experience without endangering human patients at a much lower cost than animal training.

Geometric Models From three-dimensional anatomical datasets (such as described in [Ackerman, 1991]) geometric representation can be generated. When graphically displayed using a computer they can help students familiarize themselves with the human anatomy. They may "see" different organs without having to cut open a patient.

Physical Models Normally medical interventions involve changing the geometry of the anatomy, i.e. moving, deforming, cutting, etc. Hence physical models must be constructed that describe these phenomena [Delingette, 1998].

1.3 Some existing simulations

A virtual environment test-bed for training laparoscopic surgical skill has been presented in [Tendick et al., 2000]. The test-bed includes environments for training perceptual motor skills, spatial skills, and critical steps of surgical procedures. It discusses the need for real-time, accurate methods for modeling deformable tissue behavior that occur during the operation. Further it includes a four-DOF haptic interface that allows the user to feel the forces when deforming an organ.

In [Sagar et al., 1994] a model of an eye for surgical simulation was presented. The model has been designed to both visually and mechanically simulate features of the human eye by coupling computer graphic realism with finite element analysis.

[Kuhnapfel et al., 1995] presents a flexible tool for planning and training abdominal surgery. From CT and MRI-datasets a geometric model of the patient is obtained. By attaching a physical behavior to this representation one could perform an operation virtually before repeating the procedure on the real patient.

In [Jambon et al., 2000] a gynecologic simulator is presented. Emphasis is put on the real-time performance of the model by limiting the graphical expectation.

1.4 Overview

This document is organized into five chapters discussing the following topics:

Deformation Most of the human body is composed of soft tissue. Hence, for a medical simulator it is of paramount importance to model this phenomenon. We examine the mechanical theory of the deformation in a continuum and discuss different approaches to discretize these formulations in order to model an object of complex geometry. Especially mass-spring networks and tetrahedral finite elements are compared in terms of realism and computational complexity.

Resolution A physical model can approximate the internal forces in a body. These internal forces will be used to determine the deformation of the object when external constraints are imposed. We may perform a static resolution if an equilibrium configuration exists; we shall differentiate between linear and non-linear resolution. On the other hand, if we are interested in the change of the deformation with respect to time, we may prefer a dynamic analysis. In particular we examine explicit and implicit integration methods, and compare their performance in terms of stability and suitability in the context of a real-time application.

Interaction Unless we can interact with the physical model it will be sterile in a medical simulator. Hence, the first objective is to detect whether a collision has taken place. We analyze the existing approaches for this purpose. If a collision has been detected, a suitable response must be calculated. However, due to computational limitations this response can only be calculated at a limited rate. For complex models this rate might be too low to provide stable force-feedback. Hence, we propose to use a local approximation of the contact.

Echographic Simulator In this chapter we describe the implementation of a prototype of an echographic simulator of the human thigh. Its purpose is to train practitioners in the detection of a thrombosis in the vein. Hence, to accurately model the deformation of the thigh we have proceeded to a number of measurements. Based on this data we have constructed a model and identified its parameters in order to obtain the most realistic results. Further, we have experimented with several resolution techniques to find the most suitable one in the context of a real-time simulation. Lastly, we investigate the generation of echographic images according to the interaction with the virtual thigh.

Model of the Liver We present the model of a liver based on a mass-spring network. The different anatomical properties and rheological tests of the organ are discussed. Based on this information a heterogeneous and non-linear model is proposed. Further, we compare the static and dynamic analysis and discuss their relative merits. Finally, we offer the possible perspectives of this model as part of a laparoscopic simulator.

Chapter 2

Modeling deformable objects

Contents

Ut tensio sic vis.

Robert HOOK

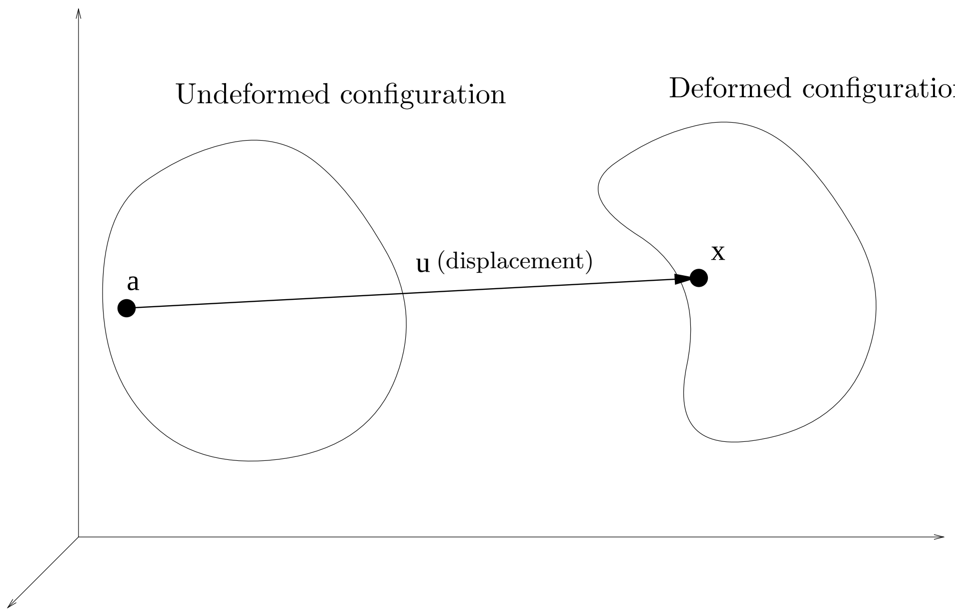


Figure 2.1: *Graphical representation of the deformation of an object. The material at location a in the undeformed configuration, is displaced to point x in the deformed configuration.*

2.1 Introduction

Much of the human body is deformable. Thus for most medical simulators it is essential to model this phenomena.

In Section 2.2 we define measures for deformation and relate them to the stress in the body. The field of mechanics has established a general theory in this respect, which we shall examine.

However, for complex geometries we cannot apply the continuum equations directly. Hence the object is divided into a set of smaller *elements*. In Section 2.3 we present different approaches and compare their realism and computational complexity.

2.2 Continuum Mechanics

Let us assume that we know the geometry of an object in its undeformed (or reference) configuration. Further, the vector a (*material coordinates*) denotes a point *within* the undeformed object. When we translate, rotate, or deform the object, the material at point a will have changed location (see Figure 2.1); we shall describe this new location by vector x (*world coordinates*). Thus the *displacement* u of the material is given by

$$u = x - a \tag{2.1}$$

For any point a in the material there exists a mapping towards its actual location in the deformed configuration given by $x(a)$. For a vector a that is outside of the material, the function is undefined. If we know this function we may also calculate the change of x with respect to a as

$$dx = F da \tag{2.2}$$

where F_{ij} is the *deformation gradient* defined as

$$F_{ij} = \frac{\partial x_i}{\partial a_j} \tag{2.3}$$

which is describes the change in the world coordinates with respect to a change in the material coordinates [Maurel et al., 1998]. The indexes i and j in Equation 2.3 define a particular value of vectors x and a. Therefore their range depends of the dimensionality of the problem.

2.2.1 Strain Tensor

Displacements alone do not necessarily mean there is a deformation. Rotation and translation do not alter the objects shape but do involve displacements. So how do we measure the deformation? The deformation gradient of Equation 2.3 is invariant to translation, but not to rotation since the derivative of vector x changes with respect to a. However, its magnitude does not! Hence we may describe deformation as the change in squared length of dx and da

$$dx^2 - da^2 = dx^T dx - da^T da \tag{2.4}$$

Rewriting this in terms of material coordinates only, according to Equation 2.2 we obtain

$$da^T F^T F da - da^T da = da^T (F^T F - I) da = da^T 2E da \tag{2.5}$$

where E is *Green's strain tensor* (also known as Lagrangian strain tensor) that is invariant to rigid body motions [Fung, 1994]. We may rewrite Equation 2.5 in terms of the partials as

$$E_{ij} = \frac{1}{2}\left(\frac{\partial x}{\partial a_i} \cdot \frac{\partial x}{\partial a_j} - \delta_{ij}\right) \tag{2.6}$$

where δ_{ij} is

$$\delta_{ij} = \begin{cases} 1 : i = j \\ 0 : i \neq j \end{cases}$$

We can also write Green's strain tensor as a function of the displacements.

$$E_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial a_j} + \frac{\partial u_j}{\partial a_i} + \frac{\partial u}{\partial a_i} \cdot \frac{\partial u}{\partial a_j}\right) \tag{2.7}$$

The Green strain tensor contains a quadratic term; this means that all large deformation analyses are *nonlinear*. However, if displacements are very small the quadratic term is negligible. Thus the *Cauchy's strain tensor* is a first order approximation of the Green strain tensor:

$$\epsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial a_j} + \frac{\partial u_j}{\partial a_i}\right) \tag{2.8}$$

It is important to remember that the gradients in the Green strain tensor are defined with respect to the reference configuration. Thus, all of the strain measures are also calculated with respect to the reference configuration. Alternatively, the *Almansi's strain tensor*, e (also known as Eulerian), builds a tensor with respect to the deformed configuration:

$$e_{ij} = \frac{1}{2}\left(\delta_{ij} - \frac{\partial a}{\partial x_i} \cdot \frac{\partial a}{\partial x_j}\right) \tag{2.9}$$

In the case of infinitesimal displacement, the distinction between the Green and Almansi strain tensor disappears: since displacements are supposed to be very small, it is irrelevant whether the gradients are calculated at the position of a point before or after deformation. For large deformations, however, they are different.

2.2.2 Strain-rate Tensor

In addition to the strain at given point we may calculate its rate of change by taking the derivative with respect to time. Thus, for Green's strain tensor E the strain-rate may be written as:

$$V_{ij} = \left(\frac{\partial x}{\partial a_i} \cdot \frac{\partial \dot{x}}{\partial a_j}\right) + \left(\frac{\partial \dot{x}}{\partial a_i} \cdot \frac{\partial x}{\partial a_j}\right) \tag{2.10}$$

Since V is derived from E, the strain-rate will also be calculated in the reference configuration.

2.2.3 Stress Tensor

The stress in a deformable body is related to the strain and the strain-rate. This relationship is determined by the elastic and viscous properties of the material. Hence, the *2nd Piola-Kirchoff stress* tensor S is composed of the elastic stress, $S^{(E)}$, and the viscous stress, $S^{(V)}$:

$$S = S^{(E)} + S^{(V)} \tag{2.11}$$

The elastic stress, $S^{(E)}$, depends on the Green's strain tensor (Equation 2.6), and the viscous stress, $S^{(V)}$, is a function of the strain-rate (given in Equation 2.10):

$$S_{ij}^{(E)} = \sum_{k=1}^{3} \sum_{l=1}^{3} C_{ijkl} E_{kl} \tag{2.12}$$

$$S_{ij}^{(V)} = \sum_{k=1}^{3} \sum_{l=1}^{3} D_{ijkl} V_{kl} \tag{2.13}$$

The relation between the stress and the strain is given by the stiffness matrix C. In three dimensions both strain and stress tensors consist of nine values. Hence to fully describe how each of the strain values affects the stress it requires $9 \times 9 = 81$ coefficients. Similarly the damping matrix D is made up of the same number of viscous coefficients. However, since E is symmetric most of the coefficients of C are redundant, and can be reduced to $6 \times 6 = 36$ different values. If on top of this the material is isotropic, the

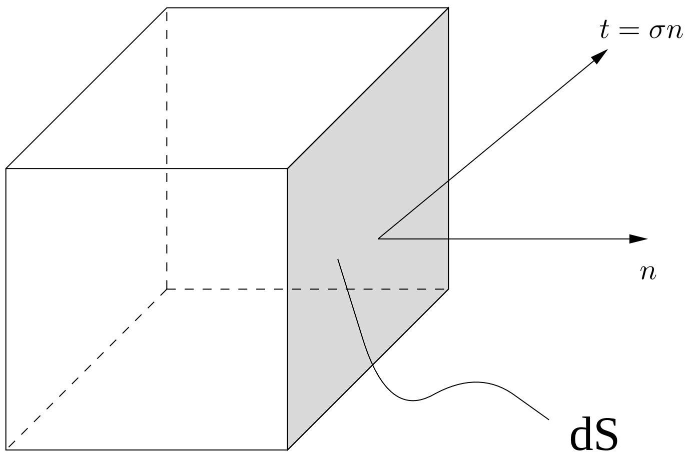


Figure 2.2: *The traction t acting on the shaded face of the cube having normal n and surface area dS.*

relationship can be fully described by only two values, μ and λ, named as the Lamé coefficients. The stress tensor is then calculated as follows.

$$S_{ij}^{(E)} = \sum_{k=1}^{3} \lambda E_{kk} \delta_{ij} + 2\mu E_{ij} \qquad \delta_{ij} = \begin{cases} 1 : i = j \\ 0 : i \neq j \end{cases} \tag{2.14}$$

The rigidity of the material is then determined by μ while the resistance to a change in volume (dilatation) is given by λ [O'Brien and Hodgins, 1999].

Since the strain tensor is symmetric, the strain-rate tensor will also be symmetric. Thus, in the same way, we may reduce the viscous matrix D to the two values ϕ et ψ.

$$S_{ij}^{(V)} = \sum_{k=1}^{3} \phi V_{kk} \delta_{ij} + 2\psi V_{ij} \qquad \delta_{ij} = \begin{cases} 1 : i = j \\ 0 : i \neq j \end{cases} \tag{2.15}$$

Given the stress we can calculate the *traction*, which is defined as the force per unit area. If n is a vector perpendicular to the surface dS (see Figure 2.2), then the traction is given by

$$t = \sigma n \tag{2.16}$$

where σ is the Cauchy stress in the *deformed* configuration. The 2nd Piola-Kirchoff tensor gives the stress in the reference configuration because it is calculated from strain

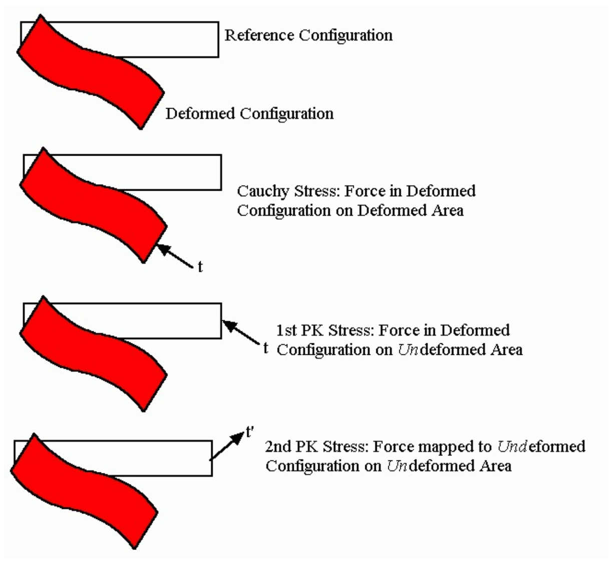

Reference Configuration
Deformed Configuration
Cauchy Stress: Force in Deformed Configuration on Deformed Area
t
1st PK Stress: Force in Deformed Configuration on *Un*deformed Area
t
t'
2nd PK Stress: Force mapped to *Und*eformed Configuration on *Un*deformed Area

Figure 2.3: *Different stress tensors and the resulting traction. (taken from http://www.engin.umich.edu/class/bme456/largedef/largedef.htm)*

measures in the reference configuration (see Section 2.2.1). The transformation from the 2nd Piola-Kirchoff stress tensor S to the Cauchy stress tensor σ is given by

$$\sigma_{jr} = \sum_{i=1}^{3}\sum_{k=1}^{3} \frac{\partial x_i}{\partial a_j} S_{ik} J^{-1} \frac{\partial x_r}{\partial a_k} \tag{2.17}$$

where J^{-1} is the inverse determinant of the deformation tensor [Bathe, 1982]. We now have derived the 2nd Piola Kirchoff stress tensor which is symmetric and energetically consistent with the Green-Lagrange strain. In other words, the strain energy density calculated using the 2nd Piola Kirchoff stress tensor with the Green-Lagrange strain will be the same as that calculated with the Cauchy stress tensor and Almansi-Euler deformation strain tensor:

$$\sigma_{ij} e_{ij} = S_{ij} E_{ij} \tag{2.18}$$

The advantage is that the expression on the right is calculated in the reference configuration while that on the left is calculated in the deformed configuration (see Figure 2.3).

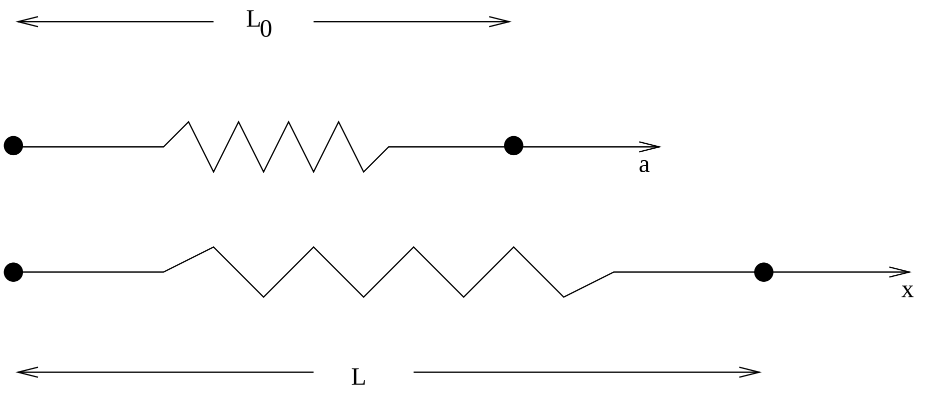


Figure 2.4: *Deformation of a spring.*

2.2.4 Constitutive Relationship

So far we have examined a *linear* relationship between the strain and the stress. Materials that obey this criteria are called “hookean elastic solids”. However, materials in the real world cannot usually be described by this hypothesis since the relationship between strain and stress is non-linear. A linear approximation is only applicable in a small range of strain. Beyond this strain, it is important to take into account the non-linear behavior if an accurate model is our aim.

For a three-dimensional object we may obtain the stress tensor as the partial derivative of the strain energy per unit volume with respect to the Green strain tensor [Fung, 1994]:

$$S_{ij} = \frac{\partial \rho_0 W}{\partial E_{ij}} \qquad (i, j = 1, 2, 3) \tag{2.19}$$

where ρ_0 is the density of the material in the initial state, and W the strain energy. If the material is *incompressible* the stress is independent of the deformation. In this case a pressure term should be added to the right-hand side of Equation 2.19.

2.3 Computational Models

2.3.1 Mass-Spring Models

Consider a spring at rest, with no forces applied (see Figure 2.4) the initial length is L_0. Now if the spring is stretched its actual length is defined by L.

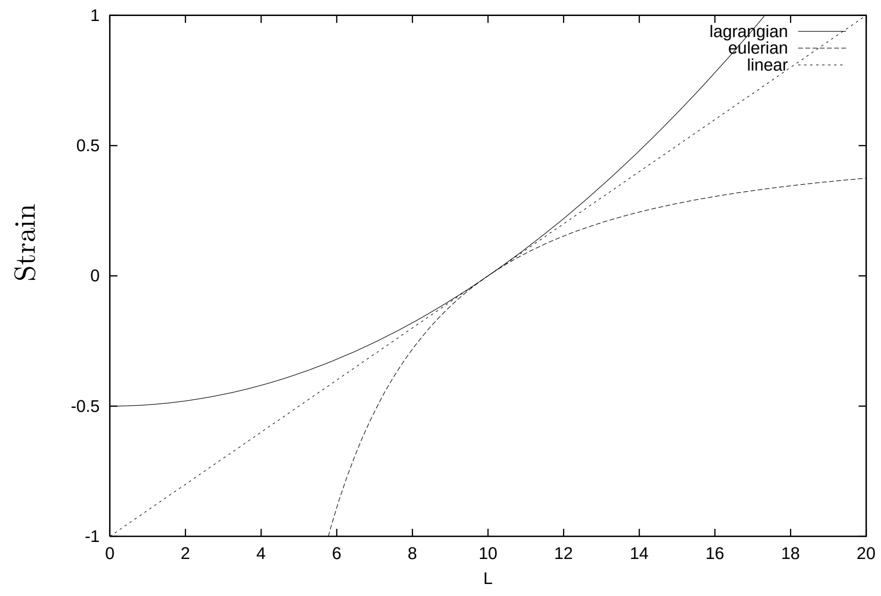


Figure 2.5: *Lagrangian (or Green) strain and eulerian (or Almansi) strain in function of the actual length of the spring. The initial length, L_0, is* 10.

The spring models only the material between two particles. The material coordinates are thus one-dimensional and the Green strain tensor (Equation 2.6) collapses to a scalar.

$$\epsilon = \frac{1}{2}\left(\frac{\partial x}{\partial a} \cdot \frac{\partial x}{\partial a} - 1\right) \tag{2.20}$$

In this case the deformation can also be expressed in terms of the distance, L, between the two particles.

$$\epsilon = \frac{L^2 - L_0^2}{2L_0^2} \tag{2.21}$$

In the same way Almansi's strain can we formulated as

$$e = \frac{L^2 - L_0^2}{2L^2} \tag{2.22}$$

In Figure 2.5 Green's and Almansi's stain measures are compared as a function of the length L. Notice the two measures have similar behavior when deformation is small, but different when deformations are large.

Having calculated the strain in the spring, we can estimate the stress. Hook's law states that the internal stress, σ, in a spring is proportional to its strain, ϵ:

$$\sigma = \mu\epsilon \tag{2.23}$$

where μ is the rigidity of the spring. This hypothesis can be verified experimentally, but only for small deformations. For larger deformations the relationship is no longer linear, i.e. does not obey Hook's law. For this reason works like [Terzopoulos and Waters, 1990] and [d'Aulignac et al., 1999a] make use of non-linear springs.

In addition to the elastic stress, the spring can exhibit viscous behavior. This is dependent on the change of deformation with respect to time, ν:

$$\nu = \frac{\partial\epsilon}{\partial t} = \frac{\partial\epsilon}{\partial L}\frac{\partial L}{\partial p}\frac{\partial p}{\partial t} = \frac{1}{L_0^2}\, p^T\, v \tag{2.24}$$

where p and v are respectively the relative position and velocity between the two particles. The full stress of the spring is, therefore, given by

$$\sigma = \mu\epsilon + \psi\nu \tag{2.25}$$

where μ is the elasticity and ψ the viscosity of the spring.

2.3.1.1 Internal Forces

To calculate the forces in three-dimensional space acting on the particles on either side of the spring we first calculate the normalized vector between their positions as

$$n = \frac{p_1 - p_2}{L} \tag{2.26}$$

Then the force on the first particle will be dependent on the cross-sectional area A of the spring.

$$f_1 = n\sigma A \tag{2.27}$$

Since each force has an equal and opposite reaction, the force on the second particle is given as

Operation	Add	Mult	Div	Sqrt	FLO	Running Total
Strain	6	4	1		11	11
Strain-rate	5	3	1		9	20
Stress	1	2			3	23
Internal Forces	3	4	3	1	11	34
Element Stiffness Matrix	9	15	1		25	59
Total	24	28	6	1	59	

Table 2.1: *Summary of the floating point operations (FLO) necessary to compute strain, stress, internal forces and stiffness matrix for spring.*

$$f_2 = -n\sigma A \tag{2.28}$$

2.3.1.2 Elementary Stiffness Matrix

The stiffness matrix indicates how much the internal force on a particle will change when its position is modified. We may express this relation as follows[1]

$$\left[\frac{\partial f}{\partial p}\right]_{3\times 3} = \frac{A\mu}{L}\left((1+\epsilon)n\, n^T - \epsilon I\right) \tag{2.29}$$

where I is the identity matrix. This yields the elementary stiffness matrix of the spring:

$$K_{6\times 6} = \begin{bmatrix} \frac{\partial f_1}{\partial p_1} & \frac{\partial f_1}{\partial p_2} \\ \frac{\partial f_2}{\partial p_1} & \frac{\partial f_2}{\partial p_2} \end{bmatrix} \tag{2.30}$$

2.3.1.3 Experimental Results

In Table 2.1 we have counted the number of floating point operations (FLO) needed for each stage towards calculating the forces and the stiffness matrix. The number of operations are calculated assuming that values calculated in previous stages are reused. For example, the normal vector does not have to be recalculated for the evaluation of the stiffness matrix since it has already been used to find the internal forces.

[1]Thanks to Sepanta Sekhavat for deriving this expression on the back of a pizza box.

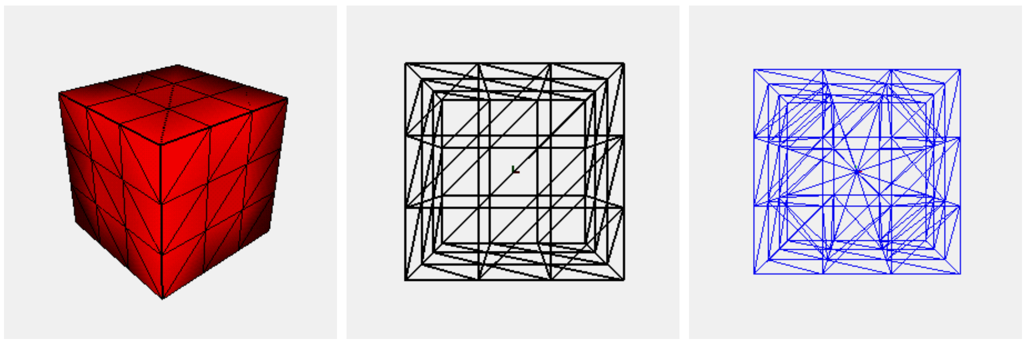


Figure 2.6: *The cube on the left is composed of a set of triangles on the surface (middle) and a set of tetrahedrons (right)*

Surface Points	56
Faces	108
Total Points	57
Tetrahedrons	120
Springs	230
Force Eval per sec.	3440
Elements per sec.	791200

Table 2.2: *Experimental spring-damper network implementation results for the cube shown in Figure 2.6.*

In Figure 2.6 we see how the polygonal surface representation of a cube may be used to decompose the object into a set of tetrahedrons. In this case the software by [George and Borouchaki, 1998] was used. Then for every edge of a tetrahedron a spring is added between the two points joined by the edge. However, many tetrahedrons share the same edge. In such cases only one spring is created. Thus the object is modeled as a set of elastic bars joined by point hinges. Table 2.2 summarizes the characteristics of the cube and the experimental results for the implementation.

To test the quality of the deformation of the cube we apply a uniform force on all the particles while the backside is fixed. For a cube made of an elastic material that is isotropic and homogeneous, we would expect the cube to stretch in the direction of the force. However, there is also an unnatural twist (see Figure 2.7). This is because the physical behavior is dependent on the the geometrical representation. In this case, the diagonal springs on the surface are all parallel to each other, making it easier for the cube to deform in one way than in the other, thus twisting it.

A considerable problem when using spring-damper networks, is that the deformation

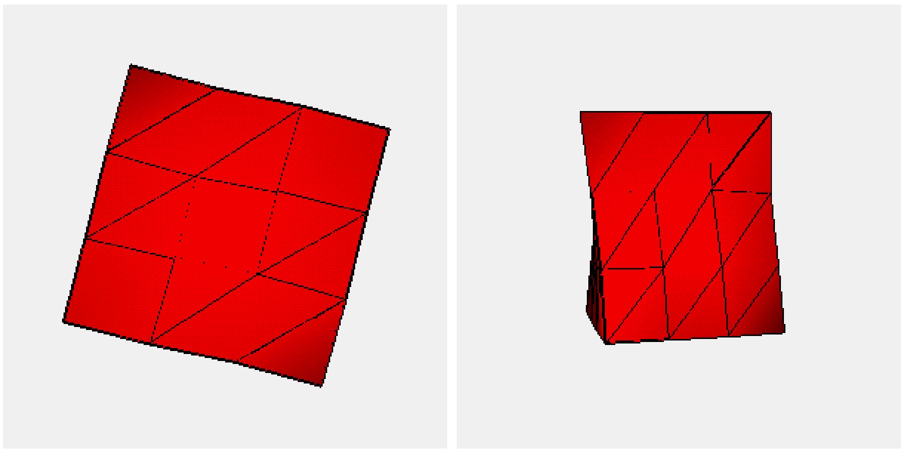


Figure 2.7: *Deformation of the cube, modeled with a set of springs, in a forcefield. The particles on the back of the cube are immobile. The force pulling the other particles forwards produces an unnatural twist of the cube, due to the disposition of the springs.*

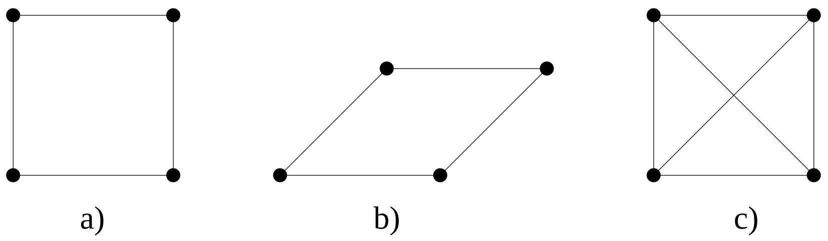


Figure 2.8: *Mass-Spring network a) rectangular b) invariant to shearing c) with additional diagonal springs to avoid shearing.*

is dependent on the configuration of its connectivity. Consider the case of a rectangle modeled by point masses on its corners connected by springs on the edges. The springs will try to maintain the initial length of the edges, but not the surface. As seen in Figure 2.8 the rectangle is invariant to shearing. In order to avoid this phenomena it is possible to add diagonal springs. However, this increases the computational complexity. Alternatively, [Joukhadar and Laugier, 1996] introduces the use of angular springs, aiming at keeping a constant angle between three particles.

The difficulty of modeling surfaces using springs has been addressed by [Van Gelder, 1998], who demonstrates that an exact simulation of elastic membranes by triangulated spring meshes is impossible. However, he provides a methodology for an approximate solution.

In [Boux de Casson et al., 2000] we examine the global behavior of a three-dimensional object physically modeled by mass-springs. A cylinder is discretised into

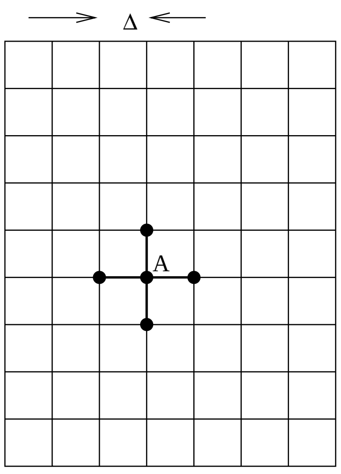


Figure 2.9: *Regular grid with spacing Δ used to evaluate the deformation gradient at point A by finite differencing.*

a set of tetrahedra. The mass is distributed on the vertices in function of the volume of the tetrahedron and springs are placed on their edges. A traction is applied to the bottom of the cylinder while the top remains fixed. Its total elongation corresponds well with the individual behavior of one spring.

In [Terzopoulos and Waters, 1990] a spring lattice is used to model facial deformation to model several human emotions. This procedure was then extended in [Lee et al., 1995] in order to make use of laser range data for the geometric reconstruction of faces.

2.3.2 Finite Differences

One way to find the deformation gradient necessary to compute the strain (Equation 2.6) is finite differencing. Given a regular grid (Figure 2.9) we want to find the gradient at point A.

$$\frac{\partial x}{\partial a_i} = \frac{x(a+\Delta_i) - x(a-\Delta_i)}{2\Delta_i} \tag{2.31}$$

However, this approach assumes a regular discretisation of the space. [Debunne et al., 1999] have extended the procedure to irregular grids.

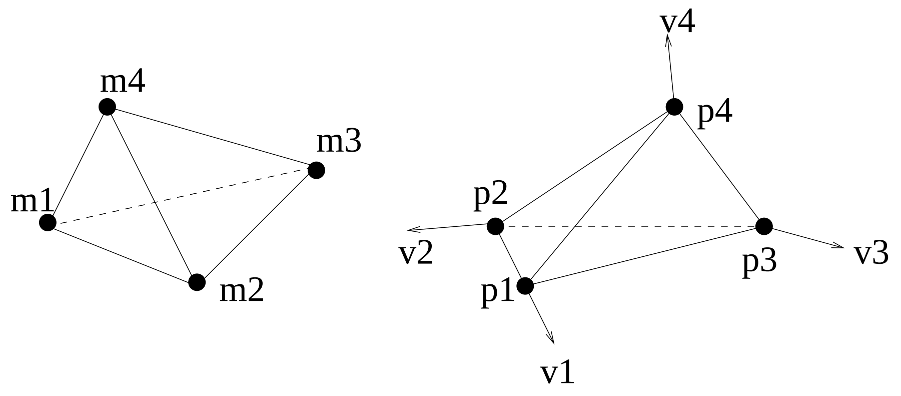


Figure 2.10: *On the left position of the nodes of a tetrahedron in reference configuration given by vector m. On the right positions and velocities of the nodes in the real world configuration given by vectors p and v respectively.*

2.3.3 Finite Element Method

The finite element method (FEM) decomposes a continuum into smaller building blocks, or elements, of various shapes. For example, a membrane may be represented by a set of triangles. Then for each triangle we can solve the continuum equations of Section 2.2.

2.3.3.1 Tetrahedral Elements

The simplest volume element is the tetrahedron. The volume is defined by four nodes and edges between every pair of points. We shall define the position of the nodes in the reference configuration as m. Hence these are the positions of the nodes *before* deformation. The position of the nodes in world coordinates, i.e. after deformation or rigid body motion, is given by vector p. Similarly the velocity of the nodes are represented by vector v in the world configuration (see Figure 2.10).

The position of a point a in the tetrahedron's reference configuration, given the barycentric coordinates, b, can be obtained through linear interpolation as follows [O'Brien and Hodgins, 1999, Delingette et al., 1999].

$$\begin{bmatrix} a \\ 1 \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} b$$

We can use the same interpolation technique between the positions and velocities in the deformed configuration.

$$\begin{bmatrix} x \\ 1 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} b$$

$$\begin{bmatrix} \dot{x} \\ 1 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} b$$

Hence to find the barycentric coordinates, b, for a location in the reference configuration

$$b = \beta \begin{bmatrix} a \\ 1 \end{bmatrix}$$

where β is

$$\beta = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1}$$

If we make the assumption that the same barycentric coordinates represent the same point of material we can write the position in the world configuration as a function of the material coordinates

$$x(a) = P\beta \begin{bmatrix} a \\ 1 \end{bmatrix}$$

$$\dot{x}(a) = V\beta \begin{bmatrix} a \\ 1 \end{bmatrix}$$

where P and V are defined as

$$P = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix}$$

$$V = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}$$

To compute the values of the strain tensor (Equation 2.6) and strain-rate tensor (Equation 2.10) we need the derivative of x with respect to a, or, in other words, the change in world coordinates when we move in material coordinates. Hence,

$$\frac{\partial x}{\partial a_i} = P\beta\delta_i \tag{2.32}$$

$$\frac{\partial \dot{x}}{\partial a_i} = V\beta\delta_i \tag{2.33}$$

2.3.3.2 Internal Forces

Using the partials found above we can calculate the strain tensor, and consequently the stress in the element. Once we have these we can compute the internal elastic energy on the material, W_{int}, as

$$W_{int} = \frac{1}{2} \int_V E^T \; S \tag{2.34}$$

Then, the internal force, f_{int}, on each node of the tetrahedron can be expressed as the partial of W_{int} with respect to the position

$$f_{int} = \frac{\partial W}{\partial p} \tag{2.35}$$

as explained in [Cotin et al., 2000]. Rewriting we obtain

$$f_{int} = \int_V \; B^T \; S \;\; dV \tag{2.36}$$

As the change in force is dependent on the change in strain, we define B as the change in strain with respect to displacement. Due to symmetry the strain tensor E only has six different values. Thus

$$\frac{\partial E_{6\times1}}{\partial u_{12\times1}} = B_{6\times12} \tag{2.37}$$

where u is the displacement. Since the Green tensor is non-linear, the partial is non-linear as well. Thus B is a linear approximation at the current displacement.

$$\frac{\partial E_{ij}}{\partial P_{kl}} = \left(\beta_{li} \sum_{p=1}^{4} P_{kp}\beta_{pj} \right) + \left(\beta_{lj} \sum_{p=1}^{4} P_{kp}\beta_{pi} \right) \tag{2.38}$$

if $i = j$ then this simplifies to

$$\frac{\partial E_{ii}}{\partial P_{kl}} = 2 \left(\beta_{li} \sum_{p=1}^{4} P_{kp}\beta_{pi} \right) \tag{2.39}$$

2.3.3.3 Elementary Stiffness Matrix

Let us now find the tangent stiffness matrix of the element, that is the change in internal force on the nodes when one of the positions of the nodes changes. This matrix is denoted as K and is of size 12×12 for a four-node tetrahedron, since each node has three degrees of freedom. Thus,

$$K_{12\times12} = \int_V B^T_{12\times6}\, C_{6\times6}\, B_{6\times12}\ dV \tag{2.40}$$

where C denotes the 36 elastic coefficients relating strain to stress [Delingette et al., 1999].

2.3.3.4 Mass Matrix

The mass matrix is a discrete representation of a continuous distribution of mass. A *consistent* element mass matrix is defined as

$$M = \int_V \rho\, b\, b^T\ dV \tag{2.41}$$

where ρ is the mass density of the material and b the barycentric coordinates. A simpler formulation is the *lumped* mass matrix, which is obtained by assembling the mass at the nodes of the element. For example the total mass of the element may be calculated and distributed equally in function of the number of nodes. Alternatively several lumping schemes exist that convert a consistent mass matrix into a lumped one [Cook et al., 1989]. As for efficiency, lumped mass matrices are simpler to form, occupy less storage space, and require less computational effort (see Section 3.4).

2.3.3.5 Experimental Results

Table 2.3 summarizes the number of floating point operations necessary for each stage towards computing the internal forces and the stiffness matrix. We can see that the number of operations necessary to compute the internal forces for one tetrahedral element as compared to a spring is approximately 14 times more. However, the number of elements discretizing the cube in Figure 2.6 being less, evaluation of the forces is only slower by a factor of 4 (see Table 2.4). It should be noted that for the evaluation of the

Operation	Additions	Multiplications	FLO	Running Total
Green's Strain tensor	42	54	96	96
PK2 Stress tensor	18	42	60	156
Internal Forces	156	248	404	560
Element Stiffness Matrix	1084	1300	2384	2944
Total	1300	1644	2944	

Table 2.3: *Summary of the floating point operations (FLO) necessary to compute strain, stress, internal forces and stiffness matrix for a tetrahedral element.*

Surface Points	56
Faces	108
Total Points	57
Tetrahedrons	120
Elements	120
Force Eval per sec.	890
Elements per sec.	106800

Table 2.4: *Experimental FEM implementation results for the cube shown in Figure 2.6.*

force, the strain and stress tensors are updated every time. This corresponds rather well with the factor 5 found in [Picinbono et al., 2001].

The deformation of the cube does not present the problems experienced with spring-damper networks: there is no twist when pulling at the bottom nodes, and volume conservation can be controled (Figure 2.11). Further [Debunne et al., 2001] demonstrates that behavior is independent of the discretisation (assuming that there are no badly conditioned tetrahedra).

2.3.4 Particle Systems

The basic idea behind particle systems is to decompose the object into a set of particles that can interact with each other. Many different types of interaction have been proposed. One of the most commonly exploited is the Lennard-Jones inter-particle relationship as used in [Miller and Pearce, 89].

$$f(r) = \frac{A}{r^{12}} - \frac{B}{r^6}$$

where A and B are constants and r the distance between the two particles. The

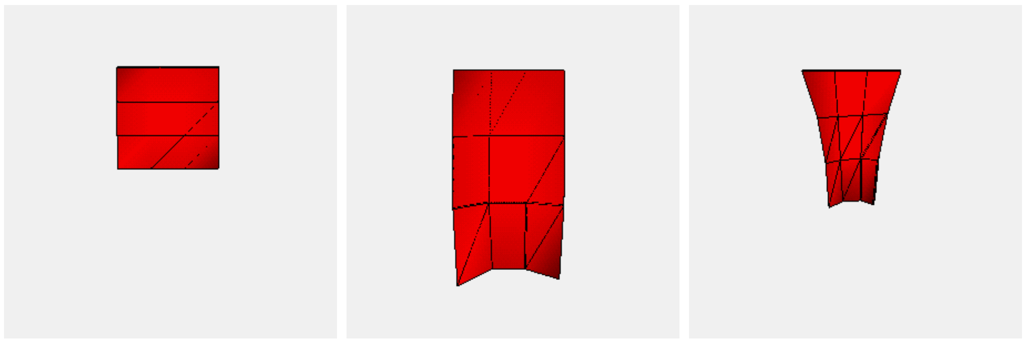


Figure 2.11: *Forces applied to the bottom nodes of the cube (modeled with the FEM) at rest (left), with small λ (middle), and $\lambda = 0.01$ (right).*

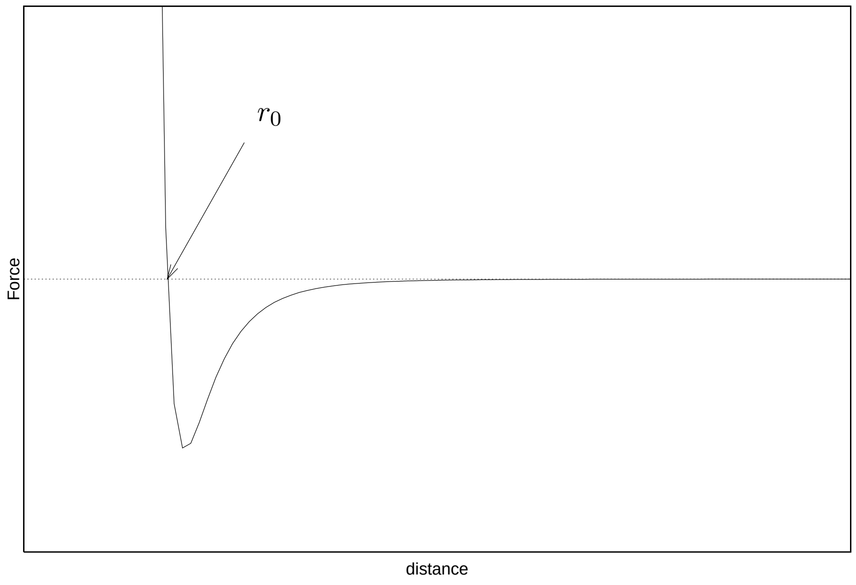


Figure 2.12: *Relation inter-particulaire de Lennard-Jones.*

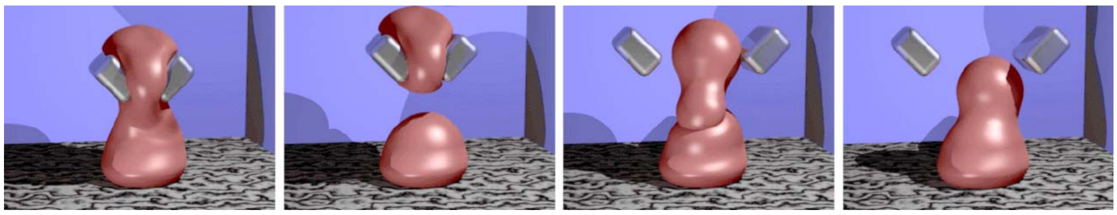


Figure 2.13: *Particle system rendered with an implicit surface (Source: [Desbrun, 1997]).*

force is a function of the distance as can be seen in Figure 2.12. We can thus see that as the distance approaches zero, the force trying to separate the particles tends towards infinity. Inversely, if there is a large distance between the particles, then the force is very small. Equilibrium is found at distance r_0.

Alternatively [Luciani et al., 1991] propose to place a sphere around every particle to protect it from inter-penetration with other particles. The particles attract each other but the spheres hinder this attraction beyond a given radius.

For physical realism [Desbrun, 1997] uses the Smoothed Particles Hydrodynamics (SPH) to approximate a continuous material. This approach is relatively independent from the discretisation used. The resolution of the object is thus adapted according to the deformation in order to optimize computation with a minimal loss in accuracy. Further, implicit surfaces are used for graphical representation.

The absence of a fixed neighborhood is advantageous when splitting an object in two; no change to the model is necessary. However, maintaining a given structure is problematic since the particles tend to regroup in a heap if they are not constrained by external forces.

Also the evaluation of the forces is computationally expensive. The force acting on a particle depends on the interaction with *all* other particles, giving rise to a quadratic complexity. Although several methods exist that reduce this complexity, the evaluation remains relatively large.

The lack of a fixed topology in particle systems is both its main advantage and primary limitation. Fluids, for example, are a good candidate for particle modeling. However, objects with a fixed structure, such as parts of the human anatomy, are difficult to model with such an approach.

2.3.5 Constraints

In the previous sections we have looked at how for a given deformation the internal force is calculated. However, for rigid articulated bodies there is no deformation of any link, but the degrees of freedom are nevertheless *constrained* [Witkin, 1999]. While optimization of the solution of loop-free systems is given by [Featherstone, 1983], [van Overveld and van Loon, 1992] proposes an iterative solution of the system achieving real-time performance for several tens of constraints, including constraints that introduce loops. A non-iterative solution by [Baraff, 1996] is based on Lagrange multipliers, and solves a system with n loop free constraints and k constraints that introduce loops at a cost of $O(kn)$.

In [Gascuel and Gascuel, 1992] the constraint forces are not explicitly calculated and are not incorporated into the dynamic equations. They are approximated by iterative tunings in the displacement of the constrained objects. [Palazzi, 1993] uses displacement constraints on different levels of refinement to control the level of global and local deformation; this approach is similar to that of multi-grid solvers for partial differential equations.

However, the use of constraints is not limited to rigid bodies. [Provot, 1995] uses displacement constraints in conjunction with mass-spring networks to simulate cloth. To avoid overly stretchy cloth the maximum distance between any two particles is limited to a given threshold. Within the admissible distance the springs operate normally, but if the distance is greater than the maximal distance, displacement constraints are used to instantaneously restore the maximal distance between the particles. Using an iterative method the system converges in practice.

A similar approach is taken by [Gibson, 1997] where minimal *and* maximal distances are imposed between a particle and its neighbors on a regular grid. The constraints are imposed using a specially tailored iterative algorithm where only the part of the object which is being deformed is updated. This leads to a very fast update, even for many particles, when the deformation is local. In addition particles are connected by springs, which force the object back into its original configuration after having applied the kinematic constraints (N.B. this relaxation carried out when timing constraints allow it).

2.4 Conclusions

In this chapter we have introduced mechanical measures for strain, which enable us to quantify the deformation. Further we establish the link between this deformation and the stress present inside a body. From this we may calculate the internal forces due to the deformation.

Nevertheless the measures for strain and stress are not constant throughout a complex object. This obliges us to divide the object into a set of simpler geometrical entities for which we may solve the continuum equations. Several approaches exist towards this purpose.

Mass-spring networks have in the past been a very popular approach towards modeling deformation, due to their easy implementation and relatively low computational complexity. However, the primary problem with this approach is that the physical behavior of the model is intrinsically dependent on the topological description of the network. A further issue with this method is that volume conservation is difficult to enforce; however, [Costa Ferreira and Balaniuk, 2001] have presented promising results towards the solution of this problem.

In a way the springs used in mass-spring networks, are one-dimensional *elements* that model the material between two points. However, when one-dimensional elements are used to model a three-dimensional continuum, topological dependencies arise. The finite element method (FEM) allows the use of volumetric elements (such as tetrahedrons), that will allow a physical model of an object that is independent of its topology. Further, phenomena such as volume conservation can be easily enforced.

Until recently non-linear resolution of the FEM has been considered too computationally expensive for real-time applications. This myth is now being disproved by works such as [Zhuang and Canny, 2000], [Wu et al., 2001], and [Picinbono et al., 2001] that have used the FEM as part of an interactive application. It is, however, true that the FEM demands more calculations than the mass-spring network for the same geometrical discretisation. We have determined experimentally that the internal forces for an object modeled with tetrahedral finite elements takes approximately four times longer to calculate than using a mass-spring network.

Chapter 3

Resolution of static and dynamic systems

Contents

So many flops,
So little time...

Lynn KILLINGBECK

3.1 Introduction

In the last chapter we have seen how to calculate the internal forces of an object due to a deformation. In this chapter we shall discuss how to find the configuration of the deformable body given an externally applied force.

If there exists a unique equilibrium configuration for the applied force we may resort to a static resolution. Given the change of the internal forces with respect to a change of the configuration (i.e. the stiffness matrix) of the body is linear, or may be approximated by a linear function, then we may proceed to a linear resolution of the *static* deformation (see Section 3.2). However, if the deformation is large enough that the change in the stiffness matrix can no longer be ignored, we must perform a computationally more intensive non-linear resolution (see Section 3.3).

Static resolution does not provide a time-history of the deformation. By this we mean that an equilibrium configuration is found (given it exists) without providing any information about how it got there. We may be able to find the change of the configuration with respect to time by the means of a *dynamic* resolution (see Section 3.4). In other words this equates to an integration of non-linear differential equations. Notably we examine explicit and implicit integration, and compare their performance in terms of accuracy, stability, and computational complexity.

Both static and dynamic resolution techniques may require the solution of a linear system of equations (Section 3.5). As direct methods such as Gaussian elimination are extremely expensive we examine several iterative methods that are well suited for the solution of the sparse systems arising from configuration of the deformable objects.

A classical solution for solving a non-linear system is the iterative solution of linear systems. Unfortunately this technique is computationally intensive. In Section 3.6 we investigate the possibility of using a multi-layer neural network to approximate directly the solution to a non-linear system.

3.2 Linear Static Resolution

Let u be the displacement vector for a given object. Given that we can calculate the internal forces for a known displacement, the change in force with respect to the displacements is expressed by the stiffness matrix K

$$\frac{\partial f_{int}}{\partial u} = K \tag{3.1}$$

Since at equilibrium the internal and external forces perfectly balance we can state

$$Ku = f_{ext} \tag{3.2}$$

Thus it is possible to obtain the displacement for a known external force, f_{ext}, after inversion of the stiffness matrix.

$$u = K^{-1} f_{ext} \tag{3.3}$$

The size of the stiffness matrix K is $n \times n$ where n is the number of degrees of freedom (DOF) of the system. Hence this matrix can be large, but is normally sparse (since the change in force is dependent only on the neighboring particles). However, the inverse is not. If enough memory is available the inverse can be precomputed and stored. Then it is only necessary to multiply it by the external forces to obtain to displacements.

If on a volumetric object only the external nodes are of interest the computational complexity may be reduced by the approach taken by [Bro-Nielsen and Cotin, 1996]. They apply a process known as condensation to reduce the size of the matrix to $v \times v$ where v are the DOF of the surface nodes only. In addition the matrix multiplication is only done for values of $f_{ext} \neq 0$ thereby further reducing the complexity, which then grows linearly depending on the number of nodes where forces are applied. This approach has strong similarities to the pre-calculation approach proposed in [Cotin et al., 1999].

However, if the stiffness matrix is modified (e.g. change of boundary conditions or topology) its inverse will change. For small changes [James and Pai, 1999] propose using the Sherman-Morrison formula to update the inverse rather than recomputing. This assumes that there is enough memory available to store the inverse.

Figure 3.1 shows examples of the deformations that can be calculated using linear elastic models. The back of the cube is fixed making rotation impossible. Hence, if the forces applied do not cause large deformations, the stiffness matrix will not change very since the displacements are small. However, if we only fix one edge of the cube (Figure 3.2) the cube will be able to rotate inducing large displacements. This means that the stiffness matrix will change much, invalidating the realism of a linear resolution.

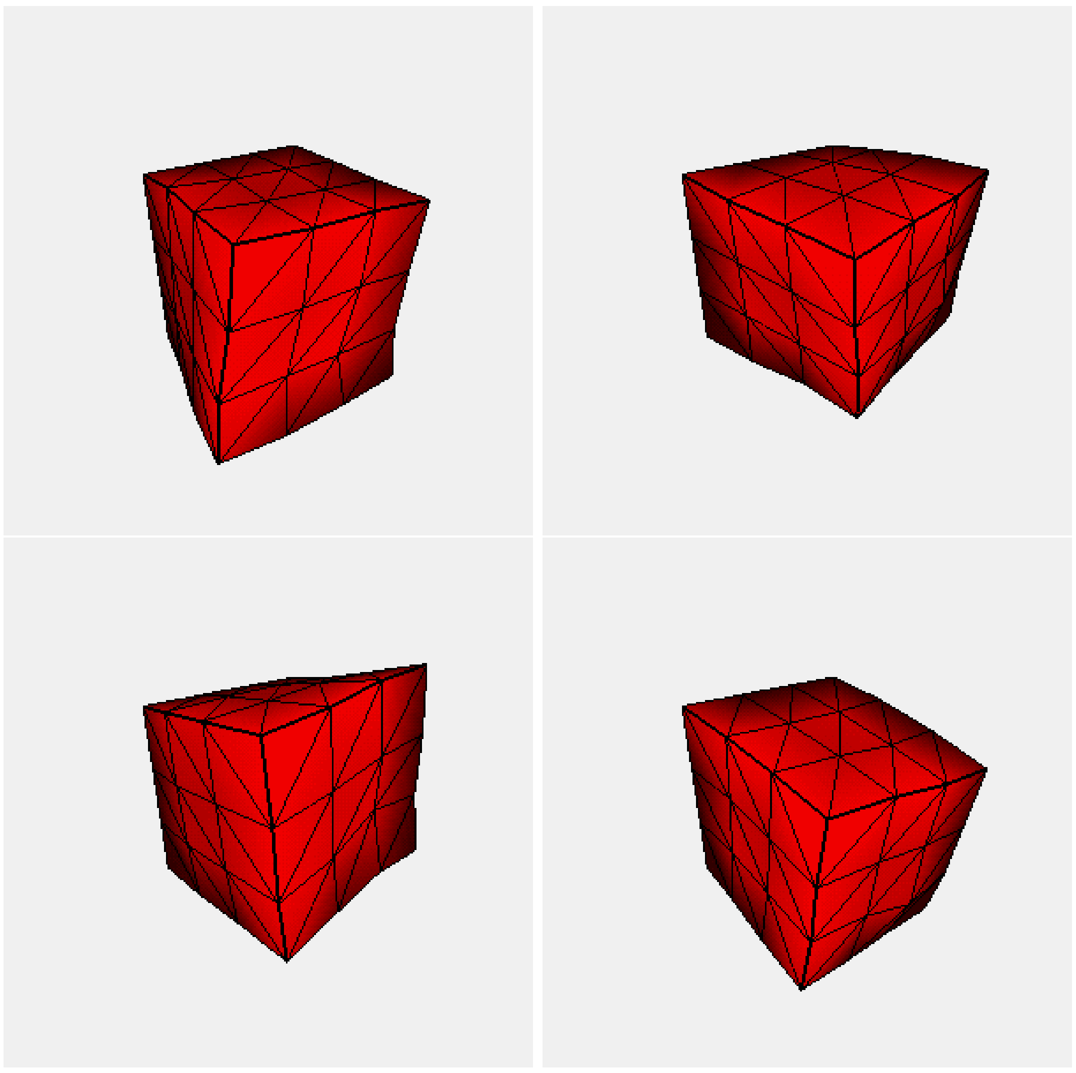


Figure 3.1: *Examples of linear deformations by a point force acting left, right, up, and down. The back of the cube is fixed.*

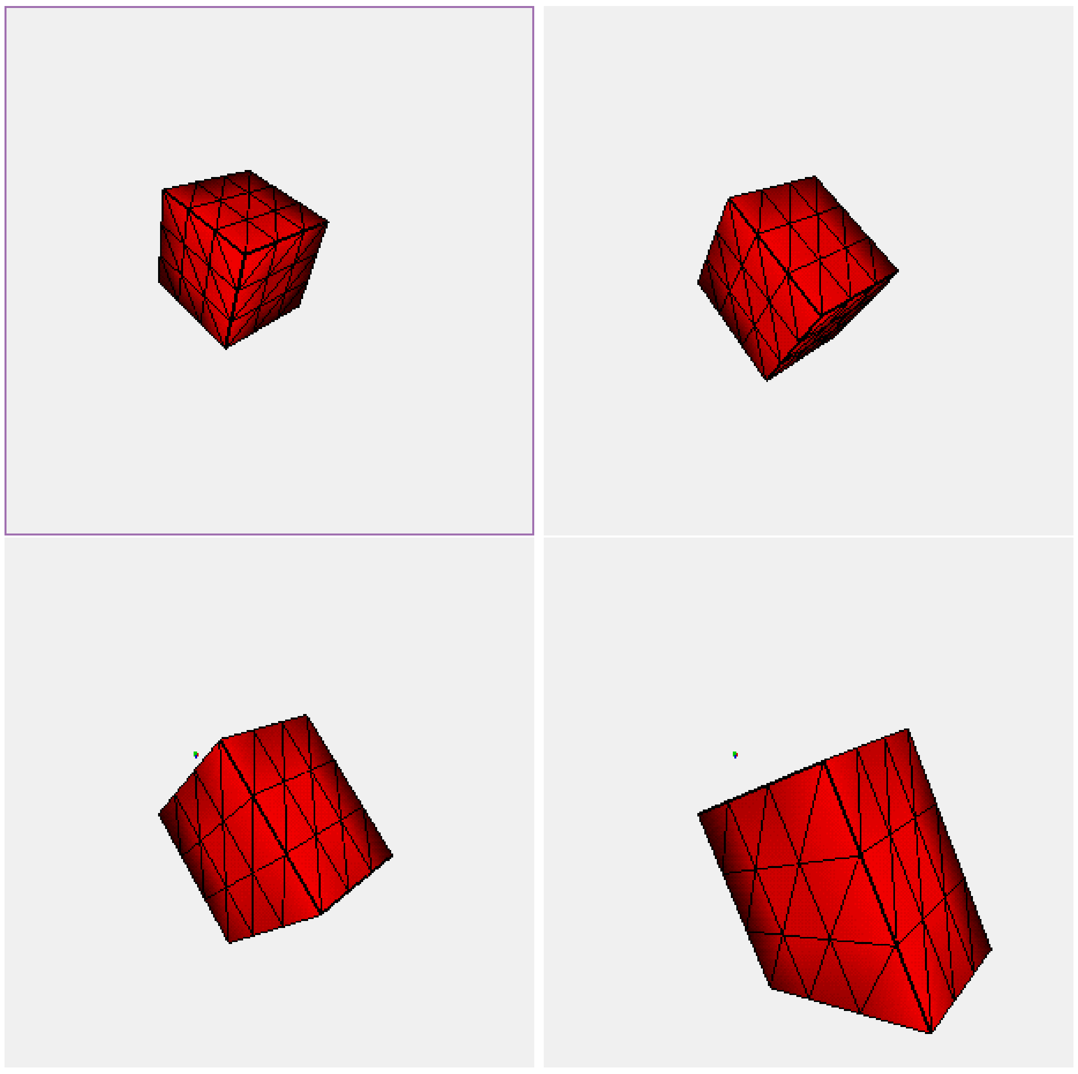


Figure 3.2: *Rotation of a cube using* **linear** *resolution. Only one edge is fixed. Note that the volume of the cube grows with increasing rotation.*

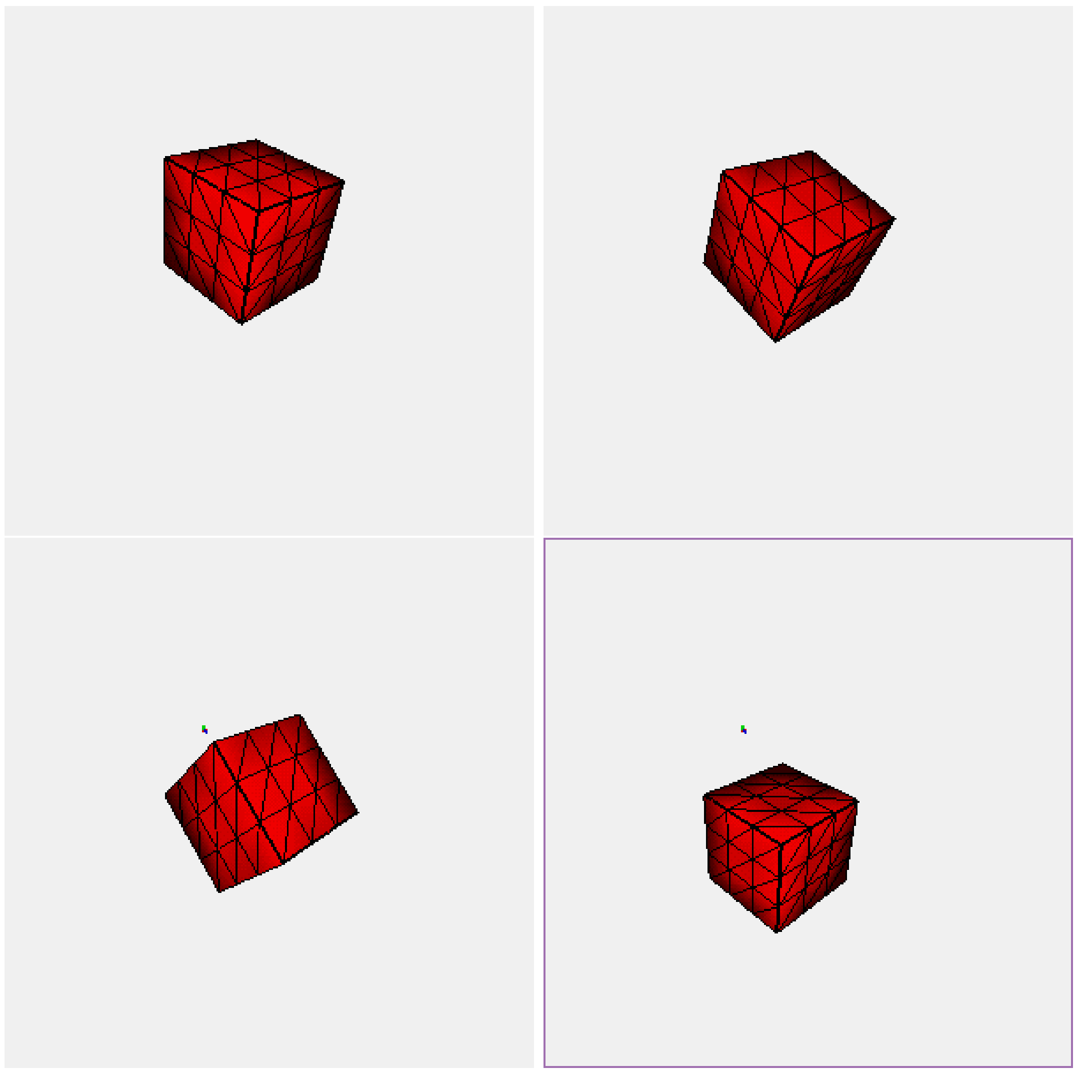


Figure 3.3: *Rotation of a cube using* **non-linear** *resolution. Only one edge is fixed. Note that the rotation does not change the shape of the cube.*

3.3 Non-linear static resolution

Linear analysis is only valid for small strains and displacements. Typically this excludes deformations of more then 5 percent or rotation. Beyond this we have to take into account non-linearities. We can differentiate between

- *Material nonlinearity:* Real materials are not Hookean. The constitutive relationship between strain and stress must be taken into account.
- *Geometric nonlinearity:* When the solid undergoes large variations in shape or rotation, the stiffness matrix will change.

An example of the latter case is given in Figure 3.2 where the effect of a rotation when using linear resolution is shown. Only when the change in the stiffness matrix is taken into account we obtain a rotation without deformation of the cube (Figure 3.3). Hence the stiffness matrix, K, is now a function of the displacements u:

$$K(u)\, u = f_{ext} \tag{3.4}$$

This gives rise to a non-linear problem since we can no longer just inverse the matrix K to find a solution for an external force being applied. Therefore we must resort to non-linear solution techniques, such as the Newton-Raphson scheme or its variant, which we describe below.

3.3.1 Newton-Raphson

The curve in Figure 3.4 shows the non-linear relationship between the displacement, u, and the force, given by the function $f(u)$. Thus when an external force, f_{ext}, is applied at displacement, u, the *residual force* is

$$r = f_{ext} - f(u) \tag{3.5}$$

Since at equilibrium the external and internal forces are balanced, it is our aim to minimize the residual force. We may evaluate the tangential stiffness matrix $K(u)$, and solve the linear system

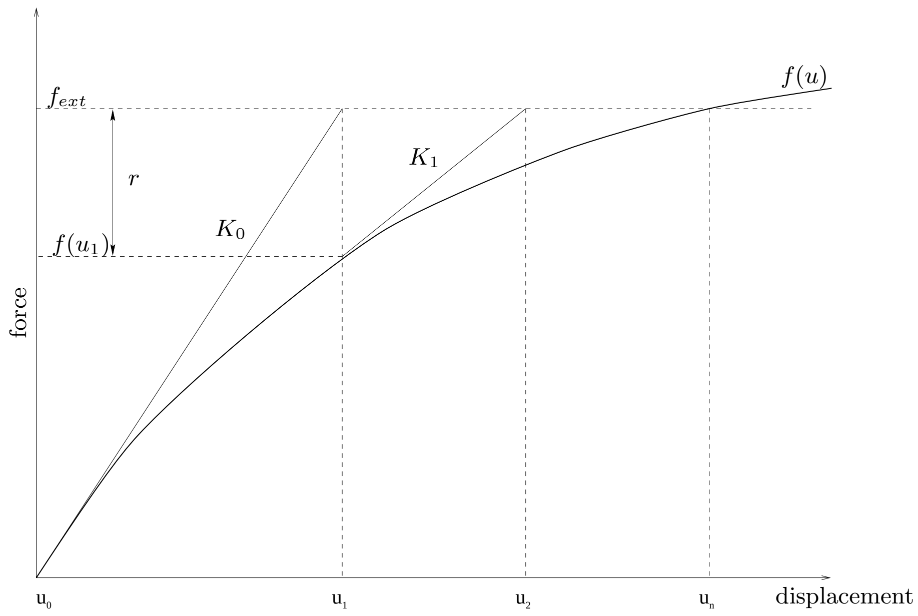


Figure 3.4: *Newton-Raphson iterative solution*

$$K(u)\Delta u = r \tag{3.6}$$

where Δu is the change in displacement. However, since the problem is non-linear the residual forces are not guaranteed to be zero. Thus we iterate the process until convergence.

$$K_i \Delta u_i = f_{ext} - f(u_i) \tag{3.7}$$

where K_i is equivalent to $K(u_i)$. The updated displacement is then obtained by

$$u_{i+1} = u_i + \Delta u_i \tag{3.8}$$

3.3.2 Modified Newton-Raphson

This method differs from the full NR (Section 3.3.1) only in that the tangent stiffness matrix is not updated. Thus we avoid reevaluating the matrix K at each iteration. However, more iterations are needed than for the full NR method to obtain the same result (see Figure 3.5). The stronger the non-linearity is, the larger the respective increase in iterations will be.

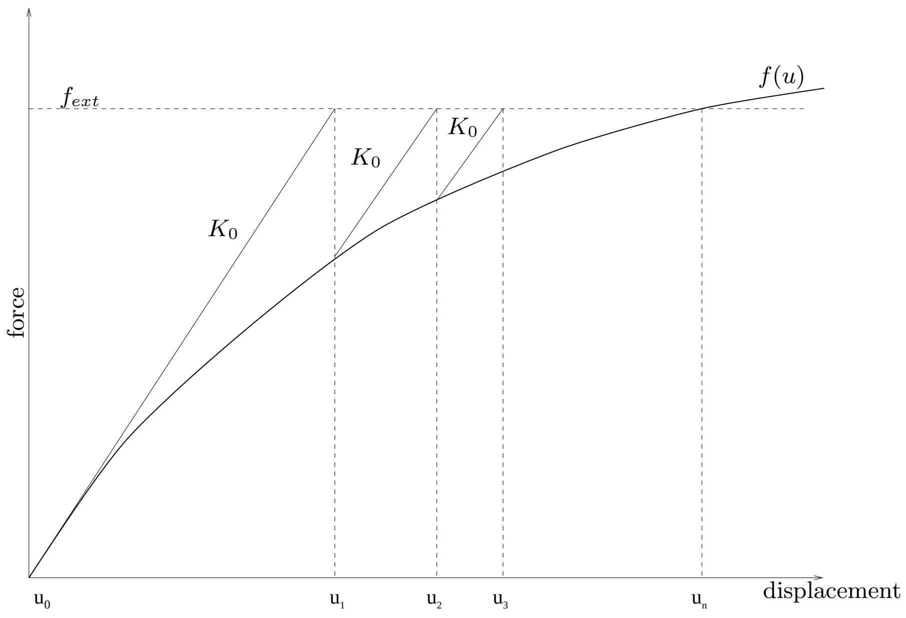


Figure 3.5: *Modified Newton-Raphson iterative solution*

An alternative to not updating the stiffness matrix at all, is to update it infrequently. For example, instead of updating at each iteration, the matrix could only be updated when it has changed considerably. Therefore we must reach a compromise between the number of iterations taken and the cost of updating the stiffness matrix.

3.3.3 Convergence

The iterative processes presented are not guaranteed to always converge. Especially hardening structures may be troubled by divergence problems. Figure 3.6 shows an example of a modified Newton-Raphson strategy, where starting from point A with stiffness matrix K_0, the solution will never converge at equilibrium B. In such a case *under-relaxation* would help. This means weighting the update in displacement, Δu, by a factor α, where $\alpha < 1$. Thus we can write the iterative update in displacement

$$u_{i+1} = u_i + \alpha(\Delta u_i) \tag{3.9}$$

In our implementation we find the value of α experimentally.

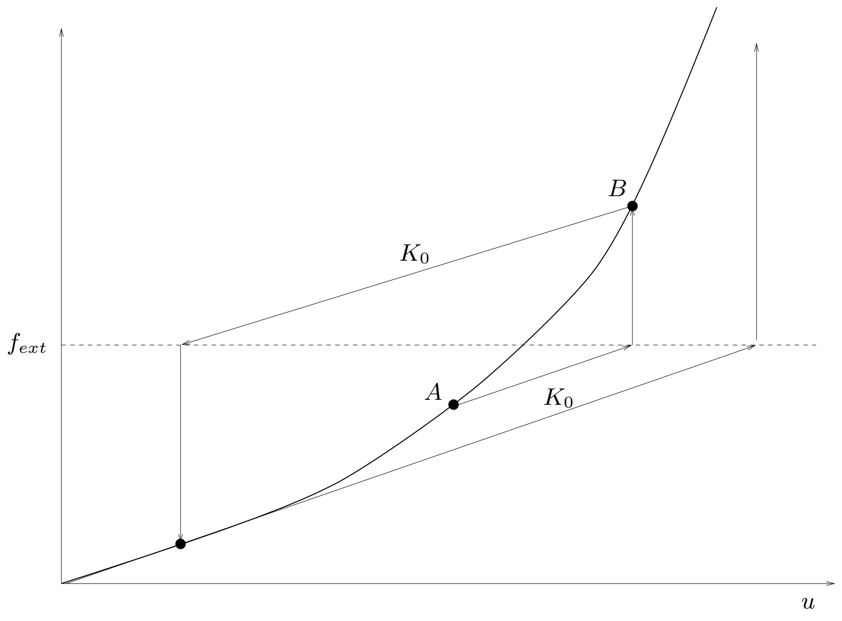


Figure 3.6: *Divergence of the modified Newton-Raphson iterative solution, starting from displacement A.*

3.4 Dynamic Resolution

When the frequency of excitation is lower than roughly one-third of the structure's natural frequency the solution is *quasi-static*, and the effects of inertia can be neglected. Otherwise we must take into account the mass matrix M and the damping matrix D.

$$M\ddot{u} + D\dot{u} + Ku = f_{ext} \tag{3.10}$$

When the mass is lumped (Section 2.3.3.4) at the nodes the mass matrix is diagonal and, therefore, easily invertible. We can re-express the problem as a second-order differential equation:

$$\ddot{u} = M^{-1}(-D\dot{u} - Ku + f_{ext}) \tag{3.11}$$

Since matrices D and K are not constant, the differential equation is *non-linear*. Therefore, there is no analytical solution to this problem. However, a variety of numerical integration techniques exist to compute position and velocity as a function of time.

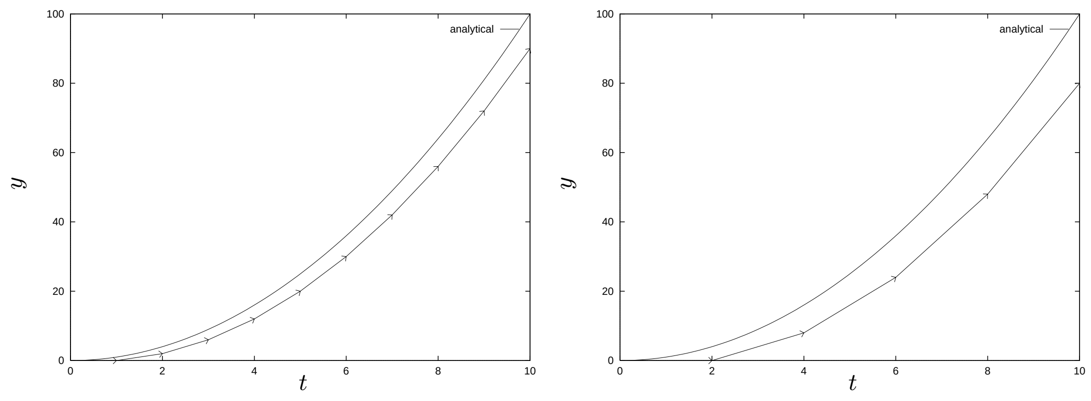


Figure 3.7: *Result of explicit Euler integration using timestep $h = 1$ (left), and $h = 2$ (right).*

3.4.1 Explicit Integration

The simplest method for finding the change in state with respect to time is the Newton-Euler method. It finds the next state by following the current derivative during one timestep h. The second order *differential equation* of Equation 3.10 can be re-expressed as two first order equations:

$$\begin{aligned} u_1 &= u_0 + h\dot{u}_0 \\ \dot{u}_1 &= \dot{u}_0 + h\ddot{u}_0 \end{aligned} \tag{3.12}$$

Then each equation can then be solved as

$$y_1 = y_0 + hf(y_0) \tag{3.13}$$

where y_1 is the next state, y_0 is the current state, and function f evaluates the derivative for a given state y. In Figure 3.7 we can see what happens if we integrate the simple differential equation $y' = 2t$. We know that the solution is $y = t^2$. Each arrow uses the derivative its beginning to find the new state. We notice that the larger the timestep, the larger the error is as well.

The explicit Euler integration method takes no notice of changing derivatives. It is actually a truncated Taylor series, discarding all but the first two terms. This means the Euler method is only correct if the first time derivative is constant. Since this is rarely the case, the *midpoint method* tries to approximate the second derivative. Using the first three terms of the Taylor series

$$y_1 = y_0 + hf(y_0) + \frac{h^2}{2} f(y_0)f'(y_0) + O(h^2) \tag{3.14}$$

where $O(h^2)$ represents the truncation error. We can rearrange it as

$$y_1 = y_0 + h(f(y_0) + \frac{h}{2} f(y_0)f'(y_0)) + O(h^2) \tag{3.15}$$

Since it would be expensive and complicated to evaluate $f'(y_0)$, we use the Taylor expansion

$$f(y_0 + \frac{h}{2} f(y_0)) = f(y_0) + \frac{h}{2} f(y_0)f'(y_0) \tag{3.16}$$

and substitute into Equation 3.15 to derive the midpoint method as

$$y_1 = y_0 + h(f(y_0 + \frac{h}{2} f(y_0))) \tag{3.17}$$

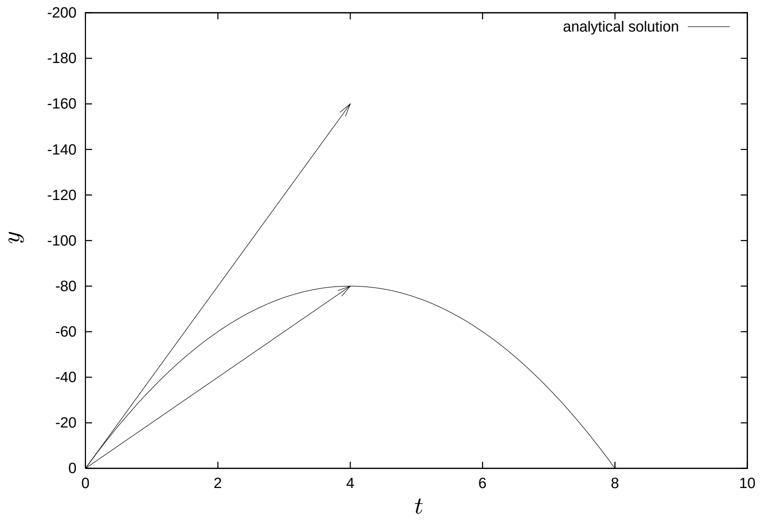


Figure 3.8: *Graphical representation of the midpoint method versus euler for an integration from $t = 0$ to $t = 4$. While the Euler method finds $y_{t=4} = -160$ the midpoint method finds $y_{t=4} = -80$ in accordance with the analytical solution.*

Thus the midpoint method takes a "trial" step half-way through the timestep, before taking the "real" step using the derivative at the half-way mark. If the second derivative

Order of consistency	1	2	3	4	5	6	7	8
Number of stages	1	2	3	4	6	7	9	11

Table 3.1: *Number of stages s necessary for different orders of consistency or accuracy. Note that after the fourth order the number of stages increases non-linearly.*

$$\begin{array}{c|c} & A \\ \hline & b^T \end{array} = \begin{array}{c|cccc} & & & & \\ & \frac{1}{2} & & & \\ & 0 & \frac{1}{2} & & \\ & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Table 3.2: *Classical values for the 4th order Runge-Kutta method arranged in a Butcher array.*

is constant, this works perfectly. For example in the case of a free fall under gravity (neglecting air drag) the midpoint method gives the correct solution. The downside is that instead of one evaluation of the derivative, it takes two. For a physical model this means that all forces (elastic and damping) have to be evaluated twice for each time step taken. The midpoint method is also called second-order Runge-Kutta. This method can be extended to higher orders. The general formulation is

$$\begin{aligned} y_1 &= y_0 + \sum_{j=1}^{s} b_j k_j \\ k_j &= f\left(y_0 + h\sum_{i=1}^{s} a_{ji} k_i\right) \end{aligned} \tag{3.18}$$

We can see that the number of stages s dictate how many times the derivative must be evaluated. By increasing the number of stages we gain in accuracy. The number of stages necessary for the eight first orders of consistency are shown in Table 3.1.

One of the most popular methods is the fourth-order Runge-Kutta (RK4) (see Table 3.2) since it has the highest order of consistency for an equal number of stages. For example, [Cotin, 1997] has used it for integrating the differential equation resulting from a dynamic analysis of anatomical structures. It requires four evaluations of the derivative per timestep. Thus it will be profitable to use this method if we can at least take a timestep twice as big as with the midpoint method for the same accuracy. The feasibility of this depends on the nature of the problem. Hence, let us thus revisit Equation 3.10;

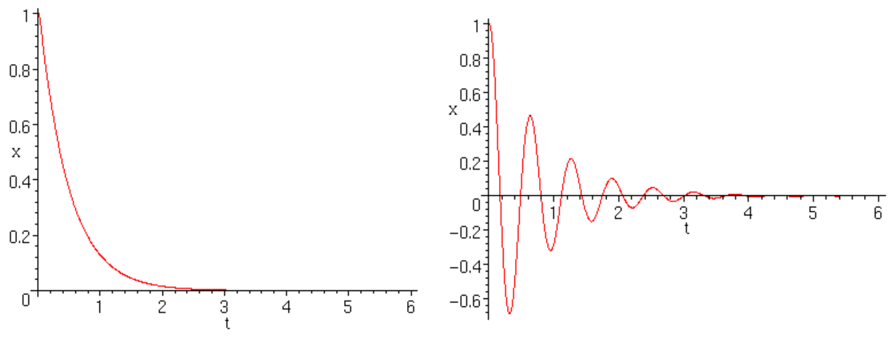


Figure 3.9: *Overdamped (left) and underdamped (right) cases.*

if we assume that the mass, damping and stiffness matrices are constant we can express the derivative of the position as

$$y' = f(y) = \lambda y \tag{3.19}$$

where y is the position and λ is

$$\lambda = \left[-\frac{D}{2M} \pm \sqrt{\left(\frac{D}{2M}\right)^2 - \frac{K}{M}} \right] \tag{3.20}$$

Therefore, the value of z is calculated as

$$z = \lambda h = \left[-\frac{D}{2M} \pm \sqrt{\left(\frac{D}{2M}\right)^2 - \frac{K}{M}} \right] h \tag{3.21}$$

Notice that when the sum in the square root is negative the result is a complex number. We can solve for the position analytically because the ordinary differential equation (ODE) is then linear, yielding

$$y_1 = y_0 e^z \tag{3.22}$$

Depending on the value of z distinct solutions are possible. If z is purely real (i.e. value under the square root positive), an *over-damped* response will follow. If, on the other hand, z has a real and an imaginary component, the solution will be *under-damped*.

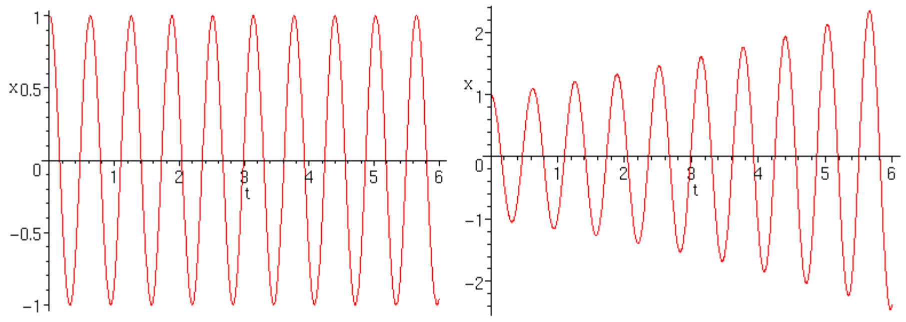


Figure 3.10: *Undamped (left) and divergence (right) cases.*

Finally, if z is purely imaginary (i.e. no damping), the response will be *undamped.* See Figure 3.9 and Figure 3.10.

The numerical solution of Runge-Kutta methods is then given by the Taylor series [Hairer et al., 1987]

$$y_1 = y_0 + y_0(\lambda h) + y_0\frac{(\lambda h)^2}{2!} + y_0\frac{(\lambda h)^3}{3!} + \dots \tag{3.23}$$

In Figure 3.11 we compare the numerical solution (for *one* timestep starting from y_0) of explicit methods of different order with an analytical solution for the over-damped case where z is purely real. We notice that the higher the order of the numerical solution, the better the approximation to the analytical solution is. In other words, the solution is more accurate. However, sooner or later the solutions diverge. It is important to note the timestep for which $|y_1| > |y_0|$. This means that the solution increases rather than decreases as it should. Thus the numerical solution *must not* surpass this timestep limit in order to avoid divergence (Figure 3.10). This statement is related to the Courant-Friedrichs-Lewy stability criterion, which stipulates that the distance traveled by a wave in the material must be less than the distance between two points of discretisation [Debunne, 2000].

The *stability domain* of Runge-Kutta methods of different order for a given z is shown in Figure 3.12. From this we can see that first and second order methods need damping to be stable. Also it can be shown that higher order does not necessarily imply better stability. For example in the over-damped case where z is purely real, the stability boundary is the same for first and second order methods. However, the most important conclusion to be drawn is that if the value of λ is large, the timestep h must be small in order to stay within the stability domain. Of course, this is computationally expensive

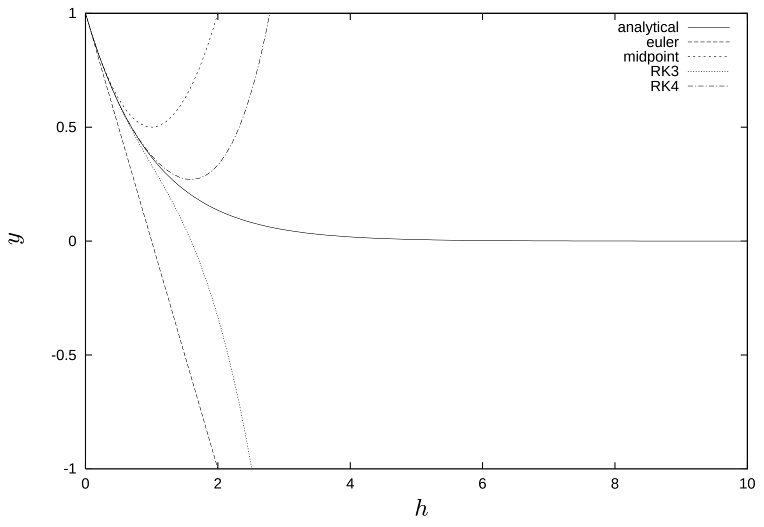


Figure 3.11: *Comparison of explicit methods of various order with the analytical solution of $y' = \lambda y$ (a.k.a. Dahlquist test equation).*

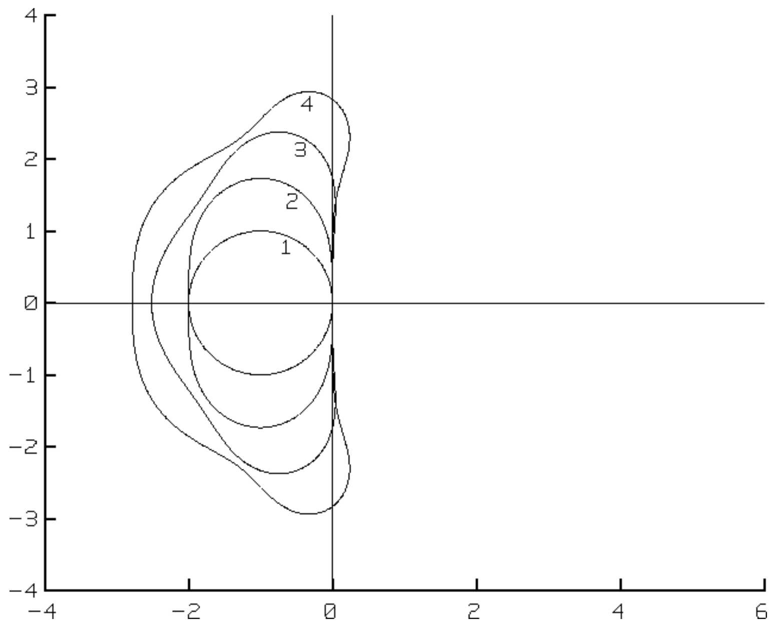


Figure 3.12: *Stability regions of Runge-Kutta methods of order from one to four for a given complex number z. Real component is on the horizontal axis, while imaginary component is on the vertical axis [Hairer and Wanner, 1991].*

since the derivative has to be evaluated at least once every timestep. For stiff objects this may mean that it is impossible to compute the dynamics in real-time[1].

In the linear case we can easily examine the accuracy of a numerical method since the exact or analytical solution is known. However, in the non-linear case this is not true but we do know that a method of higher consistency order will give a more accurate solution than a lower one for the same stepsize. Thus if we compute at least two sets of solutions with different consistency, this would give an estimation of the local truncation error. It can be used to adjust the stepsize so that an accurate solution can be obtained. If the error is too great, one can reduce the stepsize so that the solution is bounded within a specified tolerance. If the error is much too small as compared with the acceptable tolerance level, the stepsize can be increased to reduce the computational load [Hairer et al., 1987].

An alternative for adjusting the stepsize is given by [Joukhadar, 1996] that uses the assumption that the total energy of the system remains constant in order to adaptively control the size of the stepsize. If the energy of the system grows beyond a given tolerance the stepsize is reduced. Similarly, if there is very little change in energy the stepsize is increased. The tolerances that guarantee stability are normaly chosen experimentally. However, evaluation of the total energy of the system is computationally expensive and must be done at each timestep.

Even though adaptive step-sizing may reduce the computation there remains the possibility that the stepsize is reduced to the point that real-time execution may no longer be feasible. Thus there is no guarantee that real-time constraint are enforced while respecting a given error tolerance.

3.4.2 Implicit Integration

In the previous section we have seen that explicit methods are affected by stability problems. The change in state is based on the derivative at the same state. Implicit methods are different: the change in state is calculated by the derivative at the *next* state. Hence Equation 3.10 may be rewrite as the two following equations:

$$u_1 = u_0 + h\dot{u}_1$$

[1] "There are at least two ways to combat stiffness. One is to design a better computer, the other, to design a better algorithm."
H. Lomax in Aiken 1985

$$\dot{u}_1 = \dot{u}_0 + h\ddot{u}_1 \tag{3.24}$$

Each equation may be re-expressed and solved through Equation 3.25, but in practice if suffices to solve first for $\dot{u}_1$, as then u_1 becomes trivial to compute.

$$y_1 = y_0 + hf(y_1) \tag{3.25}$$

Thus at the new state the derivative will take you back to where you came from[2]. The system is therefore reversible in time and hence, completely stable. However, we do not know the derivative for a non-linear system at the next state in advance. But if we linearize Equation 3.10, we can solve analytically for the position after a given timestep h by

$$y_1 = \frac{1}{1 - h\lambda} y_0 \tag{3.26}$$

So if the real part of the eigenvalue (λ from Equation 3.21) is less than zero, i.e. $\Re(\lambda) \leqslant 0$, the absolute value of y_1 will always be less then the absolute value of y_0. Thus this method is absolutely stable, or *A-stable*. Hence for any given λ in the half-plane around the negative real axis, h can take *any* positive value [Hairer and Wanner, 1991].

Figure 3.13 shows the evolution of y for an increasing timestep on the x-axis. We notice that the numerical solution using the implicit method converges more slowly than the real, analytical solution. This is why it is said that implicit methods introduce "artificial damping". However, as h approaches infinity the value of y tends to zero. Thus the error decreases and it eventually converges towards the real solution. We call a method displaying this property *L-stable* or stiffly accurate. The explicit method, of course, is not L-stable.

Similarly to explicit methods, implicit methods come in different orders of consistency. For example the implicit midpoint method is given by

$$y_1 = y_0 + hf\left(\frac{y_1 + y_0}{2}\right) \tag{3.27}$$

while the implicit trapezoidal method is given by

[2] "If you know your history, Then you would know where you coming from."
Bob Marley, *Buffalo Soldier*

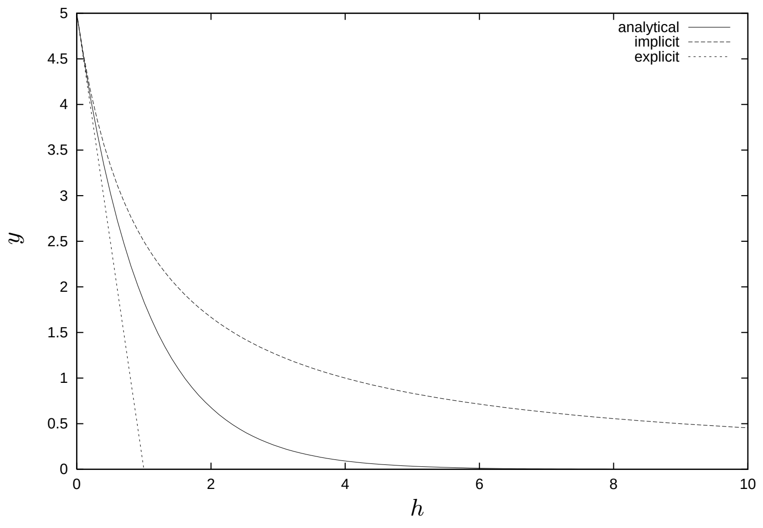


Figure 3.13: *Stability of implicit methods for the Dahlquist test equation ($y' = \lambda y$).*

$$y_1 = y_0 + h\frac{f(y_1) + f(y_0)}{2} \tag{3.28}$$

However, neither the implicit trapezoidal or midpoint method is L-stable, and does not converge towards the real solution as $h \to \infty$. Thus these methods tends to introduce "oscillation" into a numerical solution.

Until now we have only studied the stability and convergence of linear systems. Since the equations we want to solve are non-linear, we will now generalize A-stability to non-linear problems. If for a contractive system for all $h \geq 0$

$$|y_1 - \hat{y_1}| \leq |y_0 - \hat{y_0}| \tag{3.29}$$

where y_1 and $\hat{y_1}$ are numerical approximations after one step starting with y_0 and $\hat{y_0}$, then the method will be called *B-stable*. In other words, the methods is stable for a general nonlinear problem and the solution of the method does not magnify the initial error. Hence, if a method is B-stable, it will also be A-stable, but the opposite is not necessarily true. One should note that the convergence for nonlinear problems, or *B-convergence* may or may not correspond to the consistency of the method, i.e. the accuracy of the method we normally refer to. Table 3.3 summarizes the stability properties of some common methods. Note that the implicit trapezoidal methods is not B-stable while the implicit midpoint is.

Method	A-stable	consistency	L-stable	B-stable	B-convergence
Explicit Euler	No	$O(h)$	No	No	No
Implicit Euler	Yes	$O(h)$	Yes	Yes	$O(h)$
Implicit Midpoint	Yes	$O(h^2)$	No	Yes	$O(h)$
Implicit Trapezoidal	Yes	$O(h^2)$	No	No	No

Table 3.3: *Properties of common numerical methods.*

Hence an implicit method gives rise to a non-linear equation. For a deformable object there can be many hundreds or thousands of equations. Since they are coupled in most case we need to solve them in a system. [Hauth and Etzmuß, 2001] choose to solve the non-linear system using a modified NR method. Given that

$$G(y_1) = y_1 - y_0 - f(y_1)h \tag{3.30}$$

the function G is minimized by

$$\frac{\partial G}{\partial y} \Delta y = -G(y) \tag{3.31}$$

This means solving a linear system several times each timestep with the due computational overhead. To avoid iteration to solve the non-linear system, [Baraff and Witkin, 1998] use *semi-implicit* integration and linearize the equations by approximating the next state, y_1, as

$$y_1 = y_0 + h \left[f(y_0) + \frac{\partial f}{\partial y}|_{y=y_0}(y_1 - y_0) \right] \tag{3.32}$$

This does not guarantee the right solution for y_1, since the Jacobian matrix changes from one state to the next, and roughly corresponds to taking one NR step for the solution of the non-linear system. Thus it is not guaranteed to be stable for the non-linear case (A-stable but not B-stable) although it normally is, as long as the Jacobian does not change very much over the timestep. However, the resulting linear system needs only to be solved *once* per timestep. Hence, rearranging the equation we obtain

$$\Delta y \left[\frac{1}{h} I - \beta \frac{\partial f}{\partial y}|_{y=y_0} \right] = f(y_0) \tag{3.33}$$

where y is the state vector expressing the positions and velocities of all particles.

Thus Δy represents the change in state and $f(y_0)$ the derivative at the current state. Further, I is the identity matrix and $\frac{\partial f}{\partial y}$ the Jacobian matrix at the *current configuration.* The constant β ranges from $\beta \in [0, 1]$, and defines the amount of "implicitness" (note that if $\beta = 0$ the method reverts to an explicit integration). The resulting square matrix is composed of n^2 elements, where n is the size of the state vector. However, since particles in most objects are nearly never fully interconnected, this matrix is in the overwhelming majority of cases sparse. Thus for large objects it is essential to store these matrices in a sparse format if we want to keep memory requirement within reasonable limits.

[Desbrun et al., 1999] precompute the Jacobian matrix and assume that it is constants during the simulation. Hence they can pre-invert the matrix and store it. The update of the state is then given by multiplying the matrix by the derivative vector. However, since the Jacobian changes as the system evolves, a correction step is applied to account for the error.

[Debunne et al., 2001] decouples the Jacobian such that each particle is independent of all others. The solution of the linear system is then reduced to solving n systems of size 3×3. Since the Jacobian is assumed to be constant, the inverses of these matrices are precomputed. This represents a dramatic decrease in both computational and storage requirements. However, since the systems are decoupled, convergence of this method will be slower than when using a global Jacobian.

3.4.3 Modal Dynamics

A normal mode is a solution for the motion of a system in which all state variables move with simple harmonic motion and all at the same frequency. They may, however, have different amplitude and phase. Any motion is then a *superposition* of these normal modes. The transformation from the modal coordinates $\bar{u}$ to the original displacements u is then given by

$$u = \phi\bar{u} \tag{3.34}$$

where ϕ are the eigenvectors [Basdogan, 2001]. Then we may rewrite Equation 3.10 as

$$\phi^T M\phi\ddot{\bar{u}} + \phi^T D\phi\dot{\bar{u}} + \phi^T M\phi\bar{u} = \phi^T f \tag{3.35}$$

Now $\phi^T M \phi$, $\phi^T D \phi$, and $\phi^T K \phi$ are *diagonal* matrices; hence, the equations are independent.

[Pentland and Williams, 1989] further reduce the order of the problem by only considering the low-frequency modes when solving the system. By this they not only reduce the size of the system but also numerical instabilities are avoided. Since the high-frequencies have large eigenvalues they are responsible for much of the divergence problems of explicit methods (see Section 3.4.1). Thus by truncating the associated modes larger timesteps may be taken. Of course, the eigenvalues must be computed, and this can be an expensive procedure. In the cited case the mass, damping, and stiffness matrices are taken as constant. Thus the set of modes is also constant and the eigenvalue problem need only be solved once. Impressive speedups can then be observed, especially when only using first and second order modes.

However, [Krysl et al., 2001] present findings that modal reduction techniques may also benefit non-linear problems. Even though the eigenvalues have to be reevaluated, faster solution times are obtained. For implicit integration the technique shows faster results because of the smaller size of the system that needs to be solved. For explicit integration the possibility of taking larger timesteps by filtering out the high-frequencies accounts for faster solutions.

3.5 Solution of large linear systems

We have seen that both static and dynamic resolution may involve the solving of linear systems of the form

$$Au = b \tag{3.36}$$

where A is a given real $n \times n$ matrix and b a given real column vector of size n. It is desired to determine the unknown column vector u. For the case $n = 3$ we can rewrite the system as

$$\begin{array}{ccccccc} a_{11}u_1 & + & a_{12}u_2 & + & a_{13}u_3 & = & b_1 \\ a_{21}u_1 & + & a_{22}u_2 & + & a_{23}u_3 & = & b_2 \\ a_{31}u_1 & + & a_{32}u_2 & + & a_{33}u_3 & = & b_3 \end{array} \tag{3.37}$$

For cases where n is large, *direct methods* such as the Gauss-Jordan method are very slow due to cubic complexity of the solution. Besides speed, this method is also limited by the amount of memory available on the computer, since the inverse of the matrix is explicitly computed and must be stored. Even when the matrix is *sparse*, i.e. has only a few non-zero elements as compared to the total number of elements of A, the inverse may not be. Thus, for moderate sizes of n, the quadratic storage requirements soon attain the limits of available memory on most computers. It sould be noted that not all direct methods calculate the inverse; indeed, Gaussian elimination solves the linear system without computing the inverse, and is faster than Gauss-Jordan by a factor of 3.

However, it is wasteful to use general methods of linear algebra on such problems, because most of the $O(n^3)$ arithmetic operations devoted to solving the set of equations or inverting the matrix involve zero operands. Direct methods for solving sparse equations, then, depend crucially on the precise pattern of sparsity of the matrix [Press et al., 1992].

The influential work by [Young, 1971] and the rapid increase in computational power available popularized *iterative methods*. These methods can often be applied to sparse systems without a precise pattern for a lesser computational cost then direct methods.

3.5.1 Jacobi Method

If the diagonal values of A are non-zero, we can solve each equation for the corresponding unknown

$$\begin{aligned} u_1 &= \tfrac{1}{a_{11}}(b_1 - a_{12}u_2 - a_{13}u_3) \\ u_2 &= \tfrac{1}{a_{22}}(b_2 - a_{21}u_1 - a_{23}u_3) \\ u_3 &= \tfrac{1}{a_{33}}(b_3 - a_{31}u_1 - a_{32}u_2) \end{aligned} \tag{3.38}$$

With the Jacobi method we chose arbitrary starting values $u^{(0)}$ and compute $u^{(1)}$ using Equation 3.38. This process is repeated *iteratively* until convergence.

$$\begin{aligned} u_1^{(n+1)} &= \tfrac{1}{a_{11}}(b_1 - a_{12}u_2^{(n)} - a_{13}u_3^{(n)}) \\ u_2^{(n+1)} &= \tfrac{1}{a_{22}}(b_2 - a_{21}u_1^{(n)} - a_{23}u_3^{(n)}) \\ u_3^{(n+1)} &= \tfrac{1}{a_{33}}(b_3 - a_{31}u_1^{(n)} - a_{32}u_2^{(n)}) \end{aligned} \tag{3.39}$$

We can also rewrite this in matrix formulation as

$$u^{(n+1)} = -D^{-1}Eu^{(n)} + D^{-1}b \tag{3.40}$$

where D has the same diagonal elements as A but whose off-diagonal elements are zero. On the other hand , the diagonal elements of matrix E are all zero but the off-diagonal elements are identical to those of A. Thus $A = D + E$.

If we let $B = -D^{-1}E$ and $z = D^{-1}b$ we can rewrite in a condensed form as

$$u^{(n+1)} = Bu^{(n)} + z \tag{3.41}$$

As $u^{(n)}$ really is the sum of the exact solution u and the error term $e^{(n)}$ we may write

$$\begin{aligned} u^{(n+1)} &= B(u + e^{(n)}) + z \\ &= Bu + Be^{(n)} + z \\ &= u + Be^{(n)} \end{aligned} \tag{3.42}$$

Hence each iteration does not change the "correct part" of $u^{(n)}$ and therefore

$$e^{(n+1)} = Be^{(n)} \tag{3.43}$$

Thus if the error term is expressed as a linear combination of the eigenvectors of B, the largest associated eigenvalue (i.e. spectral radius $\rho(B)$) will be the component which takes the longest to converge [Shewchuk, 1994]. The *spectral radius* is defined as

$$\rho(B) = \max|\lambda_i|, \qquad \lambda_i \text{ is an eigenvalue of } B \tag{3.44}$$

It now becomes evident that if we want to minimize the error term, the spectral radius of B must be smaller than one.

3.5.2 Gauss-Seidel Method

The Jacobi method converges only very slowly since it takes a whole iteration for the new values $u^{(n+1)}$ to be propagated. Thus for large systems the Jacobi method is largely of theoretical interest. The Gauss-Seidel method uses the values $u^{(n+1)}$ as soon as they are available.

$$\begin{array}{lcllll} u_1^{(n+1)} & = & \frac{1}{a_{11}}(b_1 & -a_{12}u_2^{(n)} & -a_{13}u_3^{(n)} &) \\ u_2^{(n+1)} & = & \frac{1}{a_{22}}(b_2 & -a_{21}u_1^{(n+1)} & -a_{23}u_3^{(n)} &) \\ u_3^{(n+1)} & = & \frac{1}{a_{33}}(b_3 & -a_{31}u_1^{(n+1)} & -a_{32}u_2^{(n+1)} &) \end{array} \tag{3.45}$$

It should be noted that the Gauss-Seidel method depends on the order in which the equations are examined. The Gauss-Seidel method is sometimes called the *method of successive displacements* to indicate the dependence of the iterations on the ordering. This is important because if the matrix is sparse the dependency of each equation is not absolute. This means not every equation is dependent on all others. A good choice of ordering can enhance the rate of convergence, while a bad ordering will degrade it [Barrett et al., 1994].

3.5.3 Successive Over-Relaxation (SOR)

The SOR method is a further improvement of the Gauss-Seidel method. The new values of $u^{(n+1)}$ are weighted by a real parameter ω.

$$\begin{array}{lcllll} u_1^{(n+1)} & = & \omega\frac{1}{a_{11}}(b_1 & -a_{12}u_2^{(n)} & -a_{13}u_3^{(n)} &) + (1-\omega)u_1^{(n)} \\ u_2^{(n+1)} & = & \omega\frac{1}{a_{22}}(b_2 & -a_{21}u_1^{(n+1)} & -a_{23}u_3^{(n)} &) + (1-\omega)u_2^{(n)} \\ u_3^{(n+1)} & = & \omega\frac{1}{a_{33}}(b_3 & -a_{31}u_1^{(n+1)} & -a_{32}u_2^{(n+1)} &) + (1-\omega)u_3^{(n)} \end{array} \tag{3.46}$$

If $\omega > 1$ we are over-correcting. If $\omega = 1$ the method reverts back to the Gauss-Seidel method, while if $\omega < 1$ we are under-correcting. If the matrix has *property A* [Young, 1971], which often is the case for matrices arising from the discretisation of PDE (see Section 2.2), there is a direct relationship between the spectral radius ρ of the Jacobi iteration matrix (see Section 3.5.1) and the theoretically optimal value of w

$$\omega_{opt} = \frac{2}{1+\sqrt{1-\rho(B)^2}} \tag{3.47}$$

However, since calculating the spectral radius requires an impractical amount of computation, ω_{opt} is rarely evaluated explicitly.

3.5.4 Conjugate Gradients

Previously we have only investigated *stationary methods* where the computations are the same at each iteration. The conjugate gradient is one of the most popular *non-stationary* methods. This means that the computations involve information that changes at each iteration.

The choice of β (see Algorithm) ensures that $r^{(i)}$ and $r^{(i-1)}$ are orthogonal. In fact this implies that $r^{(i)}$ will be orthogonal to all previous vectors of r. It can be shown that this guarantees exact convergence for linear systems of equation in at most n iterations, where n is the size of the linear system [Saad and van der Vorst, 2000]. However, due to floating point errors, exact convergence is improbable in practice and the solution is obtained when the error drops below a defined tolerance ϵ.

Algorithm 1 Conjugate Gradients algorithm

```
Compute $r^{(0)} = b - Au^{(0)}$ for some initial guess $u^{(0)}$
$\delta_0 = r^T r$
$d = r$
**for** $i = 1$ to $i_{max}$ and $\delta_{i-1} > \epsilon$ **do**
    $q = Ad$
    $\alpha = \delta_{i-1}/(d^T q)$
    $u^{(i)} = u^{(i-1)} + \alpha d$
    $r^{(i)} = b - Au^{(i)}$
    $\delta_i = r^T r$
    $\beta = \delta_i/\delta_{i-1}$
    $d = r + \beta d$
**end for**
```

For real-time solution of very large systems even a solution in n iterations may be too expensive. However, many applications require the repeated solution of a large system where the solution u only changes minimally from one timestep to the next. In such cases using the previous result of u as the starting guess for $u^{(0)}$ can lead to dramatic gains in speed after finding the first solution. The number of iterations needed to minimize the error below a certain tolerance is very much smaller than the value of n.

Convergence of the conjugate gradients method depends on the spectral condition number of the matrix. For a symmetric positive-definite matrix it is given as the ratio of the largest to the smallest eigenvalue

$$\kappa = \lambda_{max}/\lambda_{min} \tag{3.48}$$

Then the reduction in error is expressed by

$$|e^{(n)}| \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^n |e^{(0)}| \tag{3.49}$$

From this relation we see that the number of iterations to reach a relative reduction of the error is proportional to $\sqrt{\kappa}$. This compares favorably with respect to the steepest descent method where error reduction is proportional to κ. We also find that CG converges more quickly if the eigenvalues are not spread too far away since this will make it easier to find a polynomial to minimize the error.

3.6 Artificial Neural Networks

[Cotin et al., 1999] have presented a method for pre-calculation of the displacements given the forces on the nodes using the finite element method (FEM); further assuming linear superposition of the displacements the deformation is obtained. This approach is very fast but only works for linearly elastic materials that are not subjected to rigid-body rotations. In a way we can regard this as a neural network with linear inputs and outputs, being equivalent to a matrix multiplication. In a linear case the input-output relationship is described by

$$b = Ma \tag{3.50}$$

where M is the *memory matrix*. On the other hand, for a nonlinear problem the memory matrix will depend on the value of the input

$$b = M(a)\, a \tag{3.51}$$

Artificial neural networks can be used to approximate this nonlinear relationship. Grzeszczuk et al. [Grzeszczuk et al., 1998] was the first to exploit multi-layer neural networks to approximate the non-linear change of state for a given spring-damper model. However, since in this case the dynamics are approximated rather than a static equilibrium state, the network may accumulate the error and wander off into regions for which no training data is available [d'Aulignac et al., 1997]. This introduces the need for a 'correction' step, by which the spring damper model is used to pull the input state off the object back into a known region.

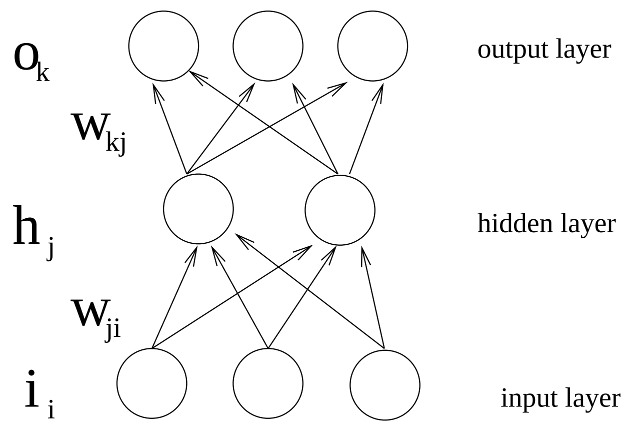


Figure 3.14: *Architecture of a Multi-Layer Perceptron with a single hidden layer. The layers are fully connected with each other.*

3.6.1 Multi-layer Perceptron

In Chapter 5 we will make use of the well known and widely used *Multi-Layer Perceptron* (MLP). The neurons on the input layer are fully connected with the neurons of the hidden layer. The importance of these connections are determined by the weights. Thus the inputs are multiplied by a weight and summed at the hidden neuron. A *sigmoid* function is then applied to this sum. The resulting value is then forwarded to all output neurons, where in turn all values from the hidden layers multiplied by a weight yield the final output (see Figure 3.14).

3.6.1.1 Backpropagation Learning Algorithm

We can easily deduce that the behavior of the MLP is determined by the value of the weights. However, by which procedure can we find the right weights for a given input-output mapping? The error back-propagation algorithm has been used with success for this purpose. For a given input where we know the desired output we can calculate the error. Using the chain-rule we may also find the change of the error *with respect to* (w.r.t.) the weights. Thus we may change the weights in order to minimize the error. Using an iterative approach and given a representative input-output mapping, it may be possible to approximate the function [Haykin, 1992].

The derivative of the error w.r.t the hidden-output weight is given by

$$\frac{\partial e_k}{\partial w_{kj}} = \frac{\partial e_k}{\partial o_k}\frac{\partial o_k}{\partial w_{kj}} \tag{3.52}$$

To minimize the error we follow this derivative by a given alpha.

$$\Delta w_{kj} = \alpha h_j (t_k - o_k) \tag{3.53}$$

Similarly for the input-hidden weights

$$\frac{\partial e_k}{\partial w_{kj}} = \frac{\partial e_k}{\partial o_k}\frac{\partial o_k}{\partial h_j}\frac{\partial h_j}{\partial n_j}\frac{\partial n_j}{\partial w_{ji}} \tag{3.54}$$

$$\Delta w_{ji} = \alpha h_j (1 - h_j) i_i \sum_{k=1}^{n} w_{kj}(t_k - o_k) \tag{3.55}$$

where the n_j represents the sigmoid activation function.

The obvious danger of this approach is that it may be trapped in local minima and never achieve global convergence. The other problem is to determine the number of hidden neurons necessary to approximate the function.

3.7 Conclusions

In this chapter we have examined some of the possible resolution techniques to find the deformation of an object.

If the evolution of the deformation with respect to time is not critical to our analysis we may resort to a static resolution. This is typically the case when the transient response to an externally applied force is of negligible interest. For a static analysis we may differentiate between the linear and non-linear case. A linear resolution is only valid for cases where the stiffness matrix does not change much for the resulting deformation. Typically large deformations introduce geometrical and material non-linearities. In such cases a non-linear resolution must be performed for accurate results. Classical methods such as the Newton-Raphson method rely on an iterative resolution of a linear system to resolve the non-linear system. This, however, represents a considerable increase in computational complexity. Nevertheless the requirements of our application may demand this (as discussed later in Section 6.4).

Alternatively a dynamic analysis of the deformation process can be made. Since the differential equations, describing the change in deformation with respect to time, are non-linear, one must resort to numerical methods. The explicit methods are simple to implement and are of low computational complexity per timestep, however, they suffer from stability restrictions that depend on the stiffness and damping coefficients of our deformation model. In practice this means that only very small timesteps will ensure stability for stiff objects, which may compromise real-time resolution. Implicit methods remove the limitations on stiffness, damping, or timestep values, but at the cost of solving a non-linear system. Nevertheless, the complexity of solving this system may, in some cases, be preferable to an explicit integration as we shall confirm experimentally later (Chapter 5).

Both static and dynamic resolution may require the solution of one or more linear systems of equations. However, direct inversion of the matrix is not a viable option for large systems as the computational and storage requirements are prohibitively large. As our systems are typically sparse (i.e. many zero entries in the matrix), iterative methods present an interesting alternative. We have examined stationary methods such as Jacobi and SOR, which have a low cost per iteration but may take longer to converge than non-stationary methods such as the steepest decent and conjugate gradients. However, the latter require more computation per iteration. In Chapter 6 we will examine their relative methods in the context of a non-linear resolution.

The solution of a non-linear system with the Newton-Raphson method requires several solutions of a linear system. On the other hand multi-layer neural networks are known for their ability the approximate non-linear function. Hence we have described the behavior and training algorithms of a multi-layer perceptron. Experimental results are delayed till Chapter 5.

Chapter 4

Interaction

Contents

Is this a dagger which I see before me,
The handle toward my hand?
Come, let me clutch thee.
I have thee not, and yet I see thee still.
Art thou not, fatal vision, sensible
To feeling as to sight?

William SHAKESPEARE,
Macbeth, Act 2 scene 1

4.1 Introduction

In the previous chapters we have seen how deformable bodies can be modeled and examined the resolution techniques available. However, the deformation is normally created by the interaction with another object. In this chapter we will discuss how to take this phenomenon into account during simulation and its implications.

The first issue is to detect that a collision has actually taken place. Since most physical models are based on a polygonal representation of the surface, we present the different techniques available to detect the collisions between convex and concave polyhedra in Section 4.2. As we shall see the different approaches return different information. Some will give the distance between objects (positive or negative), and other the polygons in contact. Once a collision is detected, a suitable reaction should be computed[1]. Here too several approaches exist. We discuss their relative properties in Section 4.3. In Section 4.4 we explain the problems that are commonly experienced when using a force-feedback device to interact with a simulation, and propose a solution by the means of a local approximation of the contact.

4.2 Collision Detection

The collision detection problem consists of determining if a set of geometric objects are colliding (or interpenetrating). When the physical response to the collision has to be evaluated (as it is the case for an interactive simulation), this basic problem (i.e. binary response) is complemented by the need to determine some of the geometric characteristics of the colliding situation, e.g. involved geometric entities, contact direction, etc. It is well-known that this computational costly problem represents a bottleneck for a large number of geometrical based algorithms, and in particular for dynamic simulation. Moreover, the complexity of the collision detection problem increases when deformable objects are involved (as it is the case when simulating biological tissues, see for instance [Cover et al., 1993]).

4.2.1 Convex Polyhedra

Several well-known and commonly used approaches for solving the collision checking problem in convex polyhedral environments can be found in the literature. Lin & Canny

[1]as Macbeth found out...

[Lin and Canny, 1991] have proposed an incremental algorithm which computes in constant time the positive distance[2] between two rigid and convex polyhedra in motion; this algorithm has been extended by Kotoku [Kotoku et al., 1992] in order to localize the contact when the distance between the two convex polyhedra is negative[3]. Gilbert and al [Gilbert et al., 1988] have proposed an algorithm (the GJK algorithm) which computes in linear time (w.r.t. the number of vertices) the positive distance between the convex-hulls of two sets of points, and which also gives an approximation of the negative distance for small inter-penetrations; later, this algorithm has been enhanced by Cameron [Cameron, 1997] in order to achieve a constant computation time when dealing with moving convex polyhedra The latter algorithm may also reuse the data (or *tracking information*) from the last distance calculation. Hence, if the configurations of the polyhedra have not changed much, the new minimal distance can be calculated much faster.

In [Sundaraj et al., 2000] a new approach (hereafter refered to as the SDM algorithm) to finding the distance between convex polyhedra was proposed. This method has different terminating conditions than the GJK approach, which makes is more robust. We compared the execution times of this algorithm with Cameron's enhanced GJK hill-climbing algorithm[4]. The objects tested are randomly generated convex polyhedra. For a pair of objects, one is placed at the origin while we apply a continuous translation to the other object through 10e4 little incremental displacements.

Figure 4.1 shows how the timings for Cameron's enhanced GJK version increase with object complexity (i.e. the number of vertices of the polyhedra). When no tracking information is used we obtain a linear relationship between complexity and execution time; for every configuration along the path of the translation the distance between the two polyhedra is calculated anew. However, when translating one polyhedron with respect to the other, the *witness points* that determine the two closest points may change little. Hence, if the last witness points are used as the starting point for the new distance calculation, this can lead to a much faster computation. When making use of this information the time is almost constant ($0.36s$ on average) for an increasing complexity.

When repeating this experiment with the SDM approach we obtain the plot given in Figure 4.2. We observe the same linear and near-constant relationship without and with tracking information respectively. However, with this approach the time increase is

[2]The distance between two polyhedra is called *positive* when the two polyhedra do not intersect.

[3]When two polyhedra intersect, their *negative distance* represents the length of the smallest translation which is needed to separate the two polyhedra.

[4]The code used was downloaded from http://www.comlab.ox.ac.uk/ cameron/distances.html

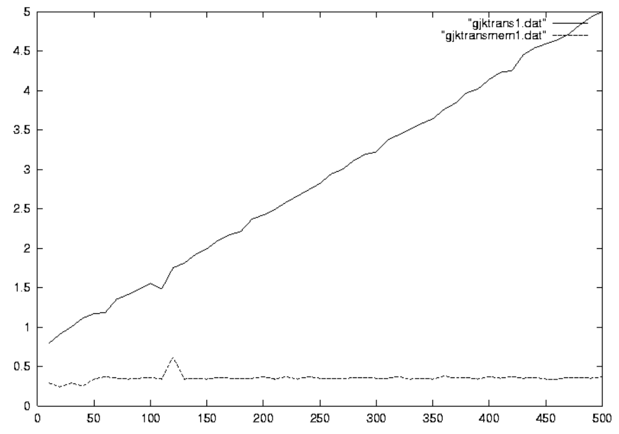


Figure 4.1: *Enhanced GJK: Execution time (y-axis) for a continuous translation with (dashed) and without (solid) using tracking information in function of the number of points of the polyhedra (x-axis).*

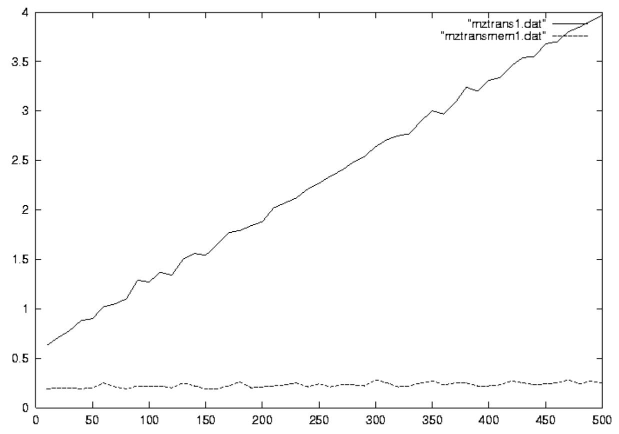


Figure 4.2: *SDM approach: Execution time (y-axis) for a continuous translation with (dashed) and without (solid) tracking information in function of the number of points of the polyhedra (x-axis).*

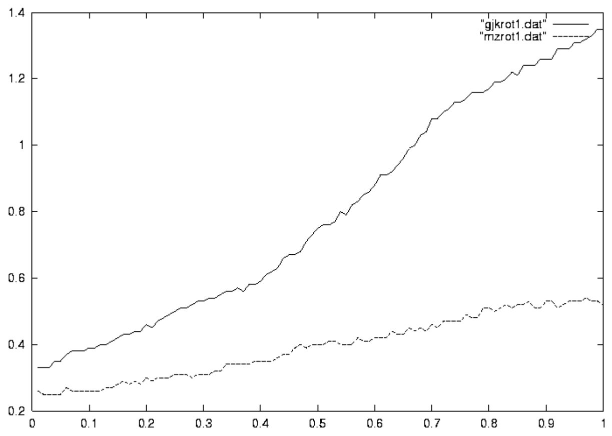


Figure 4.3: *Evolution of the time (y-axis) for a continuous rotation with different rotational velocities (x-axis) with enhanced GJK (solid) and SDM (dashed), both using tracking information.*

slower with rising complexity of the objects, and the near-constant time drops to 0.24s on average.

For the experimental results shown in Figure 4.3 we placed two objects with 50 points each a given distance apart and then applied a continuous rotation over 10e4 steps to one of them. The plot shows how the timings evolve for different rotational velocities. The higher the rotational velocity, the larger the change in the closest points will be. Hence, when using the tracking information, the time to compute the new minimal distance will also increase as the rotational velocity grows. We observe that the SDM approach gives lower timings even at high rotational speeds as compared to the enhanced GJK method.

The above results were obtained using a *SGI Octane (175MHz)*, and the code was compiled using the *CC* compiler with *-Ofast* optimization. However, some caution should be taken when interpreting these results, since they represent a purely practical analysis dependent on the implementation details.

4.2.2 Concave polyhedra

In Section 4.2.1 we presented several methods for the detection of a collision between convex polyhedra. However, those approaches are not directly applicable to concave objects. In this section we will examine possible solutions to this problem.

4.2.2.1 Pruning

Given two concave polyhedra the simplest algorithm is a systematic collision check between each and every pair of their facets. This, however, leads to a quadratic complexity that makes its application prohibitively expensive for all non-trivial problems. [Joukhadar et al., 1996] proposed to initially check for a collision of the convex hulls of the concave polyhedra. In the event of a collision between the convex envelopes only facets facing each other will be considered for collision detection; this pruning leads to a lower algorithmic complexity, but puts some restrictions on the types of contact possible. The direction of the contact and the volume of inter-penetration may also be calculated.

4.2.2.2 Convex Decomposition

The approaches in Section 4.2.1 can also be used for dealing with concave polyhedra, by using convex decompositions. However, such an approach cannot be used when dealing with deformable objects, since the related convex decompositions have to be recomputed after each local deformation of the object (i.e. on line); this should require a quadratic time algorithm (w.r.t. the number of vertices). In order to overcome this problem, Baraff & Witkin [Baraff and Witkin, 1992] have divided objects beforehand into convex sub-objects that can only be submitted to first order polynomial deformation (i.e. a face or an edge cannot be “bent”); this approach allows them to guarantee that the used decomposition is stable during the simulation (i.e. that any sub-object remains convex), and that a traditional fast collision checking algorithm can be applied among the set of related sub-objects. Unfortunately, the previous deformation constraint does not hold in some situations, and highly deformable objects have to be decomposed into a very large number of sub-objects.

4.2.2.3 Object hierarchies

If we can build a hierarchy of the sub-objects that make up an object (i.e. polygons at the lowest level), collision checking can be greatly accelerated. Rather than checking for collision at low levels of the hierarchy (polygon/polygon tests), much computation can be saved by initially checking at the higher levels and descending lower if and only if a collision has been detected.

Volino & Thalmann [Volino and Thalmann, 1994] have proposed a hierarchical algorithm which exploits the curvature properties of a surface, in order to solve the self-

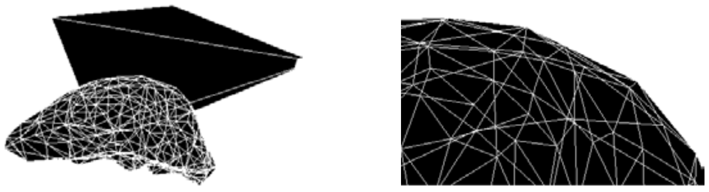


Figure 4.4: *Making use of the OpenGL bounding box to detect the polygons in interaction. On the left the bounding box intersecting the liver; on the right the polygons within the bounding box.*

collision problem. In [Gottschalk et al., 1996] oriented bounding boxes (OBB) are used to build a hierarchy of a complex object consisting of many thousands of polygons. The software for this latter approach is available as part of the library RAPID. Alternatively, axis-aligned bounding boxes (AABB) are used in [Bergen, 1998] to build a hierarchy. It is computationally simpler to use AABB, but the hierarchies tend to be deeper than with OBB. After the hierarchies have been built collision detection between rigid objects (concave or convex) is very fast. However, a problem with deformable objects is that the hierarchies can change, and recalculating them is computationally expensive.

4.2.2.4 Hardware acceleration

Alternatively, an approach for detecting the collision between deformable and rigid objects of rectangular shape has been proposed by [Lombardo et al., 1999]. It makes use of the OpenGL hardware to detect the polygons within a bounding box (see Figure 4.4). If this bounding is superimposed onto the rigid, rectangular object, the polygons in interaction can be detected when using the picking mode. This approach has the advantage of being very fast due to hardware acceleration. However, it imposes very strict conditions on the type of interactions possible as the shape of one of the objects is limited by the type of bounding box OpenGL is able to construct. Also, this approach may supply information on the distance of inter-penetration by using the z-buffer.

[Hoff et al., 2001] present a new approach for computing generalized proximity information of arbitrary 2D objects using graphics hardware. This not only includes detecting collisions, but also computing intersections, separation distance, penetration depth, and contact points and normals. Promising indications have been given that this method can be extended to three dimensions[5].

[5]presentation at Stanford Workshop on Surgery Simulation

4.3 Collision Response

From a macroscopic point of view, a collision generates interdependent physical phenomena, whose characteristics are tightly related to the mechanical properties of the involved objects: local deformations of the colliding objects, interaction forces, and changes in the velocities of the involved objects. For dynamic simulation, the problem to solve is to quantify the collision response in terms of the implied forces and of the velocities/accelerations changes.

Several types of approaches have already been considered for solving the related algorithmic problem. A first type of contribution comes from the field of Mechanics [Routh, 1905, Goldsmith, 1960, Pars, 1965], and from the study of the dynamic control of kinematic chains [Craig, 1989, Featherstone, 1987].

4.3.1 Constraints

A more algorithmic (but approximated) approach comes from the field of Computer Graphics. In this case, complex geometric models introduce so many different control parameters that physically correct behavior is hard to obtain, even for an experienced operator. Thus, in order to reduce the number of degrees of freedom, simplified dynamic models are used. The most frequent approach consists in considering collisions and contacts as geometrical constraints, and thereby obtains a uniform treatment of both interactions and articulations [Baraff, 1992].

An other type of approach, which tries to combine the advantages of the Mechanics and of the Computer Graphics models, is currently developed in the field of Robotics (mainly for teleoperation and medical applications). In this approach, the various models and algorithms developed for kinematic chains may be applied, but the representation of contacts by geometrical constraints leads to physically incorrect approximations [Mirtich, 1996]. Furthermore, the constraints-based model induces a global representation of all constraints by a system of non-linear equations which is difficult to maintain and solve (NP-hard).

4.3.2 Impulse

An approach for solving the collision response problem, consists in performing a macroscopic analysis of the phenomena, and in applying appropriate representations of some basic mechanical principles. When two objects collide with each other, they locally deform. Consequently, they store potential energy, which will be transformed into kinematic energy when these objects return to their initial forms. So, the collision force is directly related to the amount of deformation, and the force/deformation dependence is theoretically characterized by the physical nature of the involved bodies (e.g. the Hook law holds when dealing with perfectly elastic collisions). From the algorithmic point of view, one can use the impulse method for estimating the collision force when dealing with rigid bodies.

It is commonly agreed that the *Impulse model* is a good macroscopic approximation of a collision phenomenon involving purely rigid bodies: in such a case, the collision itself can be considered as instantaneous, and contact forces as impulsive (see [Mirtich, 1996] for a practical implementation of this approach). Unfortunately, this approach cannot be applied when dealing with deformable bodies, since the collision phenomenon cannot be considered as instantaneous.

4.3.3 Penalty based models

The goal of the *Penalty based models* is to give a continuous representation of the collision at a macroscopic level, consisting in defining the response as a function of the local inter-penetration. As the goal of any collision response is to eliminate inter-penetration a repulsive force F is calculated as

$$F = -\lambda x \tag{4.1}$$

where x is a geometrical measure of the inter-penetration and λ the *penalty* for that penetration. This representation by *Hooke's law* is suitable for perfect elastic collision. A viscous term, μ, can be introduced in order to model damping phenomenon:

$$F = -\lambda x - \mu \dot{x} \tag{4.2}$$

where $\dot{x}$ is the change of penetration with respect to time. This model is quite similar to that of *Kelvin-Voigt* used for vibrational mechanics.

4.3.3.1 Measure of inter-penetration

The measure of inter-penetration commonly used is the negative distance. Some algorithms presented in Section 4.2.1 compute this distance as part of the collision detection. Others, such as the hierarchical approaches (in Section 4.2.2), only return information about which polygons of the object have collided. In such cases the negative distance must be calculated explicitly for a penalty based response.

Distance, however, is not the only measure of inter-penetration. In [Sundaraj and Laugier, 2000] the volume of inter-penetration is calculated from the intersecting elements detected by the collision check. Nevertheless, the cost of calculating the volume is superior to the cost of calculating the negative distance.

4.3.3.2 Non-linear penalty model

The traditional linear penalty model frequently used in Computer Graphics exhibits well known non physical side effects: the generation of fictitious sticking forces produced by the viscosity term, and the non physical nature of the λ and μ parameters. In order to overcome theses problems, we have chosen to make use of a *non-linear Penalty model.* In Figure 4.5, the collision force is plotted as a function of the penetration value: during the ballistic phase $F = 0$; then, the force increases until it reaches maximum compression and reduces. When using a traditional viscous-elastic model,the force F becomes negative at the end of the collision, because of the presence of the damping term (see. Figure 4.5.a); this should not occur and tends to keep the two objects stuck together (fictitious sticking forces). The non linear model

$$F = -\lambda x - \mu \dot{x} x \tag{4.3}$$

proposed by K. H. Hunt and F. R. E. Crossley [Hunt and Crossley, 1975] allows us to avoid this problem, and to compute F while respecting the theoretical shape of Figure 4.5.b; this solution also suppresses the discontinuity of the repulsion force at the beginning of the collision. Hence the reaction forces given by the non-linear model are

$$\vec{F_c} = \begin{cases} (-\lambda x - \mu \dot{x} x)\vec{k} & \text{if } x < 0 \\ \vec{0} & \text{otherwise} \end{cases} \tag{4.4}$$

where λ is the rigidity factor of the collision, μ is a damping factor (which represents the dissipation of energy), x the inter-penetration, and $\vec{k}$ the contact direction

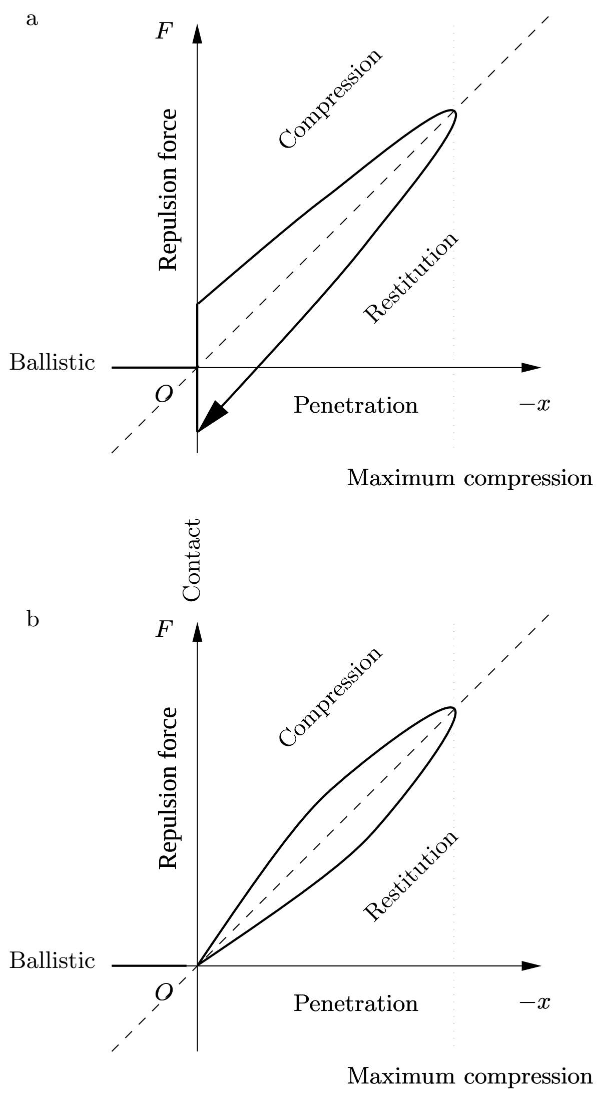


Figure 4.5: *(a) Regular viscous-elastic and (b) non linear viscous-elastic model.*

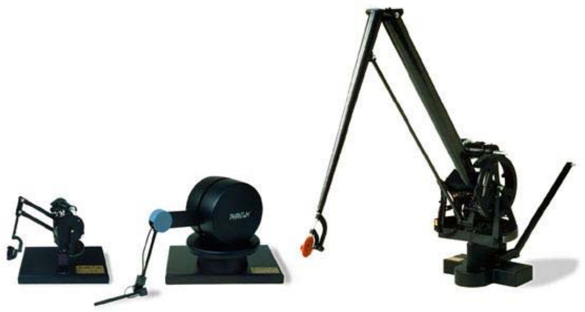


Figure 4.6: *Different models of the PHANToM force-feedback device*

[Deguet et al., 1998]. Given that the coefficient of restitution e is given as

$$e = 1 - \alpha v_i$$

where v_i is the impact velocity, the damping coefficient μ may be approximated as

$$\mu = \frac{3}{2}\alpha\lambda \tag{4.5}$$

[Marhefka and Orin, 1996] show how the constant $\frac{3}{2}$ is consistent with Hertzian theory for contacting spheres. To avoid visible inter-penetration, the rigidity constant has to be quite large, which leads to large forces being generated. However, if an equal and opposite force vector is passed to the force-feedback device at the global simulation rate (when $<< 1kHz$) this may lead to unacceptable trembling. To resolve this problem we propose to use a local approximation at a higher frequency, as explained in the next section.

4.4 Haptic Interface

The last decade has seen the introduction of several force-feedback devices into the market. They allow the user not only to see the virtual environment, but also feel it. Several products are now commercially available, ranging from force-feedback gloves to devices specifically developed for virtual surgery such as the *Laparoscopic Impulse Engine*. However, one of the currently most popular haptic interface is the PHANToM[6]. Some of the models available are shown in Figure 4.6.

[6]see http://www.sensable.com

They consist of a stylus attached at the end of an articulated arm. The stylus is held by the user to move in the virtual environment. Thus the position (and orientation as an optional feature) of the stylus may be calculated. According to the laws of the virtual environment forces may be applied to the end effector of the articulated arm. Typically these forces depend on the interaction with objects in the environment, and therefore, there is a need for collision detection and appropriate response. However, force-feedback devices require a high update rate for the force resulting from these collisions. For example the update frequency for the PHANToM is around $1kHz$. For interaction with large, complex objects it may be computationally impossible to respect such high rates.

Problem The disparity between the physical model update frequency and the haptic rendering frequency can compromise the haptic interface. The haptic device needs to display forces at rates around 1KHz to maintain a realistic sense of touch. Simple solutions as forces interpolation cause delays in the haptic rendering, and presenting force feedback in the presence of even small delays has been shown to create operator-induced instabilities [Richard et al., 1996].

[Astley and Hayward, 1998] proposed a multi-resolution approach where only areas of haptic interest are updated at a high frequency. [Cavusoglu and Tendick, 2000] proposed reducing the order of the deformable model to calculate an approximation of the elastic force at a higher frequency. [Picinbono and Lombardo, 1999] use extrapolation to smooth the discontinuities in the forces. They evaluate three methods: constant extrapolation, linear extrapolation over time, and linear extrapolation over position. Previously, [Mark et al., 1996] suggested decoupling the haptic servo loop from the main application for modeling the collision with rigid objects.

4.4.1 Haptic interface using a buffer model

[Balaniuk, 1999] proposed to bridge the frequency disparity between the physical model simulator and the haptic device by using a “buffer model”. The haptic interface is conceived as an external module of the physical simulator, able to estimate the contact forces to be displayed using its own simplified physical model. This simple model locally emulates the “real” physical model and can estimate contact forces at the haptic rate. The buffer model is a generic numerical model, defined by a set of parameters, continuously adapted in order to fit the values informed by the physical simulator. The contact forces estimation is based on a very simple physical simulation, with a point interacting with a non deformable constraint surface defined by the buffer model.

The position and the shape of this surface will be defined by the model parameters. The constraint surface is defined in the haptic device configuration space. In an ideal framework, the haptic device would offer force and torque feedback, and the proxy would move in a 6-D configuration space. In this case, the proxy motions can be expressed in full probe motions (6 dof : translation and rotation). The PHANToM haptic device we use has no torque feedback, and the proxy is a point in a 3-D configuration space. The proxy motions are expressed only by translations of the probe in the virtual environment (3 dof). The probe does not rotate.

Inside the haptic interface we use a constraint-based framework. The haptic probe location inside the configuration space is defined by two values: its effective position and a virtual location, subject to the constraining obstacles. This virtual location, called the “god-object” in [Zilles and Salisbury, 1994] and “virtual proxy” in [Ruspini et al., 1997], is the place the described object would be if obstacles were infinitely stiff.

As described in Section 4.2, the interaction force is calculated with respect to the inter-penetration distance of two colliding bodies. This force is then used by the physical model to compute its update in state. Thus the rate at which force values can be supplied is limited by the execution time of collision detection and, thereafter, the update of the physical model.

Our approach consists in making a first order approximation of the collision forces which can be calculated at a much higher frequency. This approximation is made through the use of a *local model* of the contact; in the case of a rigid tool interacting with a deformable object we make the (false) assumption that the surface of the deformable object remains in the same position between two updates of the physical model. The surface of the object that is in interaction with the tool is approximated by a simple geometric primitive such as a sphere or a plane. Thus the inter-penetration distance between the tool and the object can be found at a much higher frequency (i.e. simpler and thus faster distance computation) and, therefore, the force values supplied at an increased rate. This, even though only gross approximations have been used, leads to much smoother haptic interaction. Thus, we may divide this process into two categories:

- **Model update** Using the information about distance and the derivative of the distance, a simple *local model* of the objects surface is constructed. Thus this approximation is updated at the frequency of the physical model, and remains static in between those updates.
- **Haptic loop** Once the approximation of the surface is in place the distance

between the haptic position and the surface is minimized analytically using Lagrangian multipliers. This distance multiplied by the stiffness constant yields the force values.

4.4.1.1 The model manager

The model manager interacts with the physical model, passing the position and obtaining the minimum distance between the haptic probe and its closest contact point on the object of interaction. Using this distance and its derivatives, the manager will adapt the constraint surface to locally fit these values. The physical model must also transmit the local stiffness at the contact point.

To explain how the buffer model is updated let us consider the distance D as being defined by the function $f : \Re^n \rightarrow \Re$ given by the collision detection (Section 4.2):

$$D = f(x_i)\ (i = 1, ..., n) \tag{4.6}$$

As this function can be expensive to calculate, but is necessary to calculate the haptic feedback, we want to approximate it by a simplified buffer model. Hence, the buffer model can be defined as a function $\phi : \Re^{r+n} \rightarrow \Re$, which can be written as :

$$\phi(u, x) \approx f(x) \tag{4.7}$$

for certain $u = [u_1 ... u_r]^T \in \Re^r$. We will refer to $u_1, ..., u_r$ as the *parameters* and to $x_1, ..., x_n$ as the *variables*. The variables correspond to the configuration space dimensions. Hence the buffer model ϕ may be a geometrical entity whose parameters u will be *adapted* (see [Balaniuk, 1999] for details) in order to best fit the function $f(x)$.

For example, in Figure 4.7 the sphere represents the local model. Then the center and radius of the sphere are the variables of the local model. Hence, the variables, u, must be adapted as the position of the probe, x, changes in order to best approximate the distance D.

$$\phi(u, x) = \sqrt{(x_1 - u_1)^2 + (x_2 - u_2)^2 + (x_3 - u_3)^2} - u_4 \tag{4.8}$$

where (u_1, u_2, u_3) determine the sphere center and u_4 the sphere radius. The vector $x = [x_1\ x_2\ x_3]$ represents the probe Cartesian coordinates, and $u = [u_1\ u_2\ u_3\ u_4]$ denotes

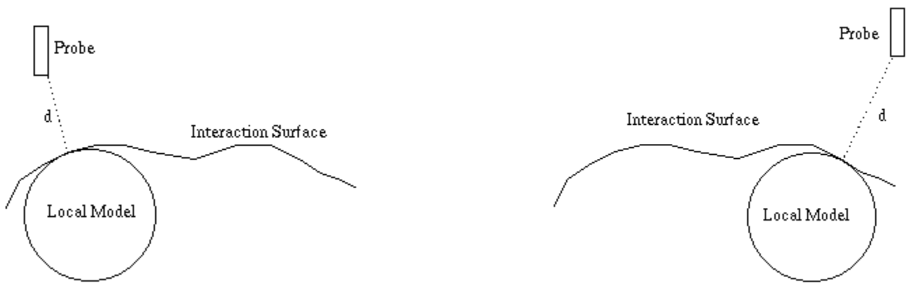


Figure 4.7: *The local model approximating the surface.*

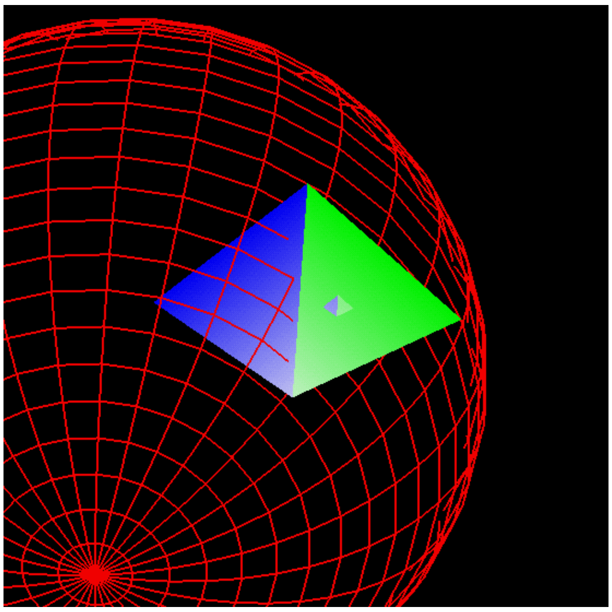


Figure 4.8: *Local model adapting itself to the surface of the tetrahedron*

the parameters to be identified by the model manager.

In Figure 4.8 we can see how the local model represented by the sphere adapts itself to the surface of the object. Hence the collision detection and response with the sphere is trivial to compute, and may run at a much higher frequency (i.e. $1kHz$) than global simulation.

In [Mendoza and Laugier, 2000], rather than using a simple geometric shape such as a plane or sphere, a set of triangles in the neighborhood of the contact are used to build the local model. It is shown that this approach has advantages when interacting with concave objects; since many deformable objects are or become concave during manipulation, this means that it is suitable for interaction with deformable bodies.

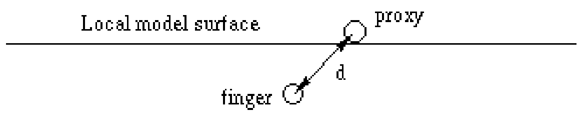


Figure 4.9: *Haptic interaction with the local model.*

4.4.1.2 The haptic loop

The haptic loop follows the constraint based rendering framework. The contact forces are estimated using the constraint surface, the proxy position, the user motion and the informed stiffness.

To implement this framework, at each time step of the haptic loop we obtain the haptic device configuration (the "probe position") and we change the proxy position to reduce its distance to the probe position subject to the constraint surface.

Generically, this problem can be written as :

$$\begin{array}{ll} minimize & D(x) = \frac{1}{2}(x-p)^T(x-p)\ s.t. \\ & \phi(x) \geq 0. \end{array}$$

where x is the proxy position and p the probe position.

Before moving the proxy we must test if it will collide with the constraint surface. The collision detection between the proxy and the surface can be simplified when the ϕ function defines a large surface. Using ϕ we simply compute the distance between the probe position and the constraint surface. If the distance is positive or zero then the probe is in the free space and there will not be collision; in this case the proxy can be moved directly towards the probe position. If the distance is negative then the probe is inside the obstacle. The proxy is moved until it reaches a contact with the constraint surface in its linear path towards the probe position. At this point it is no longer possible to move directly to the probe's position, but it may be still possible to reduce the distance between the proxy and probe positions, keeping the surface contact. The minimization problem becomes:

$$\begin{array}{ll} minimize & D(x) = \frac{1}{2}(x-p)^T(x-p)\ s.t. \\ & \phi(x) = 0. \end{array}$$

Zilles and Salisbury [Zilles and Salisbury, 1994] and Ruspini *et al.* [Ruspini et al., 1997] solve this problem using constraining planes and Lagrange multipliers. We use the same approach introducing a first order approximation for ϕ, corresponding to a constraint plane tangent to the constraint surface and going through the proxy position ($x = xp$). See [Balaniuk, 1999] for more details in this approach.

Force estimation In the constraint based framework force estimation becomes easy after the proxy's location is determined. Simple impedance control techniques can be used to estimate the forces, using the given local stiffness.

Given that our local model is a first order approximation of the collision response as given by Equation 4.3 in Section 4.3.3 the haptic force is

$$f = -\lambda(x-p) - \mu\frac{\partial(x-p)}{\partial t}(x-p) \tag{4.9}$$

where λ is proportional to the local stiffness, x the proxy position and p the probe position. μ is the damping of the contact, which is related to the coefficient of restitution by Equation 4.5.

In the constraint based framework, a number of additional effects can be easily implemented. Friction can be simulated by controlling the proxy motion. Texture can be simulated by controlling the constraint plane [Ruspini et al., 1997]. Force shading is used to make a regular polygonal surface to be perceived as a continuous smooth surface. All these effects have also been implemented in our approach.

4.5 Conclusion

In this chapter we have shown that the collision response is intrinsically based on the information given by the collision detection. However, detecting a collision for objects consisting of many hundreds or thousands of polygons is an expensive procedure.

Penalty based response treats the collision as a continuous evolution of contact forces in function of the inter-penetration; however, if the inter-penetration is not evaluated often enough the forces will exhibit unnatural discontinuities. Not only will this lead to an erroneous collision response, but also it will create a problem for haptic interaction.

For smooth haptic interaction using the PHANToM force-feedback device the output forces must be supplied at a rate of approximately $1kHz$. Since such frequencies are impossible to achieve for non-trivial applications on current computers, a local model is introduced to approximate the contact. This local model is of a simpler geometry than the whole model, allowing collision detection and distance calculation at high rates. Therefore the evolution of the contact forces is continuous guaranteeing smooth haptic interaction. It should be noted that this approach is *not* limited to rigid objects; for deformable objects, the position of the constraint plane can change from one simulation step to the next as we shall see in the next chapter.

Chapter 5

Echographic Simulator

Contents

Theory without practice cannot survive and dies as quickly as it lives.

Leonardo DA VINCI

5.1 Introduction

Echography, in general terms, exploits the information which an acoustic signal provides when it is reflected off a structure to determine the position and shape of the latter. The frequency of these acoustic signals lie within the range of 1MHz and 12 MHz, thus higher then the sounds which are perceptible by humans. Therefore, they are also referred to as *ultrasounds*.

History Even though they had already been discovered by P. Curie in 1880, they would have to wait until the 1970's to find an application in the field of medicine. Since, they have been widely used as an inexpensive and non-traumatic means of diagnosis. A common exam is the echography of the thigh to detect a thrombosis in the vein. A healthy vein will compress under the influence of an external force while a vein affected by thrombosis will only partially or even not at all compress, depending on the stage in the evolution of the illness. Depending on the pressure the practitioner applies with the echographic probe on the thigh, he will get an image from which the current state of the vein can be deduced, and hence a possible thrombosis diagnosed.

Motivations However, the learning process of this procedure is somehow long and only after approximately 1000 echographic exams an acceptable competence in acquired. The first 500 exams will have to be carried out under the supervision of an experienced practitioner. Virtual environments present an alternative to the conventional medical training scheme. It is possible to create an interactive 3D simulation environment, where the doctors can manipulate or cut dynamically and geometrically correct models of organs and tissues with an haptic interface. The idea is similar to using flight simulators to train pilots. Virtual environments give an environment where there is no risk to a patient, and therefore less stressful. They are interactive and three dimensional contrary to books. Virtual environments also give a unique advantage, as it is possible to generate arbitrary anatomies and pathologies, so that the doctors can be trained for cases that are not frequently encountered.

Objectives The goal of this work is to lay the groundwork for the development of an echographic simulator with force feedback. In the final system, the trainee will be looking at artificially generated echographic images, and interacting with a computer simulated dynamical thigh model through a haptic interface. Constructing realistic but computationally efficient models is the main challenge in developing a virtual reality

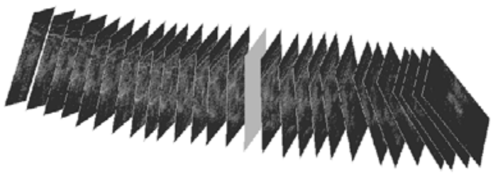


Figure 5.1: *Creating an image from a set of real echographic images. The highlighted image was obtained by interpolation.*

training simulator. In this application it is necessary to have models for deformable tissue being manipulated by the doctor as well as models to construct artificial echographic images.

In this chapter, a dynamic model of the human thigh based on experimentally determined deformation characteristics will be presented, followed by a discussion of the results and future directions.

5.2 Previous Work

5.2.1 Echographic Image Generation

[Wibaux et al., 1997] generated echographic images directly from the geometric model of the anatomy. The echographic process itself is simulated by modeling the refraction of the signal caused by the organs. In the quest for real-time performance a simplified refraction model is used to generate the images.

Henry [Henry, 1997] examined how a set of echographic images could be used to construct a more general model that would take into account the orientation and pressure exerted with the probe on the thigh. The approach is based on an interpolation method; from a real set of echographic images taken, it will build an echographic image for *any* position and orientation of the probe (see Figure 5.1 and 5.2)[1].

Further, it is possible to deform the interpolated image obtained with respect to the pressure that is applied to the probe. The criteria on which this deformation depends includes important factors such as the arterial and venous pressure. By modifying these criteria we can simulate a set of pathologies on which medical students could be trained for the identification of the latter.

[1]Figures kindly provided by the TIMC-GMCAO project

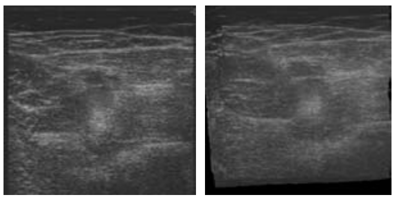


Figure 5.2: *On the left we can see the real image, and on the right the image obtained by interpolation.*

5.2.2 Physical Modeling

However, for a meaningful echographic simulation it is of paramount importance to consider the forces involved in such a procedure since they will dictate the deformation of the tissue, and therefore, the subsequent image that is acquired.

[Basdogan et al., 1996] developed an interactive 3D model of the human thigh along with a set of surgical tools to simulate procedures in virtual environments. Laffont [Laffont, 1997] studied the implications of the deformation of such a system, which are essential in the development of a realistic simulator coupled to a haptic interface. He based his model on a system of inter-connected springs.

Mass-spring networks already been successfully applied to areas such as facial animation [Lee et al., 1995] and have the decisive advantage of being easy-to-implement and faster than finite-element simulation, even though lately considerable speed-ups have been achieved in this domain through the use of pre-calculated matrices. This has resulted in, for example, the real-time simulation with force-feedback of the human liver [Bro-Nielsen and Cotin, 1996] based on the *Visible Human* dataset. However, the model used is inconvenient in the sense that the assumption is made that the tissue is hyper-elastic and linear in its deformation. This is clearly not the case. As biomechanical experiments confirm, the stress-strain curve of human tissue is non-linear.

[Lee et al., 1995] tackles this problem by using a three-layer model which takes into account the properties of the epidermis, the fatty sub-cutaneous tissue, the muscle, and the bone. The great difficulty for such a system lies in the identification of the parameters of the individual springs that will give the same results to an external force being applied as the measurements in the real world.

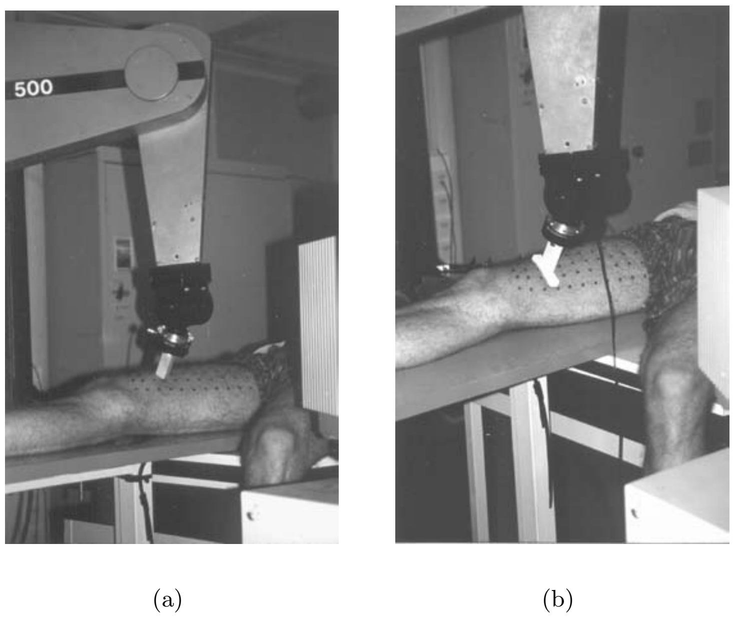


Figure 5.3: *The two probes used for measuring the behaviour of the thigh. A probe with a punctual contact (on the left) and the pseudo-echographic probe with a larger contact surface (on the right).*

5.3 Modeling of the Thigh

We have constructed a dynamic model of the human thigh based on experimental measurements of its elasticity. The work presented in this section has been done in collaboration with Cenk Cavusoglu of UC Berkeley in the framework of the France-Berkeley cooperation.

5.3.1 Experimental Setup and Data Acquisition

In order to model an object such that its behavior corresponds to reality, measurements must be taken on the real object. In this case we are interested in the deformation of the thigh with respect to an external force which is applied. Intuitively we can affirm that the deformation of the thigh is not the same depending on the shape of the object used to provoke this deformation, or more precisely, the contact surface of that object. Since our aim is to build a generic simulation which will allow a physically correct behavior which is almost independent of the object we choose to deform the thigh with, two different objects have been used to measure the behavior of the thigh in terms of penetration distance with respect to the external force being applied.

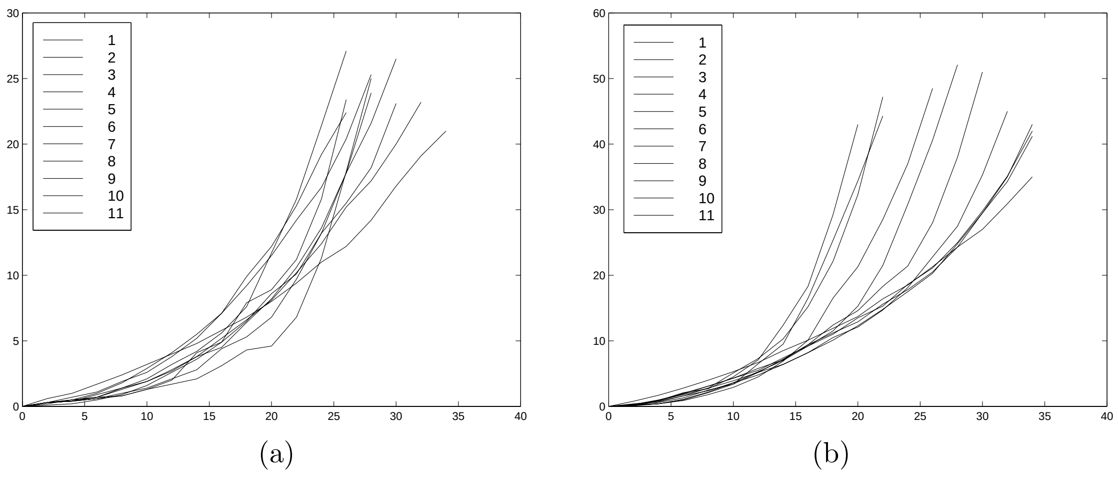


Figure 5.4: *Plot of the reaction force in function of the penetration distance at 11 different points on the thigh using a probe with a punctual contact (on the left) and the pseudo-echographic probe with a larger contact surface (on the right).*

The first of these has a tip of pyramidal shape to provoke a punctual force response, while the second one has the same contact surface area as a typical echographic probe. These pseudo-probes are then mounted on a force sensor which in turn is mounted on a PUMA articulated arm. The probe is then positioned perpendicularly to the surface of the thigh at each of 64 points where measurements will be taken. These points are regularly distributed over the area where the echography is performed when trying to detect a thrombosis in the vein. The robotic arm then advances $2mm$ using the reference of the end effector, i.e. the probe pushes along the axis which is normal to the surface of the thigh at the given point. The force is recorded and the procedure is repeated up to an upper force limit.

Figure 5.4a shows the non-linear relationship between the penetration distance and the reaction force at 11 different points along the thigh. The difference in the curves may be accounted for due to the fact that the thigh is not homogeneous and that the amount of the material varies with the location (e.g. in some regions fatty tissue might be pre-dominant, while in others there may be very little separating the epidermis from the bone).

Figure 5.4b plots the values of the forces for the same points as above but using the second probe with a larger contact surface. As one might expect this results in a larger force for the same penetration distance since the external force applied is distributed over a much larger area (note that the pressure is defined as force per unit area).

The assumption is that from these two distinct sets of data it will be possible to make a model which will respond correctly to not only to the two probes used for measurement purposes, but also other probes of a different shape.

5.3.2 Model construction

Based on the experimental data and the computational requirements, a two layer lumped element model is chosen. Multi-layer models were also used by several authors in the literature, for example in [Lee et al., 1995] for facial animation or in [Nebel, 2001] for modeling soft tissue with volumetric finite elements.

The two layer model is composed of a surface mesh of masses, linear springs and dampers, and a set of nonlinear springs orthogonal to the surface to model volumetric effects by giving normal support to the surface mesh (see Figure 5.5). One end of the nonlinear springs is fixed assuming that it is attached to the bone, which remains immobile. Then, the deformation-force relation of the nonlinear springs is of the form

$$f(x) = \frac{x}{ax + b} \tag{5.1}$$

where x is the change in length of the spring. The nonlinear spring response is chosen to model the incompressibility of the thigh after a certain deformation. In the model, the values of the surface elements are chosen uniform whereas the parameters of the nonlinear spring vary around the mesh to model heterogeneous nature of the thigh mentioned above, while keeping the number of unknown parameters small [d'Aulignac et al., 1999a].

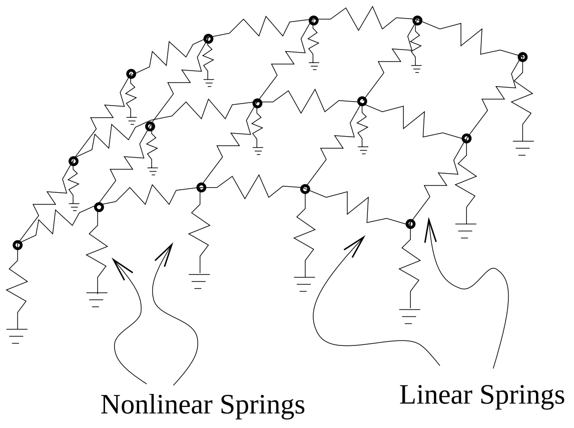


Figure 5.5: *Two layer model of the thigh.*

It should be noted that the non-linear springs make no assumptions on the composition of the interior of the thigh. There is no notion of the varying thickness of the soft tissue as the forces calculated are based only on the experimental data available and the displacement normal to the surface. In this sense this approach differs drastically from the FEM, where the topology is fixed and the elasticity chosen according the material properties.

However, interestingly, the non-linear spring relationship can be rewritten if we set $a = \frac{1}{AE}$ (where A is the cross-sectional area of the spring and E the elasticity) and $b = aL_0$ (where L_0 is the initial length between particles). Equation 5.1 then becomes

$$f(x) = \frac{x}{\frac{x}{AE} + \frac{L_0}{AE}} \tag{5.2}$$

$$= \frac{AEx}{x + L_0} \tag{5.3}$$

and since $x = L - L_0$

$$f(L) = AE\frac{(L - L_0)}{L} \tag{5.4}$$

Hence, we rediscover a variant of Almansi's strain mesure encountered in Equation 2.22. The values for the elasticity may then be evaluated as $E = \frac{1}{Aa}$, and the initial length as $L_0 = \frac{b}{a}$. This allows us to reconsider the internal forces as functions of strain and elasticity.

5.3.2.1 Parameter Estimation

Estimation of the model parameters from experimental measurements is a critical part of the modeling. Several types of methods can theoretically be used for solving this problem: an Artificial Neural Network approach requiring to make use of an adapted modeling technique [Szilas and Ronco, 1995], the Simulated Annealing approach which often lead to get stuck in some local minima when dealing with our type of function [Bonomi and Lutton, 1988], the Genetic algorithms [Goldberg, 1989] which have already given some good results when identifying the physical parameters of a deformable object (see [Louchet et al., 1995] and [Joukhadar et al., 1997]), and the Steepest Descent method [Minoux, 1983] which was used for this application.

We have used a two step optimization approach based on nonlinear least squares estimation.

1. The experimental data of the indentor is first fit to a simple model without any surface elements, i.e. only the nonlinear springs.
2. The results of the previous fit is used as the initial conditions for the parameter estimation of the complete model; the parameters of the surface *and* nonlinear springs are adjusted in order to best match the experimental data of the indentor *and* the pseudo-echographic probe.

This approach is chosen to avoid problems with local minima. We have obtained better results with this method than using genetic algorithms, perhaps, as the number of parameters is small, while genetic algorithms are usually better suited for problems involving large numbers of unknowns.

One thing to note here is that the parameter estimation is based on a simplified interaction model between the tissue and probe, making the assumption that the indentor will principally act on one spring since the contact area is small, while the larger probe acts on three nodes simultaneously. In this simplified interaction model, the nodes that are not in contact with the probe are kept stationary. This simplifies the force calculation as only a small number of springs are deformed for any given sample in the dataset.

The mean absolute error between the measured values and the values estimated by the model is 1.05 Newtons, with standard deviation 1.84 Newtons, over the whole dataset (i.e. all measurement samples obtained with both probes). This is equivalent to an approximate error of 5 percent. The distribution of the error can be seen in Figure 5.6, showing how many of the values estimated by the model exhibit a given error with respect to the real measurements.

5.3.3 Graphical Model

The area of the thigh of interest for an echographic exam is relatively small as compared to the whole leg. Thus it is sufficient to model the deformation only in this area. However, to give the user an increased sense of realism and as landmarks for orientation in the virtual environment, we also render the lower leg and other side of the thigh. These are not deformable and thus the position of the vertices will not change during simula-

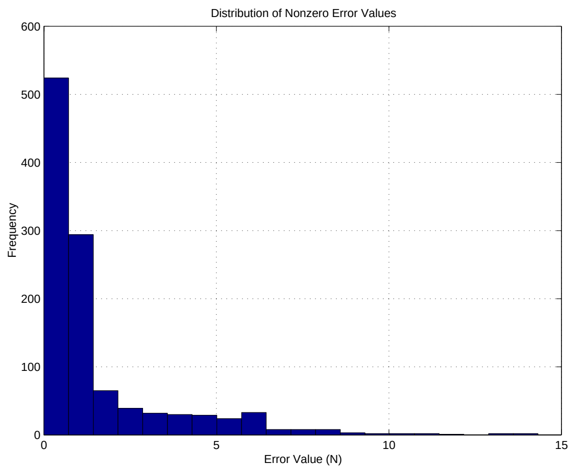


Figure 5.6: *Frequency distribution of the error between the measured values and the values estimated by the model.*

tion. It is therefore possible to transfer this data only once to the graphics hardware, and subsequently it will put only minimal demands on the CPU.

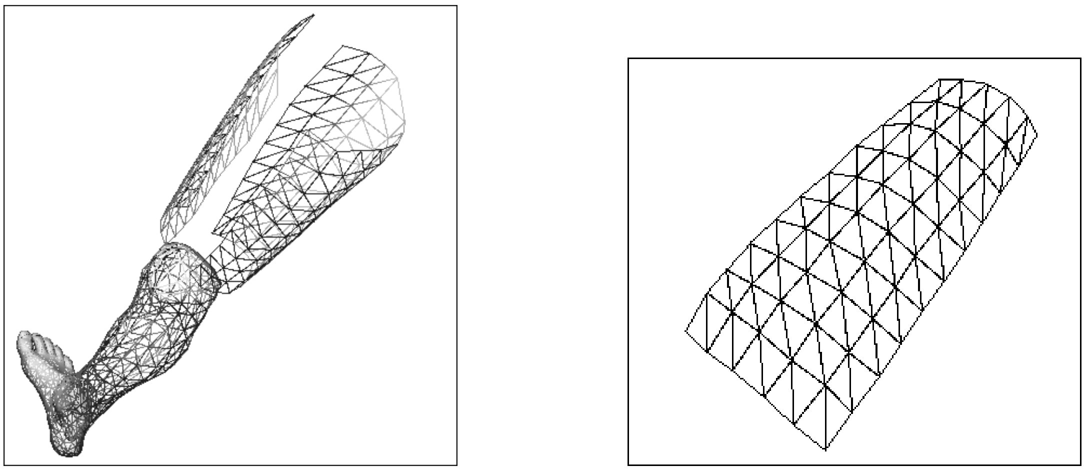


Figure 5.7: *Decomposition of the parts of the leg. The part on the left is the only part where deformation is simulated.*

Figure 5.7 shows how the area where deformation is simulated is integrated with the rest of the components that represent the leg only graphically.

5.4 Dynamic Resolution

5.4.1 Explicit integration

Given that we know the internal and external forces acting on each mass, we may deduce its acceleration. The most straightforward way to solve for the movement of the mass-points of our model is using explicit integration. However, as discussed in Section 3.4.1, explicit integrations suffers from stability problems. This means that for a given stiffness there is a limit on the value of the timestep that can be used (see Section 3.4.1 for linear stability analysis). Moreover, the larger the stiffness, the smaller the timestep must be when using explicit methods. In this case the stiffness of the spring is nonlinear as measured experimentally. The viscosity, however, has not been measured. We have adjusted the value for the viscosity by hand to ensure linear stability accordingly to Figure 3.11. For example, if using simple explicit Euler integration or 2nd order Runge-Kutta, we must have damping to maintain linear stability. Since real-time is our first and last predicament, the timestep is directly related to wall-clock time, and therefore to the execution speed of the machine we are using. On an SGI Octane 175 MHz with a single R10000 processor the timestep imposed by the real-time constraint forces us to use artificially large inertial values, i.e. large mass, to ensure linear stability. The effect on the simulation is a unnaturally slow reaction to external excitation force. Higher order methods do not allow us increase speed since the stability region does not double as we double the number of stages per timestep.

5.4.2 Implicit integration

Unsatisfied with explicit integration for our simulation, we have decided to investigate the suitability of implicit methods. Since speed rather than accuracy is our main concern, we have opted for the *semi-implicit* Euler method (Equation 3.33) having excellent stability (see Section 3.4.2). For each step in the simulation the forces on the mass-points are calculated and the linear system is solved using the conjugate gradient method.

$$\Delta y \left[\frac{1}{h} I - \beta \frac{\partial f}{\partial y}|_{y=y_{rest}} \right] = f(y_0)$$

Since we only solve the linear system once per timestep our resolution is only semi-implicit and not guaranteed to be stable, however, it is much faster to solve and has not given us stability problems in practice. Further, the required Jacobian, $\frac{\partial f}{\partial y}$, was pre-

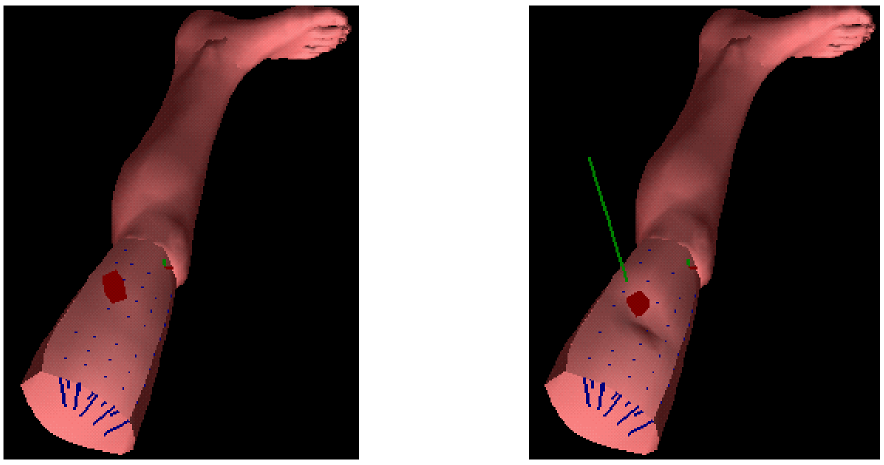


Figure 5.8: *Dynamic deformation of the thigh under pressure.*

computed only once before the start of the simulation (N.B. in this case this was done numerically, hence expensive to calculate). This is, of course, a false assumption to make since we use non-linear springs in our model; hence, the time-history of the integration is not “accurate” (nevertheless, because inertial and damping terms have been estimated experimentally this seems of limited importance). However, since forces are re-evaluated each time step, the equations will eventually converge towards the “true” equilibrium point.

5.4.3 Experimental Results

Figure 5.8 shows the model as it has been built in our simulation system. A force is being applied on the thigh using a probe which provokes a deformation which is in accordance with the measurements taken. For collision detection between the objects the algorithm outlined in [Lombardo et al., 1999] is used, while the response to a collision is determined by the approach described in Section 4.3.3. The computational speed on a Silicon Graphics R10000 machine is in the order of 100 frames per second of animation, which lets us envisage our simulation system as part of a working echographic simulator which operates in real-time [d’Aulignac et al., 1999b].

Input Neurons (Force)	88
Hidden Neurons	100
Output Neurons (Displacement)	88
Training Examples	1260

Table 5.1: *The parameters of our multi-layer perceptron.*

5.5 Static resolution using artificial neural networks

How can the MLP described in Section 3.6.1 help us to approximate the deformation of soft tissue? From the measurements taken in Section 5.3.1 we know a set of forces needed to create a given deformation on a number of points. Ideally we would like to find a function M that, given the forces f, will return the displacements u.

$$u = M(f) \tag{5.5}$$

From the measurements we know that there exists a *non-linear* relationship between u and f; hence the function M must also be non-linear. Neural networks can be considered as universal approximators of functions, including non-linear ones. Hence the function M may be approximated by a neural network given enough information about the function, i.e. the measurements.

5.5.1 Architecture

We want to use the artificial neural network to learn the displacements of the nodes given a certain force. The inputs will be the forces applied on the nodes, and the outputs their displacement in the direction normal to the surface. Thus there are as many input and output neurons as sample points on the surface of the thigh (Table 5.1).

Each training example consists of the vector f as the inputs, and the vector u as the targeted output given by the measurements in Section 5.3.1. This data only gives us information about the displacement of the nodes where we are applying a force. We assume that the forces recorded with the punctual probe (Figure 5.3a) only create a displacement at that point. Hence for the training examples acquired with this probe, vectors u and f have only one entry each.

For the probe with the larger surface area (Figure 5.3b) it is assumed that the force arises from the displacement of three points. Hence for the training examples acquired with this larger probe, vectors u and f have three entry each.

However, if we could measure the displacement of the other nodes, their displacement could also be included in the training data. In the case of the thigh this is not a major issue, since forces provoke a very local deformation, i.e. a force on a node has a small effect on the displacements of the neighboring nodes and a negligible effect on the displacement of nodes further away.

5.5.2 Experimental Results

We use a MLP with a sigmoid activation function at the hidden layer, and linear inputs and outputs. The number of hidden neurons is chosen to be 100. Using the training data collected from measurements the network is trained over 100 epochs[2] using on-line update of the weights by back-propagation (see Section 3.6.1.1 for details). The order in which the patterns are presented vary randomly from one epoch to the next to ensure better generalization. The mean-squared-error at the end of training is 0.041384.

Figure 5.9 shows how the output of the neural network differs from reality for one given point. The error with respect to the training data is rarely over $1N$ and the non-linear behavior suitably approximated.

The graphical output of the simulation (Figure 5.10) shows the deformation of the surface when a force is applied on one node.

5.5.3 Conclusions

We have shown that using a neural network we can approximate the deformation given a suitable set of training examples. A difference with respect to the mass-spring model in Section 5.3.2 is that no assumptions on the mechanical relationship between particles are made. Hence there are no topological limitations to the model's behavior. The neural network is trained directly using the measurement data eliminating the need to estimate the parameters of the springs that make up the model. However, there are some limitations: if topological changes are introduced, the function M would change, which would mean that the neural network approximation would no longer be valid. Also enough training examples must exist that suitable describe the deformation of the object.

[2]An epoch is equivalent to presenting all the training data to the network once.

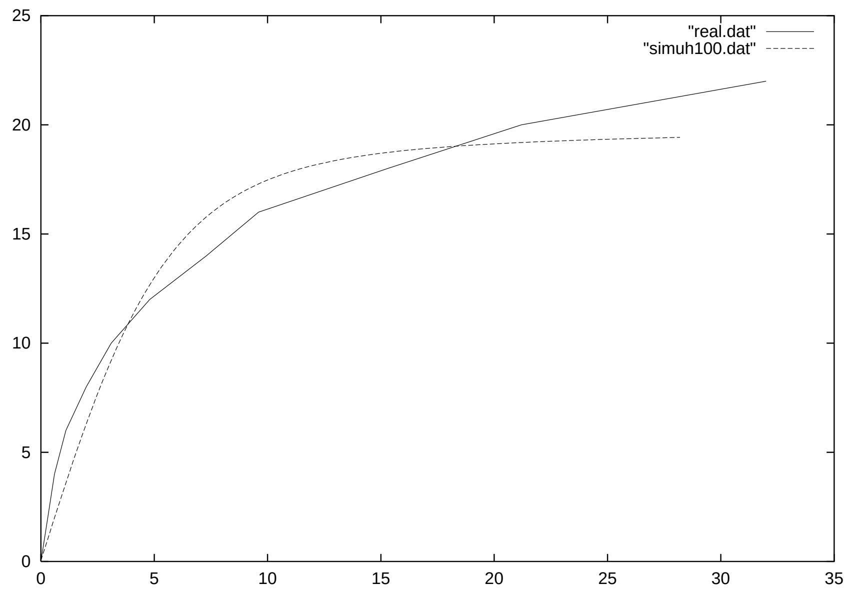


Figure 5.9: *Comparison of the displacements with respect to the forces applied at one point. The continuous line represents the real data while the dashed line is the output of neural network.*

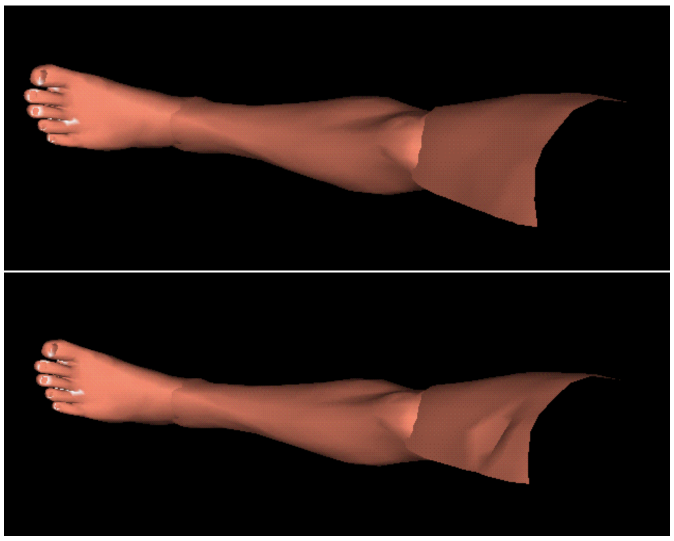


Figure 5.10: *Example of the deformation of the thigh when applying a force on one node.*

Computationally, once the neural net has been trained, the model is very fast since the evaluation of the static equilibrium for an external force is achieved in one evaluation of the outputs. For this case this means only $2 \times 100 \times 88 = 17600$ multiply-add operations and 100 sigmoid evaluation are required. This is considerably less than for the dynamic integration process for the mass-spring system; note that the simple evaluation of the spring forces already requires 8704 floating point operations, and the process may be repeated hundreds of times in an explicit integration before attaining the static equilibrium.

In the case of the application described we only have a local description of the deformation, i.e. the displacement is only known at the nodes where the forces are applied. However, in principle, if the displacements of the other nodes could be measured, this data could be incorporated into the training examples. It could, for example, be envisaged to use the output of a mass-spring of finite-element model to train a neural network, allowing much faster resolution. Since the neural network is fully connected, theoretically, it should be able to take this into account without any additional modifications to the procedure described here.

5.6 Interaction

In the previous sections the deformation of the thigh has been modeled. In this section we shall examine how to interact with the model we created using a virtual echographic probe. In Section 5.6.1 the collision detection and response are discussed while the haptic force-feedback interface is detailed in Section 5.6.2.

5.6.1 Collision detection and response

The virtual echographic probe we use to interact with the model of the thigh is simple in shape, and can be modeled as a parallelepiped. However, the surface of the thigh is represented by many polygons and can become concave during interaction.

We therefore choose to use the approach by [Lombardo et al., 1999] mentioned in Section 4.2.2. Hence to find the triangles of the thigh in collision with the probe, we render the surface (enabling the picking mode) in a orthogonal bounding box equivalent to the shape of the tool. This return the polygons inside the tool, i.e. in collision.

Distance computation Finding the polygons of the thigh in interaction with the tool will not automatically compute the distance. We make the assumption that the tool is touching the thigh using the front side of the parallelepiped. Hence the inter-penetration distance is computed as the distance between a polygon and the front tip of the tool.

Collision forces If an inter-penetration distance has been found a non-linear penalty force is calculated according to Section 4.3.3. This force is then distributed over the three particles of the colliding polygon using the barycentric coordinates of the point on the triangle closest to the tip of the tool.

5.6.2 Haptic Interaction

In Section 5.4 we have seen that our simulation runs at approximately $100Hz$ on an SGI Octane 175 Hz. This means that we check one-hundred times for the collision, calculate the appropriate penalty for a penetration (if present), and update the position of the vertices of the deformable area according to external and internal forces each second. Even if this guarantees a stable simulation visually, i.e. at least 10 frames a second, haptic force feedback may be unsatisfactory. By this we mean that is the contact forces are transmitted to the PHANToM (see Section 4.4) at the mentioned simulation rate, unacceptable trembling may result to the low frequency of the force update. In Section 4.4.1 we propose the use of a local approximation of the contact which we shall use in this application.

5.6.2.1 Implementation

Figure 5.11 shows that there are two separate threads running at the same time. In the simulation thread an approximation of the contact is created depending on the position of the stylus controled by the the haptic device. In this case the local model is a plane parallel and passing through the nearest polygon (triangle). Any forces sent to the force feedback device are now calculated using this local approximation of the contact [d'Aulignac and Balaniuk, 1999].

5.6.2.2 Experimental Results

Figure 5.12 graphically shows the local model. As mentioned in the previous chapter, interaction forces with the simplified model are much less costly to evaluate, and there-

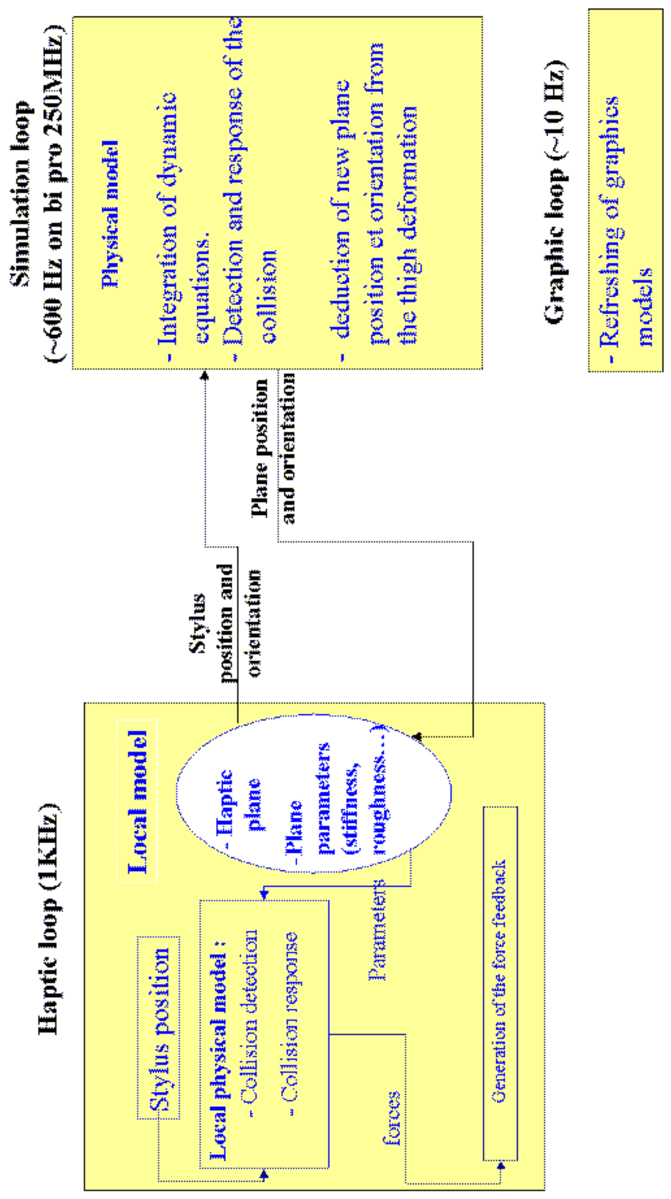

Haptic loop (1KHz)
Local model
Stylus position
Local physical model :
- Collision detection
- Collision response
- Haptic plane
- Plane parameters (stiffness, roughness...)
Parameters
forces
Generation of the force feedback
Stylus position and orientation
Plane position and orientation
Simulation loop (~600 Hz on bi pro 250MHz)
Physical model
- Integration of dynamic equations.
- Detection and response of the collision
- deduction of new plane position et orientation from the thigh deformation
Graphic loop (~10 Hz)
- Refreshing of graphics models

Figure 5.11: *Graphical overview of the two threads: the force feedback (on the left, running at* 1kHz*) and the simulation (on the right).*

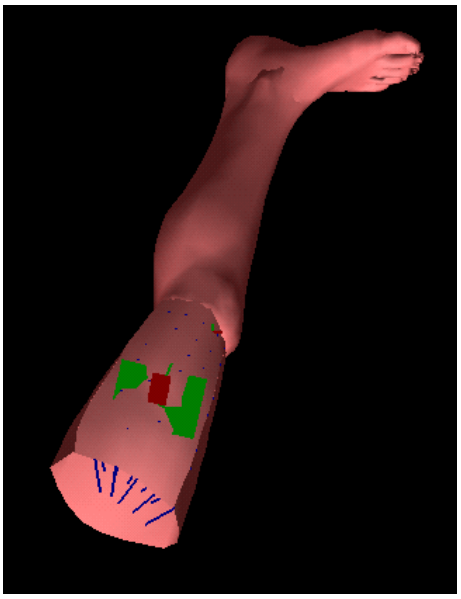


Figure 5.12: *The local model to calculate the contact force for the force feedback device is shown as the green square.*

fore, can be calculated at a much higher frequency that will give a much smoother force feedback response. Figure 5.14 shows the user interactively deforming the surface of the thigh.

Figure 5.13 shows the evolution of the magnitude of the force during a manipulation of the virtual thigh. When using the buffer model we obtain the dashed curve. However, for a similar interaction without the buffer model we obtain the solid curve. This curve presents serious discontinuities; when forces return to a zero value the user has lost contact with the surface of the thigh due to the trembling. This trembling is due to the large variation of the forces [d'Aulignac et al., 2000].

5.6.2.3 Conclusions

The local model we have implemented is a static approximation of the *contact*. By this we mean that collision detection and response are calculated using a simplified geometrical model at a much higher frequency using a multi-threaded architecture. However, the *deformation* is not modeled; the local model's parameters are constant during one step of the physical simulation (the latter calculating deformation and global collision detection).

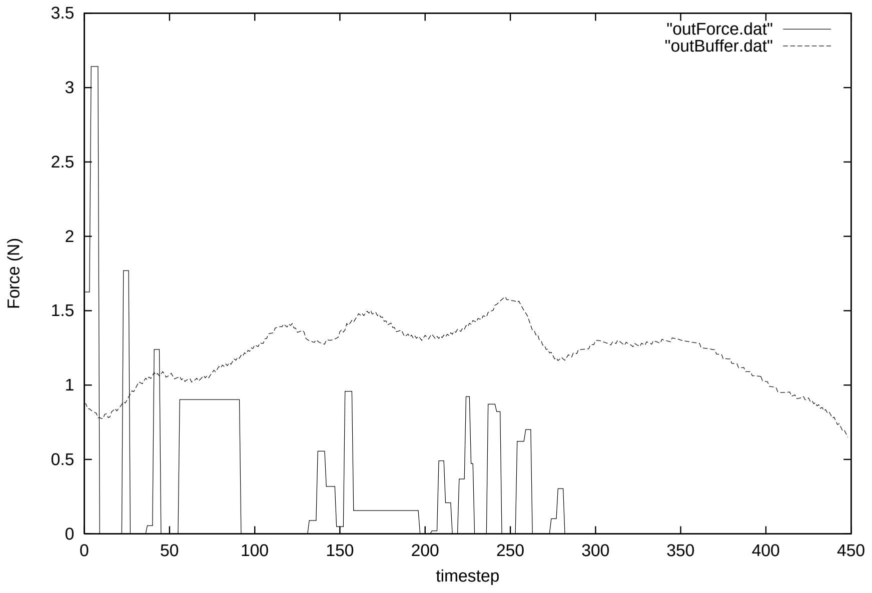


Figure 5.13: *Evolution of the force over a time interval using the haptic buffer model (dashed), and without (solid).*

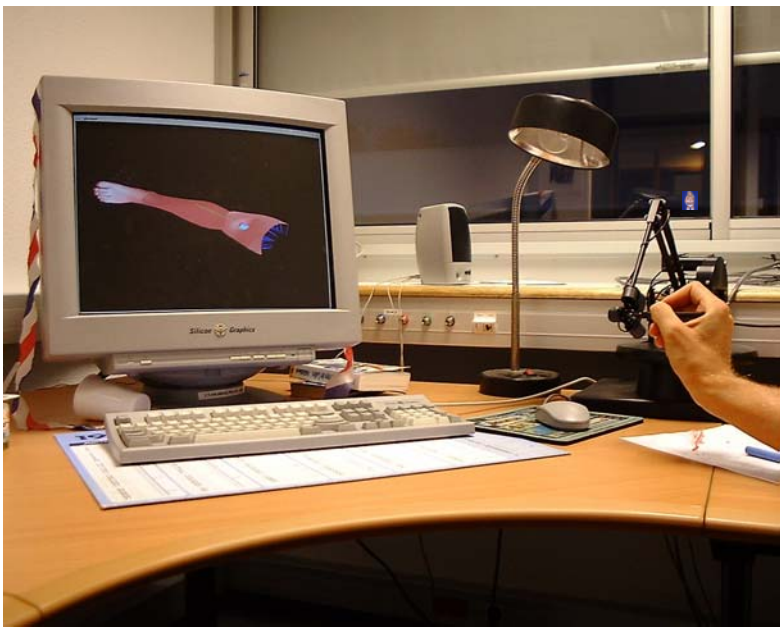


Figure 5.14: *The thigh model manipulated by the phantom.*

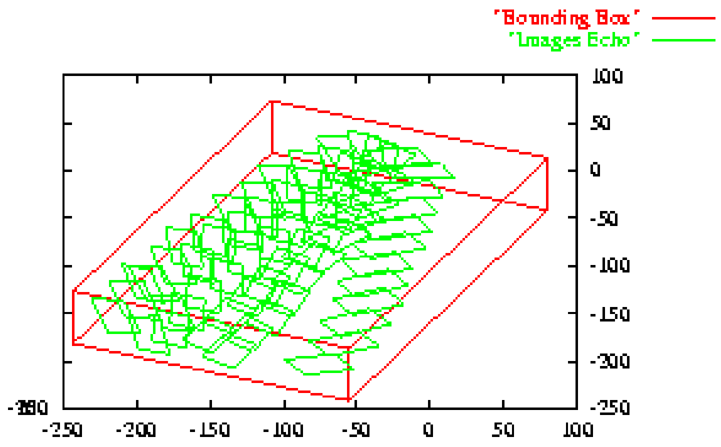


Figure 5.15: *Echographic slices arranged in a 3D volume.*

Therefore, an interesting extension of the presented approximation the contact, would be to combine it with low order models of the deformation such as presented in [Cavusoglu and Tendick, 2000] or [Astley and Hayward, 1998].

5.7 Echographic Image Generation

The work described in this section has been done in collaboration with the GMCAO project of the TIMC-IMAG[3] laboratory, who have co-supervised the internship of Stéphane Vieira.

As a pedagogical tool our simulator described so far is sterile. Images corresponding to the actual deformation must be generated to teach practitioner the *correlation* between deformation of the surface of the thigh and the resulting echographic image. Hence for any position and orientation of the probe on the surface of the thigh we must generate an image. Since we can not acquire nor store an infinite amount of of images for all points we will have to resort to some kind of interpolation technique. However, we must first prepare the data.

To collect this data we have taken echographic images at the same point as we have measured force-displacement curves. These echographic slices are then arranged within a bounding box (see Figure 5.15) which will be divided into a 3D voxel map. The voxel map is then filled depending on the intensity of the pixels of the echographic images

[3]In Grenoble, France

within each voxel. These voxels are of equilateral shape, and their size is chosen with respect to the available live memory on the machine (i.e. no swap).

Voxel Map Size In this case we have used a voxel size of $0.3mm$ leading to a memory requirement of approximately $120Mb$. Since the resolution of one pixel on the original echographic images is $0.1mm$ there will be a small loss in resolution due to memory limitations.

Interpolation It is clear that the for the number of sample images we have acquired the voxel map will be far from full. To fill the blank voxels we use interpolation. The TIMC laboratory has provided us with such an interpolation tool within the framework of a collaboration. Thus the voxel map is filled in the preprocessing stage once and, thereafter, slices through the voxel map may be generated in real-time. The resolution of the generated image is, of course, dependent on the resolution of the voxel map [Vieira, 2000].

Once the voxel map is filled, we may calculate an image for any given position and orientation on the modeled area of the thigh. Given the data can be stored in live memory this procedure is performed in real-time.

5.7.1 Image Deformation

Motivation The images to construct the voxel map where acquired without deformation: the echographic probe was placed on the sample point and minimal pressure was applied to obtain the echographic image. Hence an image generated from the voxel map will be non-deformed. However, to detect a thrombosis in the thigh, the diagnosis is based on the deformation of vein. It is therefore essential for a useful simulator to take this phenomenon into account.

Problem statement The main difficulty in modeling the deformation lies in the fact that each structure has its own way of reacting under pressure. Arteries have almost no deformation while non-pathological veins flatten. Superficial soft tissues have a near linear deformation. Deep tissues and bones have negligible deformation [Troccaz et al., 2000].

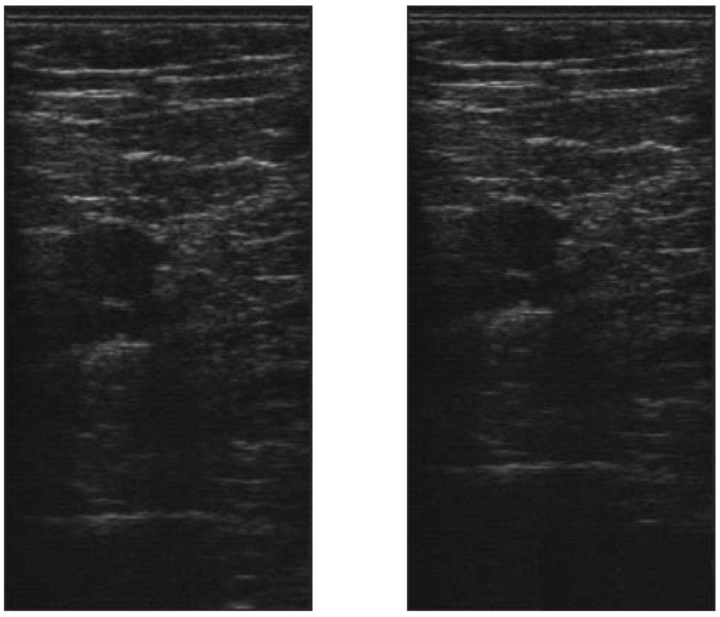


Figure 5.16: *Echographic images generated for a given location and orientation, with (right) and without (left) deformation.*

Existing Approaches The deformation of the image when pressure is applied can be obtained through the approach by [Henry, 1997]. Given a segmentation of the undeformed echographic image to find the artery, vein, and soft tissue areas their deformation can be calculated by taking into account parameters such as arterial and venous pressure. As mentioned before, by modifying these criteria we can simulate a set of pathologies. Alternatively, a deformation function can be calculated directly from the position of features on a deformed and undeformed image. However, both these approaches rely on a segmentation having been performed previously.

Since our voxel map is not segmented into the different anatomical structures we have momentarily adopted a linear deformation regardless of the tissue. Therefore, future work on this subject should examine this problem in detail; a possiblity might be to use precalculated deformation functions as described in [Troccaz et al., 2000].

5.7.2 Experimental Results

Figure 5.16 shows how the echographic image changes as an increasing pressure is applied to the surface of the thigh by the means of the virtual echographic probe. The probe's position is given by the force-feedback device (see Figure 5.17).

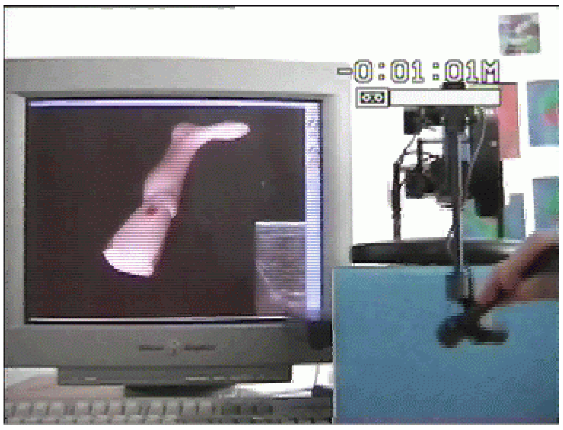


Figure 5.17: *Online generation of echographic images when interacting with the thigh using the force-feedback device.*

5.8 Conclusions

In this chapter we have described the building blocks to prototype of an echographic simulator.

Measurements have been taken on a real human thigh using an articulated robot arm in combination with a force sensor. The force-displacement curves we have obtained on different sample points on the thigh demonstrate both the non-linear and non-homogeneous elastic nature of the material.

Furthermore, we have built a mass-spring model of the area of the thigh were the measurements were taken. The topology of this model is based on the position of the sample points and by making assumptions on the anatomy based on the experimental data acquired. Hence, we have chosen a layered model with non-linear springs that approximate the measured force-displacement behavior.

As the elasticity of the thigh is non-homogeneous a protocol of parameter identification has been installed. Using a method of steepest descent we aim to represent *all* the data samples with the lowest possible error.

However, the deformation using the mass-spring network is intrinsically related to its topology and the stress-strain relationship of the spring. Using a neural network we have aimed at modeling the deformation *directly* from the data, without the need to make any assumptions on the topology. This approach has delivered promising, preliminary results that justify further research on this topic. It should also be mentioned that this method is not only limited to using experimental data directly, but the neural network's deformation function may also be trained on the output of an accurate numerical model (for example based on FEM).

To model the dynamics we have investigated the use of both explicit and implicit methods. We come to the conclusion that the overhead of solving a linear system at each time step using a semi-implicit integration is preferable to explicit integration. Because of the stiffness of the springs the latter approach is obliged to take extremely small timesteps to maintain stability.

To calculate the collision response we use the distance of penetration as a measure to calculate the penalty based response forces. However, if these forces are supplied to the force-feedback device at the rate of the physical simulation of the deformation ($100Hz$ using semi-implicit integration with constant Jacobian), unacceptable trembling will be the result. To remedy this problem we apply a local approximation of the contact, that will calculate the collision forces at a much higher rate (i.e. $1kHz$) in a separate program thread. This process suffices to largely improve the haptic sensation.

Lastly we have integrated our physical model with a generator of echographic images. Previous methods have interpolated images along one privileged direction of the thigh (namely the direction of the vein). Since, in this case, we want to provide an image at any position of the modeled area, we have created a voxel map based approach. The echographic images acquired at the sample points are used to partially fill the voxel map; using interpolation we then obtain a full 3D representation of the echographic data. From this voxel map, echographic images at any position or orientation can be found.

Chapter 6

Interactive model of the human liver

Contents

Like a surgeon,
Cutting for the very first time...

Weird Al YANKOVIC

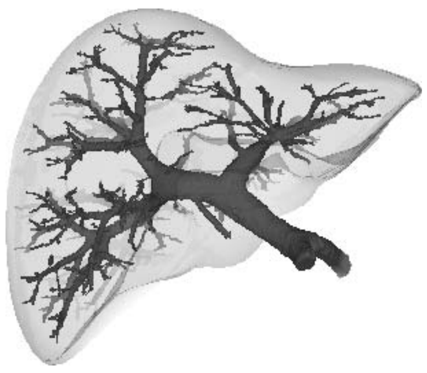


Figure 6.1: *Representation of the human liver with its vascular network (source: projet Epidaure, INRIA Sophia-Antipolis).*

6.1 Introduction

Objectives With increasingly complex surgical procedures doctors need to learn new skills. Medical practitioners are often trained by first observing a procedure followed by supervised practice on an actual patient. However, to practice during a real intervention may be dangerous for the patient. It is our long-term goal to build a surgical simulator for hepatic procedures to be used in medical training. A cornerstone of such an endeavor is the underlying physical model of the liver. Very little strain/stress experimental data on the liver is currently available, and thus we want our model to be as realistic as possible in a qualitative way. Finally, since the simulation is to be used interactively real-time performance is of paramount importance.

Relevant mechanical characteristics of a human liver The weight of the liver is approximately $1.5kg$ for an average male human. It is located in the abdominal cavity and is a very malleable body; its exact shape strongly depends on the contact interactions with the other organs located in its vicinity. It is composed of tree major parts:

1. the Parenchyma, which presents a mechanical behavior near to those of a sponge full of liquid;
2. a complex vascular network, irrigating the liver;
3. an elastic skin called Capsule of Glisson, which is quite elastic and stiffer than the Parenchyma.

Different works have aimed at analyzing the mechanical properties of the liver.

In vitro [Dan, 1999] has conducted rheological tests on material samples from the human liver of a deceased person. The samples of the Parenchyma are obtained from different areas of the liver and placed under a force sensor. Through successive displacements strain-stress curves for compression are obtained. To test the elastic properties of the outer skin of the liver, strips of it have been removed from the surface. These strips are then clamped into a measuring device and data for the relative elongation are obtained.

In vivo Alternatively, [Carter et al., 2000] and [Ottensmeyer and Salisbury, 2001] have obtained force-displacement data during an operation. A tool with a force sensor is inserted into the abdominal cavity and placed on the surface of the liver. Then the forces for increasing displacements are recorded.

The above approaches have recorded significantly different values for the magnitude of the forces. One explanation for this result may be the different mechanical properties of the tissue when it is alive and dead. However, perhaps more importantly, the structure of the tissue is fundamentally altered when it is removed from its environment, resulting in a different mechanical behavior[1].

Philosophically, the two approaches are fundamentally different. In one case the behavior of each of the components of the liver is tested separately by dissecting it into different parts. On the other hand we have a more holistic approach where the deformation of the liver, treated as one entity, if recorded directly.

6.2 Outline of the Model

Due to the above mentioned anatomic and biomechanical properties, a heterogeneous model is chosen for modeling the mechanical behavior of the liver. In the sequel, we show how we have modeled the liver using two main components: a 2D component for modeling the Capsule of Glisson and a 3D one for modeling the Parenchyma. Each of these models include a geometrical and a physical component.

[1]personal communication between F. Carter and M. Thiriet

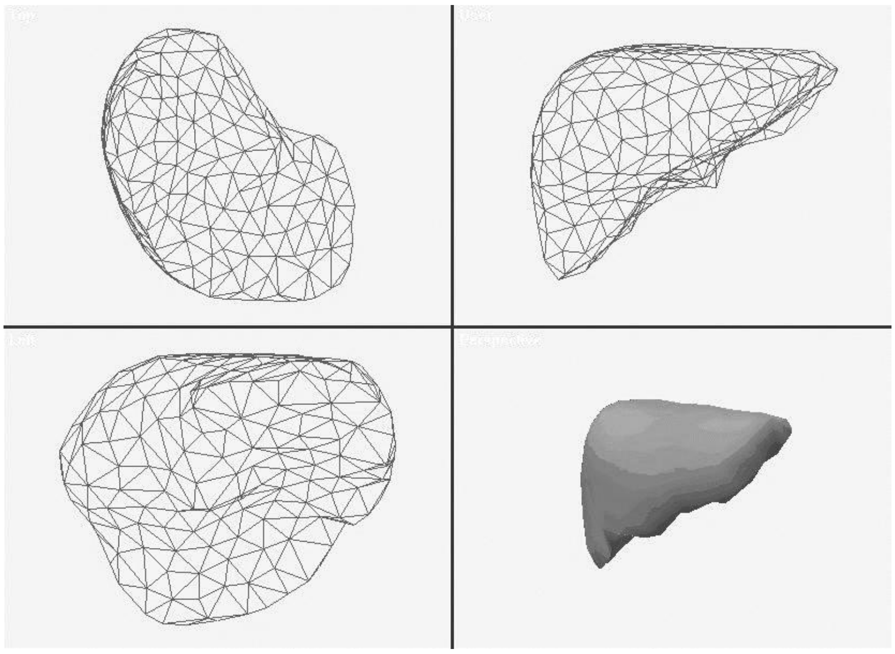


Figure 6.2: *The geometry of surface of the liver, given by a set points and triangles.*

6.2.1 Geometrical Component of the Model

The geometric component of the model is used for performing the display operations and for detecting interactions. This model is also used as a spatial frame for constructing the physical component.

We have chosen to make use of a 2D mesh consisting of triangles (see Figure 6.2), for representing the surface of the liver. This polygonal model has been constructed using the data from the Visible Human Project. It consists of 187 points, or vertices, and 370 polygons, or triangles.

Using the GHS3D [George and Borouchaki, 1998] software of INRIA the interior of the surface may be discretized into tetrahedrons. This requires the addition of points in the interior to a total of 299, to construct a set of 1151 tetrahedra. These geometrical primitives will form the basis for our physical description of the liver.

6.2.2 Physical Component of the Model

The physical component of the model is used to compute the deformations of the liver resulting from the application of a set of external forces. These forces are either applied by the operator using the virtual tool (controlled using a haptic interface), or generated

Vertices	187
Triangles	370
Particles	299
Tetrahedra	1151
Springs	1634
Particle mass	$5g$

Table 6.1: *Geometric and Physical properties of the model*

by the physical interactions with other virtual objects of the scene.

The physical model is constructed from the geometrical description of the organ, by associating point masses to the nodes of the previous meshes and by adding spring-damper connectors between appropriate local subsets of point masses (see [Boux de Casson and Laugier, 1999]). For simplicity we have decided to distribute the total weight of the liver (approx. $1.5kg$) equally over all of the particles (see Table 6.1). However, other approaches based on the volume of the tetrahedrons and density information could easily be envisaged.

In order to model the non-linear mechanical behavior of the liver, without increasing the algorithmic complexity of the approach, [Boux de Casson et al., 2000] chose to associate the following behavior to each spring:

$$\vec{F}_{spring} = \left(k_{non-lin}E^3 + k_{lin}E\right) + d\dot{E} \tag{6.1}$$

where E is the deformation of the spring. In this case we chose $E = \frac{L-L_0}{L_0}$, where L is the length of the spring and L_0 its rest length; k_{lin} and $k_{non-lin}$ are stiffness parameters, d is a viscous parameter, and $\dot{E}$ the relative velocity of the two particles.

The choice for this equation was directed by the fact that the strain-stress data shows linear behavior for small deformations (lower than 10%), while for larger strain the stress increases sharply. The sum of the linear and a cubic functions give a good approximation of this kind of behavior. An example of the characteristic shape of the function given by Equation 6.1 is shown in Figure 6.3. This non-linear relation makes the springs relatively incompressible and unstrechable.

In order to model the heterogeneousness of the liver, the connectors of the surface mesh and of the interior mesh are parameterized in such a way that they exhibit different mechanical characteristics:

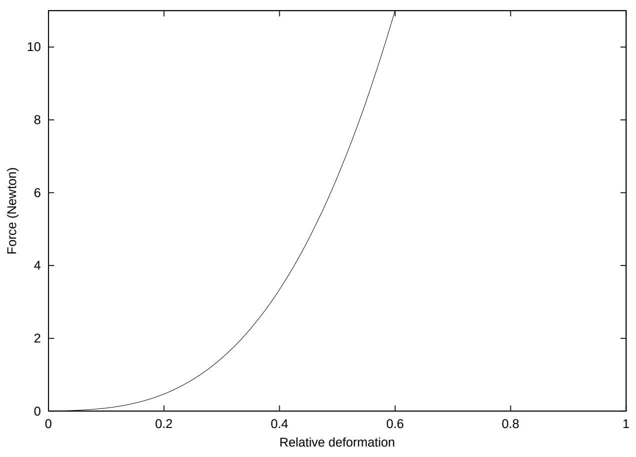


Figure 6.3: *Plot of the springs force, as a function of their relative deformation, with $k_{non-lin} = 50$ and $k_{lin} = 0.35$.*

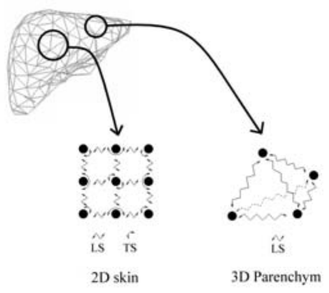


Figure 6.4: *The hybrid mesh approach. A 2D and a 3D spring-damper meshes are used to model respectively the Capsule of Glisson and the Parenchyma.*

- the *Capsule of Glisson* is modeled using high stiffness parameters ($k_{non-lin}$ and k_{lin} in Equation 6.1) and low viscous parameter, allowing us to obtain a rigid elastic behavior which tend to bring back the Capsule of Glisson to its initial shape when no force is applied on it.
- for the *Parenchyma*, the stiffness parameters are tuned to be small with respect to the viscous one, giving a over-damped response that avoids oscillations, which intuitively seem unrealistic in a surgical environment.

The combination of the two previous models (the elastic skin and the viscous volumetric internal material) gives us qualitatively and experimentally the required global behavior for the virtual liver (the behavior of a "sponge full of liquid covered by an

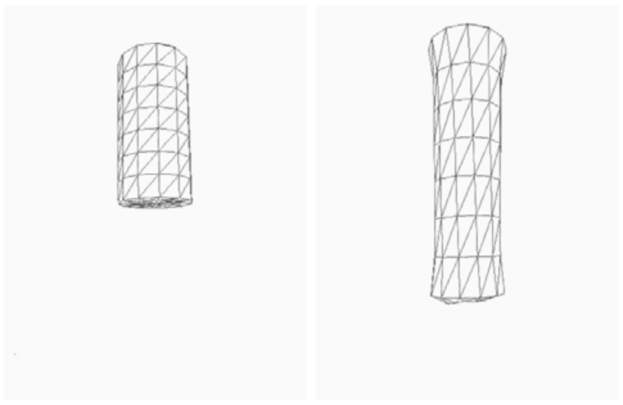


Figure 6.5: *Cylindrical model at rest (left) and submitted to a six newton force (right).*

elastic skin").

6.2.3 Validation of the deformation model

To validate that the non-linear mechanical behavior of each connector leads to a global non-linear behavior, [Boux de Casson, 2000] used a cylindrical model for virtual mechanical traction tests. The top of the cylindrical model (Figure 6.5) is fixed, and different forces are applied to all the bottom point-masses.

Figure 6.6 presents the results of the tests. We can see that the behavior of the global cylindrical model presents a similar shape to those of each of its connectors. So, the non-linear behavior of the connectors is faithfully reproduced on the global model.

6.3 Integration of the Dynamic Equations

Using first order explicit Euler integration allows 280 steps per second on an SGI Octane at 175 MHz. This represents the number of times the internal forces are calculated and the state is updated. Since our motto is real-time above all we set our timestep to $\frac{1}{280}s$ to synchronize the integration process with the wall clock time. Recall from Chapter 3 that the first order explicit euler integration of a linear dynamic equation is

$$y_1 = (1 + h\lambda)y_0 \tag{6.2}$$

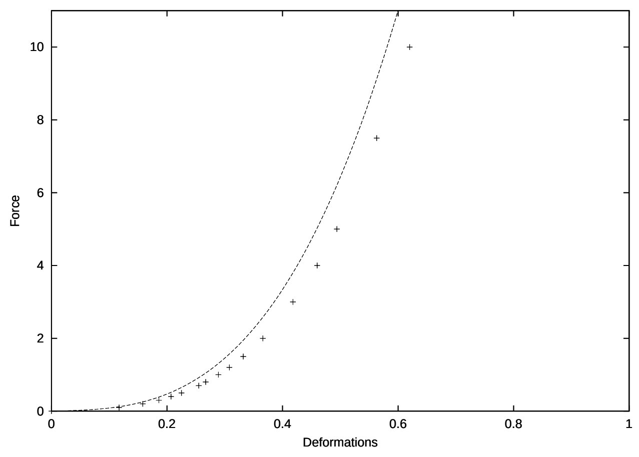


Figure 6.6: *The dots curve is the mechanical law of the connectors (Equation 6.1) and the cross were obtained by the virtual mechanical traction tests on the cylindrical model. Both curves presents the same non-linear shape.*

where h is the timestep, which here is equal to $\frac{1}{280}$. As long as the absolute value of $(1 + h\lambda)$ is strictly less than unity the solution is bounded and will not diverge. If we let $z = h\lambda$ the stability function is given as

$$R(z) = (1 + z) \tag{6.3}$$

For constant damping and stiffness coefficients, the z for each differential equation that governs the movement is given by Equation 3.21. Hence for $h = \frac{1}{280}$:

$$z = \lambda h = \left[-\frac{D}{2M} \pm \sqrt{\left(\frac{D}{2M}\right)^2 - \frac{K}{M}} \right] \frac{1}{280} \tag{6.4}$$

where the stiffness coefficient, K, for the equation is dependent on the stiffness parameters (k_{lin} and $k_{non-lin}$) of the springs and the topological structure of the mass-spring network. Similarly, the value of D depends on the viscous parameters (d) of the springs attached to the particle. As the stiffness, K, and damping, D, coefficients are expressed in terms of Newtons with respect to a change in displacement (in millimeters), the mass of the particle, M, is given in terms of mega-grams, i.e. $M = 5 \times 10^{-6}$.

The stability region is defined where $|R(z)| \leq 1$ (it is ploted in Figure 3.12). Of course this plot is only valid for the linear case, i.e. the values for stiffness and viscosity are constant. This is clearly not the case in our simulation since our models exhibits both material and geometrical nonlinearities. It is thus our aim to verify experimentally to which extent the linear stability conditions hold in the nonlinear case.

6.3.1 Experimental Results

In Table 6.2 we compute the stability function over a range of linear stiffnesses parameters, k_{lin}), with different damping coefficients, d. A set of particles are fixed and the only forces acting on the model are due to gravity.

To calculate the linear stability function the largest stiffness and damping coefficients (K and D) in the global matrices are used, as these will determine the stability for the whole system. This means that the stiffest equation in the system *will define the stability of the whole model.*

- $\mathbf{1 \times 10^{-4}}$ The stiffness is very small and the gravitational pull deforms the liver beyond recognition. If critical damping (CD) is used, the value for z is very close to the origin and therefore close to the boundary of the stability region. Even though the linear stability function is less than unity, the actual simulation diverges. I believe this is due to nonlinearities arising from the large deformation. If the system is over-damped (OD), z moves further to the center of the stability region and is also stable in practice. In case there is no damping (ND) both theory and practice indicate divergence.

- $\mathbf{1 \times 10^{-3}}$ Stiffer than the last case, the deformation is much smaller. If critically damped, z is still to the right side of the circle's center where the value of $|R(z)|$ is lowest. By over-damping z moves further to the left on the real axis until the limit of the stability region (SR) is reached. When underdamped theory predicts stability for the given case, however, the numerical simulation diverges in practice. Again I believe this is due to the nonlinearities due to large deformation.

- $\mathbf{1 \times 10^{-2}}$ The CD case still has the z to the right of the center of the SR. Thus there is still room for overdamping the system without loosing stability. On the other side only a slight underdamping will be stable. Since deformations are not very large the linear stability theory gives excellent predictions on the numerical divergence of the simulation.

- $\mathbf{5 \times 10^{-2}}$ The CD case is right in the middle of the unit circle. This gives leeway both for stable underdamping and overdamping (see Figure 6.7). Again numerical simulation behaves according to theory.

Case	Stiffness (k_{lin})	Damping (d)	$\|R(z)\|$	Simulation ($280Hz$)
ND	1×10^{-4}	0	1.01	unstable
CD	1×10^{-4}	1×10^{-5}	0.95	unstable
OD	1×10^{-4}	1×10^{-4}	0.43	stable
ND	1×10^{-3}	0	1.01	unstable
UD	1×10^{-3}	3×10^{-5}	0.92	unstable
CD	1×10^{-3}	5×10^{-5}	0.85	stable
OD	1×10^{-3}	1×10^{-4}	0.46	stable
OD	1×10^{-3}	4×10^{-4}	1.27	unstable
ND	1×10^{-2}	0	1.09	unstable
UD	1×10^{-2}	1×10^{-5}	1.07	unstable
UD	1×10^{-2}	5×10^{-5}	0.95	stable
CD	1×10^{-2}	1×10^{-4}	0.54	stable
OD	1×10^{-2}	2×10^{-4}	0.07	stable
OD	1×10^{-2}	3×10^{-4}	0.58	stable
OD	1×10^{-2}	4×10^{-4}	1.19	unstable
ND	5×10^{-2}	0	1.42	unstable
UD	5×10^{-2}	1×10^{-4}	1.20	unstable
UD	5×10^{-2}	2×10^{-4}	0.93	stable
CD	5×10^{-2}	3×10^{-4}	0.05	stable
OD	5×10^{-2}	4×10^{-4}	0.67	stable
OD	5×10^{-2}	5×10^{-4}	1.79	unstable
ND	1×10^{-1}	0	1.74	unstable
UD	1×10^{-1}	3×10^{-4}	1.15	unstable
CD	1×10^{-1}	5×10^{-4}	0.42	stable
OD	1×10^{-1}	7×10^{-4}	1.66	unstable
ND	2×10^{-1}	0	2.25	unstable
UD	2×10^{-1}	6×10^{-4}	1.28	unstable
CD	2×10^{-1}	7×10^{-4}	1.02	unstable
OD	2×10^{-1}	8×10^{-4}	2.35	unstable

Table 6.2: *Value of the stability function, $R(z)$, and experimental test for divergence for various stiffnesses in the case without damping (ND), under-damped (UD), critically damped (CD) and over-damped (OD).*

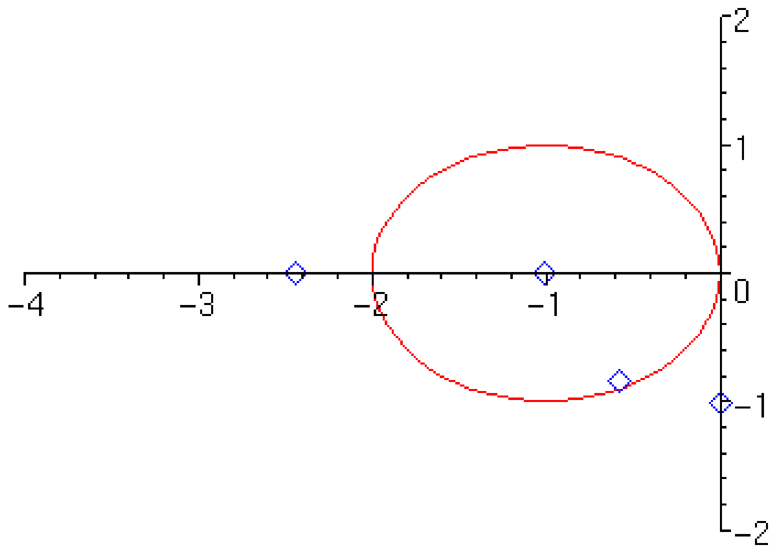


Figure 6.7: *Values of z for a stiffness of 0.05 ploted for the ND case (on imaginary axis), UD case (just inside the stability region), CD case (middle of circle) and OD case (on the real axis outside the stability domain).*

- $\mathbf{1 \times 10^{-1}}$ When critically damped z is now on the left side of the stability region. Hence UD and OD will result in a quick loss of stability (see Figure 6.8).
- $\mathbf{2 \times 10^{-1}}$ None of the values of z are inside the stability domain. However, they are closest to it in the CD case (see Figure 6.8).

Discussion Our experimental results are in excellent concordance with linear stability theory when deformations are reasonably bounded, allowing to estimate the stability of the simulation based on the parameters of the springs. We have shown that there is an intrinsic limit to the stiffness and damping of the springs. Further, the highest stiffness values only allow a stable integration when the system is critically damped (see Figure 6.8). This condition occurs when:

$$D = \sqrt{4KM} \tag{6.5}$$

In Figure 6.9 the interaction with the dynamic model of the liver is shown. The parameters of the springs are chosen to critically damp the system using the highest

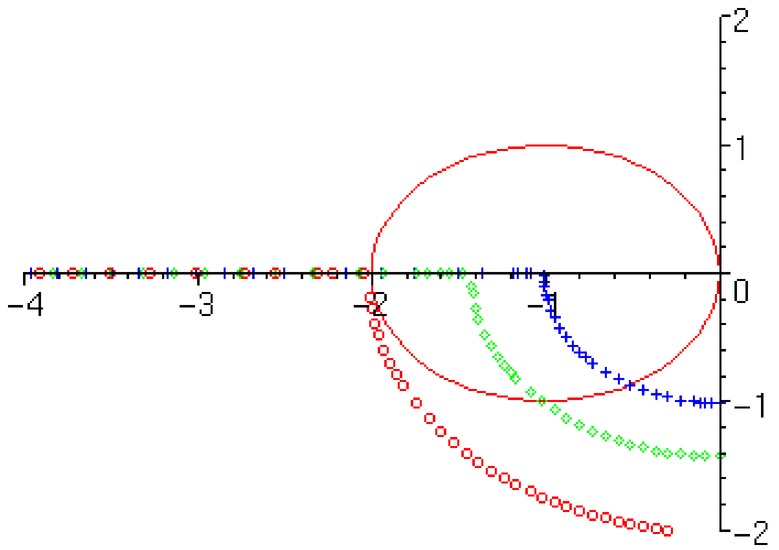


Figure 6.8: *Values of z for different stiffnesses ploted over a range of damping coefficients:* 2×10^{-1} *(circle),* 1×10^{-1} *(diamond) and* 5×10^{-2} *(cross).*

stiffness parameters that allow a stable simulation. However, these values are far softer than rheological tests [Carter et al., 2000] have approximated.

Conclusions As we have seen in this section, the feasibility of explicit integration as part of a real-time simulation, is limited by the stiffness of the differential equations. Of course, we could resort to implicit integration as in [d'Aulignac et al., 1999b]. However, as the system is over-damped and the stiffness high, the transient response of a dynamic analysis is very short. Hence, taking a large timestep with an implicit scheme often equates with the steady state response of the system or, in other words, the static equilibrium configuration.

6.4 Static resolution

Given we can neglect the dynamic phenomena in the simulation process we may resort to a static resolution. Hence, rather than finding the time-history of the deformation as is the case in the last section, we look directly for the static equilibrium position if it exists. Already [Cotin et al., 2000] have used *linear* static resolution for surgery simulation. Further, [Brown et al., 2001] have demonstrated that a *nonlinear* static

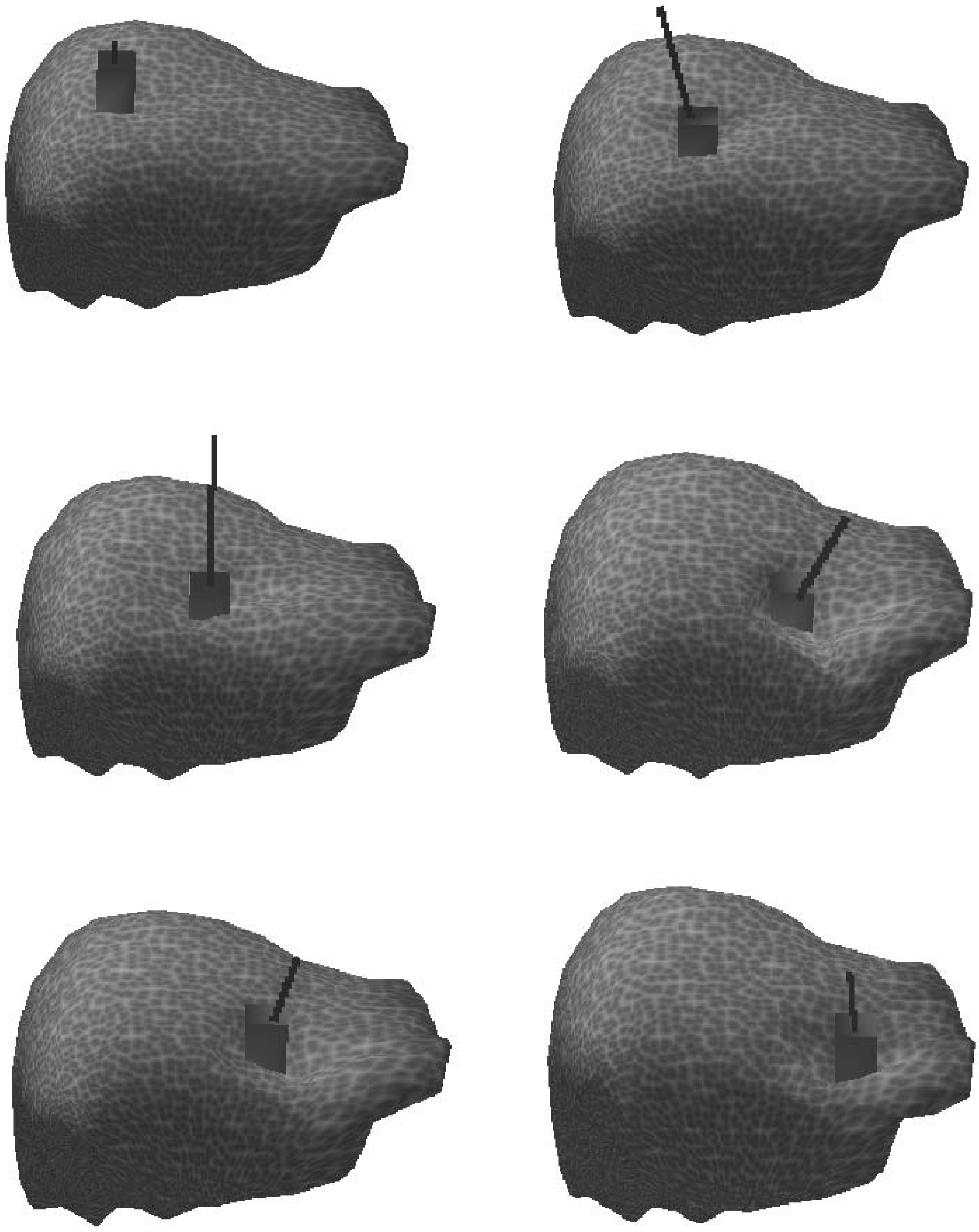


Figure 6.9: *Interactive manipulation of the liver using a* dynamic *resolution.*

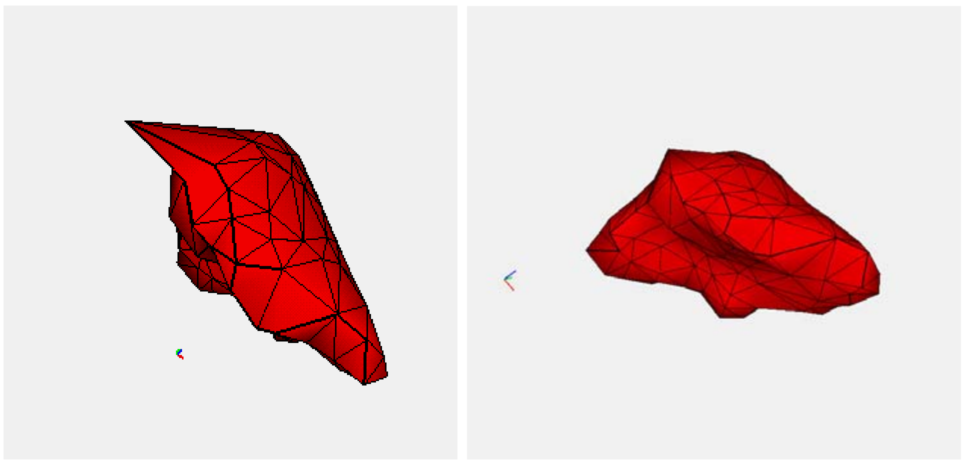


Figure 6.10: *Comparison of a linear (left) and a non-linear resolution for a given external force applied on one node.*

resolution can be computationally advantageous with respect to a dynamic integration process.

Linear vs. Non-Linear The first question that poses itself when searching for the static equilibrium of a deformable body is whether it exists and it is unique. Given the latter is true we should ask ourselves if we need to perform a linear or non-linear resolution (see Chapter 3). This choice should be based on the non-linearities present. As we have seen, these can be classified as material or geometrical. Since the elements that make up our model are of non-linear behavior it follows that there will be material non-linearities present. However, and probably more importantly, the liver may undergo large rotation during a procedure. This introduces geometrical non-linearities due to the fact that the stiffness matrix changes with increasing rotation. Figure 6.10 shows the difference in the deformation of the liver when performing a linear and non-linear resolution. The force acting on the frontal lobe lifts the liver as well as inducing a slight rotation. For the linear resolution this produces an unrealistic increase in volume. We conclude from this that a non-linear resolution is necessary in spite of its superior computational expense.

6.4.1 Algorithms

For the non-linear resolution we shall compare the full and modified Newton-Raphson algorithm. Both algorithms are based on a successive resolution of a linear system. The size of this system depends on the number of degrees of freedom (DOF) of our model. Since the mesh is built of 299 points, each one having three DOF (i.e. x, y, z), this makes a total of 897 DOF. Direct solution of system of this size by matrix inversion is slow as well as memory-consuming since the inverse will have to be stored (N.B. even if the original stiffness matrix is sparse its inverse may not be). Hence we use an iterative solution of the system by successive over-relaxation (SOR).

The difference in full and modified Newton-Raphson (NR) resolution resides in the recalculation of the Jacobian, which in this case is the stiffness matrix K. We have calculated this matrix *analytically* by assembling the contributions of each elementary matrix into the global matrix. While the Newton-Raphson method recalculates this matrix at each iteration, the modified Newton-Raphson only evaluates it once at the beginning. Of course, this implies that a modified NR iteration will be less expensive to calculate than a full NR iteration. However, this does not necessarily imply that the modified NR method is faster on a global scale, as we shall see later.

Algorithm 2 Newton-Raphson resolution

```
while 1 do
  evaluate stiffness matrix $K$
  calculate residual forces $f_{res}$
  for $i = 0$ to $n$ do
    SOR iteration for $Ku = f_{res}$
  end for
end while
```

Algorithm 3 Modified Newton-Raphson resolution

```
evaluate stiffness matrix $K$
while 1 do
  calculate residual forces $f_{res}$
  for $i = 0$ to $n$ do
    SOR iteration for $Ku = f_{res}$
  end for
end while
```

6.4.2 Experimental Results

To test the two methods we impose a force on one of the points on the surface (see Figure 6.11). We then record the time it takes for each of the methods to converge to an equilibrium. We have assumed convergence when the squared residual force drops below 1. Each method has been tested with a different number of SOR iteration for the

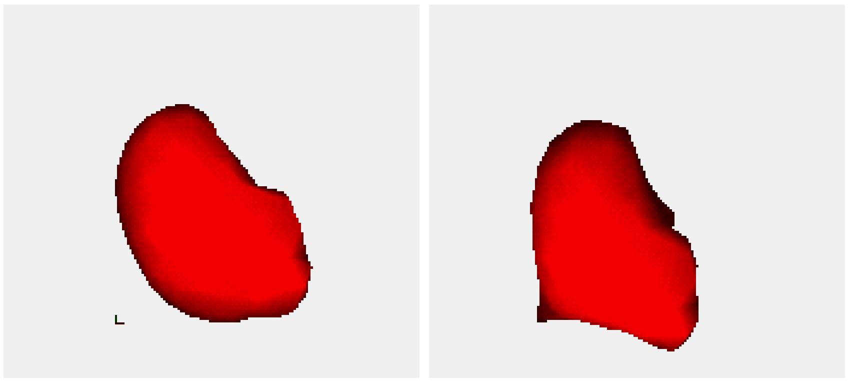


Figure 6.11: *Deformation of the liver when pulling on a point with a force vector equal to* $(-1, -1, -1)$.

solution of the linear system. The last solution of the linear system is always used as the starting point for the next resolution.

Modified Newton Raphson resolution In Table 6.3 we can see how the number of modified NR iterations it takes to achieve convergence varies with increasing number of SOR iterations. Taking more SOR iterations will increase the accuracy of the resolution of the linear system. Since the linear system is solved more accurately, the number of NR step necessary for convergence may also be reduced (compare row 1 and 2). However, beyond three SOR iterations no further reduction in NR iterations is recorded. Figure 6.12 shows how squared error does not drop faster as the linear system is solved more accurately. Since the modified NR method does not update the Jacobian the approximation of the non-linear system is not accurate. Therefore we come to the conclusion that the convergence of the modified NR method is limited by the rough approximation of the Jacobian. Solving the linear system with high accuracy is thus not useful.

Newton Raphson resolution In Table 6.4 we repeat the experiment but recalculating the Jacobian at each NR step. Since each NR step is more expensive, the only way to outperform the modified NR method is by taking much less iterations to converge. This is exactly what happens when we solve the linear system using 7 SOR iterations. We believe that in this case it pays off to solve the system more accurately since we have a much better approximation of the Jacobian. Hence we can see in Figure 6.13 how the descent of the squared error of the residual drops much faster than with the modified

Number of SOR iterations	Time in seconds	Newton iterations
1	2.46	323
2	2.93	267
3	3.71	264
4	4.55	264
5	5.39	264
6	6.24	264
7	7.14	264
8	8.01	264

Table 6.3: *Time and number of* **modified Newton-Raphson** *iterations taken for convergence (i.e. residual smaller than* 1*) in function of the SOR iteration taken. The Jacobian is constant.*

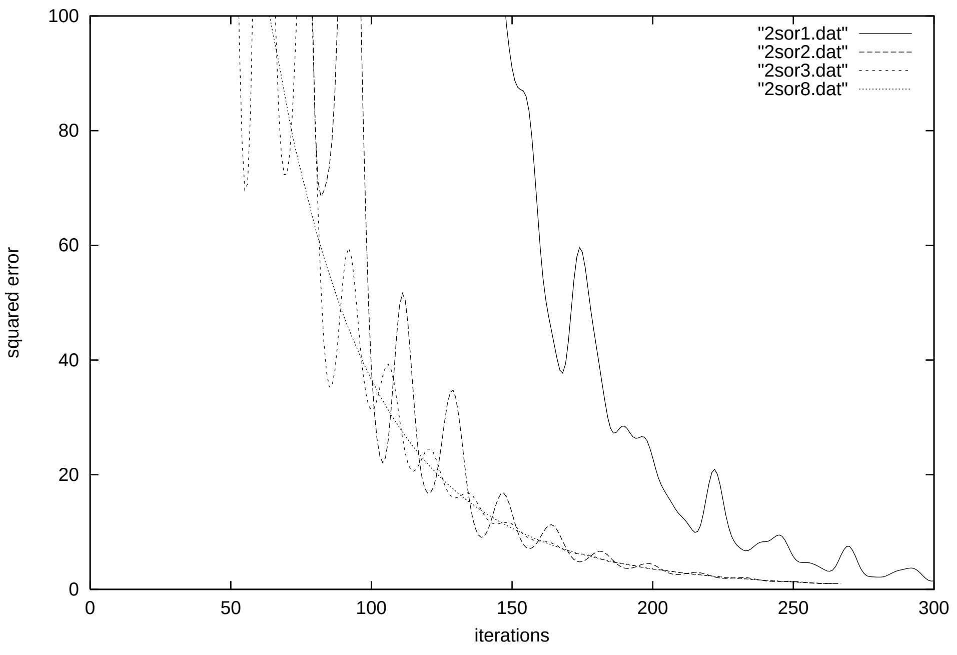


Figure 6.12: *Squared residual error with respect to the number of* **modified Newton-Raphson** *iterations taken with different number of SOR iterations.*

Number of SOR iterations	Time in seconds	Newton iterations
1	non	converged
2	6.94	330
3	4.15	171
4	3.38	123
5	2.80	91
6	2.72	80
7	2.38	64
8	2.58	64

Table 6.4: *Time and number of* **Newton-Raphson** *iterations taken for convergence (i.e. residual smaller than* 1*) in function of the SOR iteration taken. The Jacobian is recalculated for each Newton iteration.*

NR method. Thus using an updated Jacobian is globally faster than not recalculating for this particular case. We may, however, generalize that for all cases where the Jacobian changes much (i.e. large deformation or rotation) it will be computationally advantageous to update the Jacobian.

Discussion In the experimental result shown we have used repeated SOR resolution of a linear system of equations in order to solve the non-linear one. It can be seen that in some cases it is best not to spend too much effort and computational power on solving the linear system with great accuracy, as in the next Newton iteration we will have to solve it all over again. We use this hypothesis to explain why our use of a conjugate gradients (CG) method as a linear solver, failed to give better results than SOR, as the CG method will give more accurate results for the same number of iteration, but at a higher computational cost.

Conclusions In this section we have seen that static resolution can be an interesting alternative to an dynamic analysis. It can yield a static equilibrium configuration (see Figure 6.14) with less computational overheads than an integration process. Further, an illusion of dynamic behavior can easily be created by displaying intermediate stages of the NR resolution, in what we could call *quasi-dynamic* analysis. For example, if a maximum of 200 NR iterations per second can be calculated for a particular resolution, we may choose to graphically display the configuration every 20 iterations. This will yield a graphical output at a rate of 10 frames per second (see algorithm).

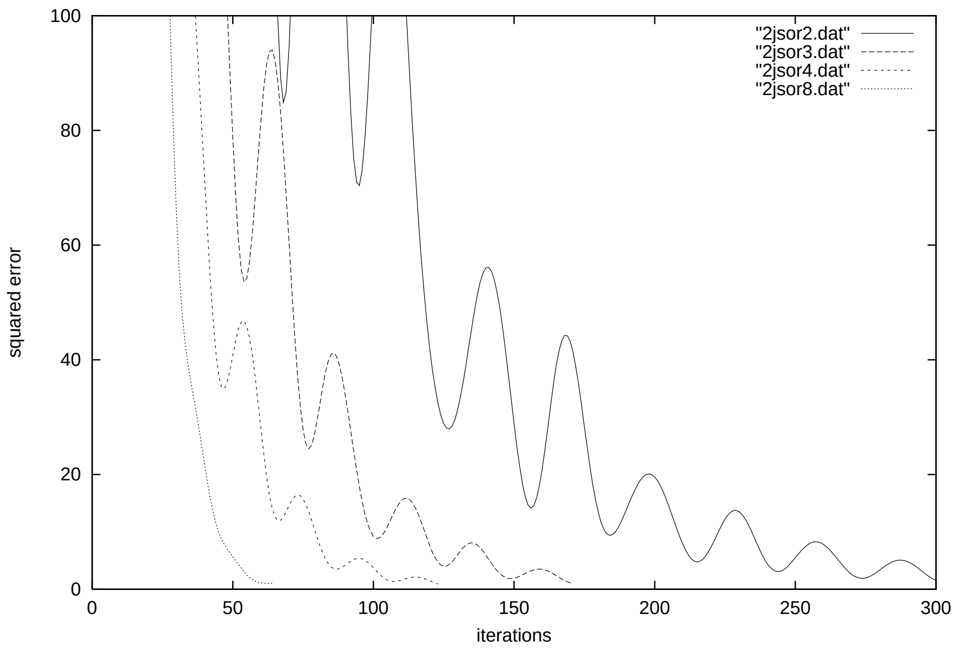


Figure 6.13: *Squared residual error with respect to the number of* **Newton-Raphson** *iterations taken with different number of SOR iterations.*

Algorithm 4 Quasi-dynamic graphical display

```
$t_{last} = t$
while 1 do
    perform NR iteration
    if $t > t_{last} + 0.1s$ then
        display configuration
        $t_{last} = t$
    end if
end while
```

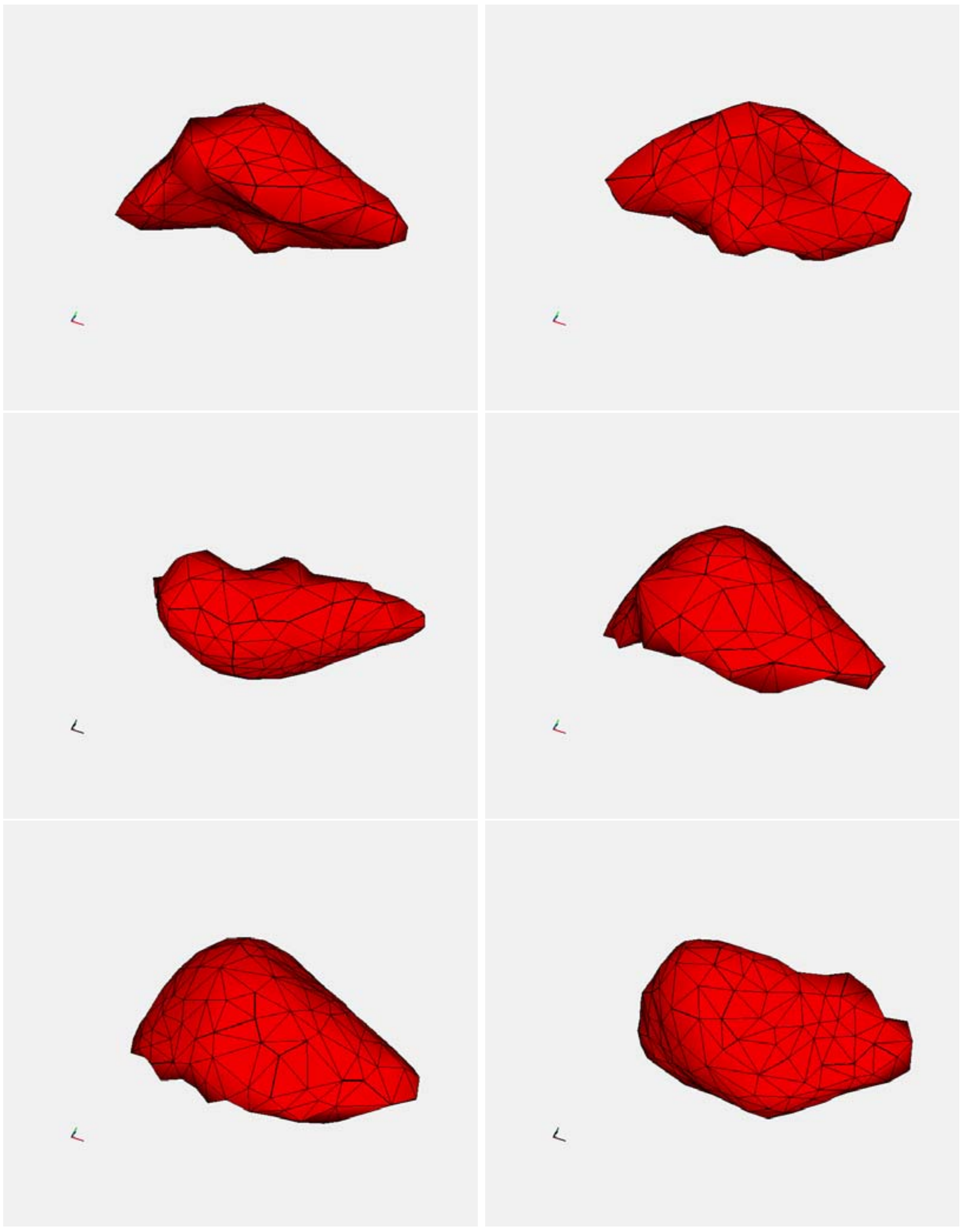


Figure 6.14: *Several static equilibrium positions due to externally applied forces, with the configuration of the liver in its undeformed state on the bottom right picture.*

6.5 Interaction

6.5.1 Collision Detection and Response

The user may interact with the deformable model of the liver using a virtual tool. Hence the collisions between this tool and the liver must be detected. We model the tool as a rigid parallelepiped as an approximation to the laparoscopic instruments used. The the tool is therefore a convex object. The liver, however, is concave. Thus we must use an algorithm that will detect a collision between a simple convex object and a concave surface consisting of a large number of polygons. There are several methods that could solve such a problem, but we have chosen to use the approach by [Lombardo et al., 1999] because of its computational efficiency due to hardware acceleration (see Section 4.2.2). A bounding box of the same size as the tool is created and the liver is graphically displayed. Using the OpenGL picking mode we will know the triangles of the liver inside the bounding box or tool.

Having detected the triangles on the surface of the liver that collided with the tool, we must calculate an appropriate response. Clearly we do not want them to be inside the tool, but on the surface where the contact takes place. We make the assumption that the contact is made on the front end of the tool; thus we may easily calculate the distance from the tip of the tool to the offending triangle. Knowing the magnitude and direction of this distance we use the non-linear penalty model presented in Section 4.3.3.

More sophisticated approaches are, however, possible as presented in [Picinbono, 2001] where the triangles are projected outside the tool in the direction of their normal vector. This allows to also use the side of the instrument to interact with the object. Unfortunately, forces acting on the sides of the tool cannot be used by our force-feedback device, as it is unable to supply torque.

6.5.2 Haptic Interaction

Figure 6.15 shows how the PHANToM force-feedback device is used to interact with the model of the liver. Position and orientation of the stylus in the real world determine the location of the virtual probe. The simulation sends back the forces due to interaction with the deformable model.

As in the last chapter a local approximation of the contact by the method described in Section 4.4 is used.

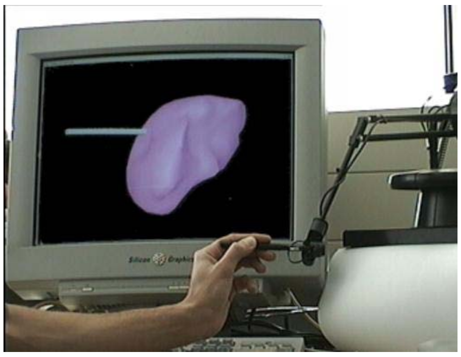


Figure 6.15: *The liver model, being manipulated using the force feedback device.*

6.6 Changing the topology

Problem Statement Normally surgeons do not operate just to "have a look around". In the case of the liver, a classical operation is to remove the part of the liver where cancerous cells are present. This, of course, requires to *cut* parts of the organ. Hence, to model a simulation of such a procedure, this phenomenon must be modeled.

Approach An algorithm to tear the model has been implemented, as it is presented in [Boux de Casson and Laugier, 2000]. The global idea is to separate the elements of simulation (i.e. the tetrahedra) when they are stretched above a given threshold. This approach gives better results than when elements are removed, because the discretisation is often coarse in real-time models. Thus when big elements are removed, one can see matter disappearing. Subdivision methods [Bielser et al., 1999] give more accurate results, but are not compatible with interactive constraint, because the number of elements to be simulated increases at each topology change, and the real-time constraint cannot always be assured. Nevertheless, the limitation of our approach is that the topology changes are constrained by the initial topology of the model.

Experimental results Our current cutting algorithm works only on membrane models. It is possible to cut a dynamic model, using the PHANToM device to control the position and orientation of a virtual tool. The model is cut exactly on the tool trajectory (Figure 6.16).

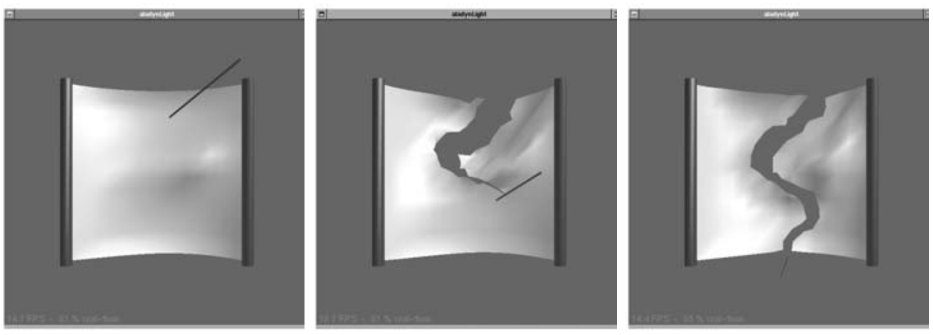


Figure 6.16: *Interactive cut of a membrane model.*

6.7 Conclusion and Perspectives

In this chapter we have described the implementation of a rudimentary simulator of the human liver.

We have chosen a mass-spring network to model the deformation of the liver, based on the computational advantages with respect to other methods (even if the latter might be more accurate). As the structure of the liver is non-homogeneous, we have chosen different behaviors for the springs based on rheological measurements that have been made. Based on this information we have used a non-linear function to calculate the elastic force of a spring due to deformation, coupled to a damping value to avoid oscillations during a dynamic analysis of the deformation. It can be verified experimentally that this behavior, local to the spring, remains valid on a global scale.

A dynamic resolution of the deformation can be performed using explicit integration methods. Since we aim to guarantee real-time performance, we choose to use a fixed timestep that guarantees synchronicity with wall clock time. However, there are limitations for the stiffness and damping values for a given timestep. We establish a formal basis that links the stiffness and damping values to linear stability theory. From this we conclude that the maximal stiffness, allowing a stable explicit integration, can only be archived in a critically damped system. Nevertheless the stiffness used in our dynamic analysis remain much inferior to those recorded in rheological tests.

We could have chosen to use implicit integration to circumvent the stiffness problems. However, we estimate that the transient response of a dynamic analysis is sufficiently small to justify a direct resolution for the steady-state, or static equilibrium. Further, we believe that the boundary conditions imposed on the liver in the abdominal cavity are sufficient to enforce the existence of a unique equilibrium configuration, necessary to any static resolution technique. However, the non-linearities (especially due to rotation of the organ during an operation) require a suitable algorithmic approach. Hence we have two different Newton-Raphson resolutions based on the iterative resolution of a linear system using SOR iteration. By graphically displaying the configuration at intermediate Newton iterations we create the illusion of a *quasi-dynamic* behavior of the liver at a lower computational complexity than a dynamic process.

Lastly, we have discussed the possible implementation of a cutting algorithm in our simulator. This is of paramount importance in a laparoscopic operation, whence the need to model this phenomenon. Preliminary result have been given on a suitable cutting algorithm, however, a model of the vascular network remains to be implemented, which is an essential building-block towards a workable prototype of a surgical simulation.

Chapter 7

Conclusions

7.1 Summary of findings

Deformation We have examined different approaches towards modeling the deformation of an object with complex geometry. In particular we have compared the use of mass-spring networks and tetrahedral finite elements. We have found that the behavior of mass-spring networks is intrinsically dependent on the topological configuration of the springs. This problem arises from the fact that we try to model a three-dimensional object by one-dimensional elements. Volumetric finite elements, on the other hand, do not suffer from this limitation. Further, they may easily model phenomena such as incompressibility. However, they also demands more computation per element. We have found experimentally that an object modeled with tetrahedral finite elements is about four times slower to compute the internal forces than the mass-spring network for the same object.

Resolution When solving for the deformation of an object we may choose between a static and a dynamic analysis. A static resolution can be envisaged when a unique equilibrium configuration for a given external constraint exists. For small deformations a linear resolution will suffice, however, for larger deformations or rotations a non-linear resolution must be performed. Alternatively, if we are interested in the change of the deformation with respect to time or no equilibrium position exists, a dynamic analysis is necessary. To solve the differential equations arising from this resolution either explicit of implicit integration can be used. The stability of explicit methods is directly linked to the stiffness and damping coefficients of the physical system, imposing a limit on the time-step that can be taken. Implicit methods do not suffer from this limitation but may demand the solution of a non-linear system.

Interaction The collision detection between complex polygonal models is an expensive and time-consuming task. However, penalty based methods must be provided with collision information at a fast rate for realistic collision response. Further, if the collision forces are sent to a force-feedback device at low frequency unacceptable haptic sensation will result. To circumvent this problem a local approximation of the contact is introduced. Since the geometry of the local model is much simpler, collision detection and response can be performed at a much higher frequency, resulting in a improved haptic display.

Echographic Simulator We have measured the forces due to the deformation of the thigh using a force sensor. Based on this data a mass-spring model using non-linear connectors was constructed. Further, the parameters of the springs have been identified to best match the measurements. Alternatively, we have examined the feasibility of using a neural network to approximate the the static deformation, based directly on the measurement data, with promising results. For a dynamic analysis conclude that, for this case, it is beneficial to use implicit integration due to the stiffness of the springs. This allows a stable integration with collision detection and response at a frequency of $100Hz$. Thus, to provide higher rates for the haptic device, a local approximation of the contact has been implemented. Experimental results show that better force-feedback can be achieved using this approach. Lastly, the physical model has been integrated with a echographic image generator resulting in a first, rudimentary prototype of an echographic simulator of the human thigh.

Model of the Liver We present a model of the liver based on the rheological data available in the literature. However, explicit integration techniques have proven to be unstable for the reported elastic and viscous properties of the liver. Hence, we have found that a non-linear static resolution can provide an interesting alternative, given that the transient response to a deformation of the liver is negligible.

7.2 Analysis

The choice of the physical model used must be based on the context of the simulation. *Do we require speed or accuracy?* Ideally, of course, we would want both. However, if this is impossible to achieve because of the computational overheads, we must carefully examine the tradeoffs between accuracy and speed. Hence, in some cases we may decide to accept the lower physical realism of mass-spring networks due to the lower computational complexity with respect to finite elements. On the other hand, if accuracy is essential, we may choose to use finite elements and pay the price (as, for example, in [Payan et al., 2000]). In any case, this decision must always be based on the objectives of the application.

Similarly, the need for a static or dynamic resolution depends on the procedure we aim at simulating. It may be that a dynamic analysis is of paramount importance to the application [Hilde et al., 2001]. If this is the case, we have no choice but to integrate the system with respect to time (here too the accuracy vs. speed dilemma resurges: an implicit integration may allow us to take lager timesteps allowing a faster simulation, but we will also loose in accuracy). However, if the deformations are sufficiently slow, it may be possible to perform a static analysis for a reduced computational complexity. Again, the answer to this question should always be based on the requirements of the procedure.

A further expensive task in a simulation is the collision detection. As the computation of the inter-penetration distance on the global model is too slow for haptic rendering, we propose to use a simpler local model. Here we have yet another example of trading in accuracy for speed. The simulation of the contact on the rigid contraint plane allows extremely fast computation on the collision forces, however, the deformation is ignored between two updates of the physical model (and therefore less accurate). Nevertheless, it may be possible to find compromises between speed and accuracy. For example, we could extend the accuracy of the local model by allowing a deformation of the contraint surface based on the elasticity of the material adjacent to it.

In conclusion, we have implemented and compared different physical models, resolution techniques, and interaction approaches allowing the development of prototype simulations for relevant medical procedures. It is here that one of our main contributions lies: the analysis of these methods in terms of accuracy and computational complexity are essential towards to the building of virtual medical simulators. Hence, we hope that further work involving feedback from medical staff and clinical test will permit to extend this work in order to eventually provide an efficient training tools for practitioners.

7.3 Perspectives

Algorithms Clearly there are still possibilities for improving the algorithms used for the resolution of the linear and non-linear systems. In particular multi-grid solver could offer an interesting alternative to the successive over-relaxation or conjugate gradients solvers used in this work.

Parallelism Many of the procedures for simulating physical systems can be parallelized. In particular the linear solvers and the calculation of the internal forces would be suitable candidates. However, great care should be taken when distributing the task on a multi-processor platform in order to minimize the dependencies on the access to data between processors as his will diminish the speedup that can be achieved [1]. Hence, for best results a separation based on the geometry of the object analyzed should be performed.

Hardware Acceleration It is my personal belief that the next great increase in computational power for solving physical systems will come through the introduction purpose-built hardware [Bishop and Kelliher, 2001]. Much in the same way as 3D graphic cards have revolutionized the possibilities for displaying complex scenes consisting of millions of polygons, I expect that physical simulation cards will enable us to simulate objects composed of millions of elements in real-time on computer architectures that do *not* cost millions of francs.

[1]LOGIC, n. The art of thinking and reasoning in strict accordance with the limitations and incapacities of the human misunderstanding. The basic of logic is the syllogism, consisting of a major and a minor premise and a conclusion – thus: *Major Premise*: Sixty men can do a piece of work sixty times as quickly as one man. *Minor Premise*: One man can dig a posthole in sixty seconds; therefore – *Conclusion*: Sixty men can dig a posthole in one second. This may be called the syllogism arithmetical, in which, by combining logic and mathematics, we obtain a double certainty and are twice blessed. *Ambrose Bierce, The Devil's Dictionary*

List of Figures

Bibliography

[Ackerman, 1991] Ackerman, M. (1991). Viewpoint: The visible human project. *Journal Biocommunication 18(2):14.* (cited on page 3)

[Astley and Hayward, 1998] Astley, O. and Hayward, V. (1998). Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 989–994, Leuven (BE). (cited on page 73, 101)

[Balaniuk, 1999] Balaniuk, R. (1999). Using fast local modeling to buffer haptic data. In *Proc. of the Fourth Phantom User Group Workshop - PUG99*, Boston (US). (cited on page 73, 75, 78)

[Baraff, 1992] Baraff, D. (1992). *Dynamic Simulation of Non-Penetrating Rigid bodies.* PhD thesis, Cornell University. (cited on page 68)

[Baraff, 1996] Baraff, D. (1996). Linear-time dynamics using lagrange multipliers. In *SIGGRAPH 96 Conference Proceedings*, Computer Graphics Proceedings, Annual Conference Series, pages 137–146. ACM SIGGRAPH, Addison Wesley. ISBN 0-201-94800-1. (cited on page 28)

[Baraff and Witkin, 1992] Baraff, D. and Witkin, A. (1992). Dynamic simulation of non-penetrationg flexible bodies. *Computer Graphics*, 26(2). Proc. of Siggraph'92. (cited on page 66)

[Baraff and Witkin, 1998] Baraff, D. and Witkin, A. (1998). Large steps in cloth simulation. In Cohen, M., editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 43–54. ACM SIGGRAPH, Addison Wesley. (cited on page 50)

[Barrett et al., 1994] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition.* SIAM, Philadelphia, PA. (cited on page 55)

[Basdogan, 2001] Basdogan, C. (2001). Real-time simulation of dynamically deformable finite element models using modal analysis and spectral lanczos decomposition methods. In *Medicine Meets Virtual Reality*, pages 38–44, Amsterdam. IOS Press. (cited on page 51)

[Basdogan et al., 1996] Basdogan, C., Loan, P., Rosen, J. M., and Delp, S. (1996). An interactive model of the human thigh for simulating surgical procedures in virtual environments. In *Advances in Bioengineering ASME.* (cited on page 84)

[Bathe, 1982] Bathe, K.-J. (1982). *Finite Element Procedures.* Prentice Hall, Inc. (cited on page 13)

[Bergen, 1998] Bergen, G. V. D. (1998). Efficient collision detection of complex deformable models using aabb-trees. Technical report, Department of Mathematics and Computer Science, Eindhoven University of Technology. (cited on page 67)

[Bielser et al., 1999] Bielser, D., Maiwald, V. A., and Gross, M. H. (1999). Interactive cuts through 3-dimentional soft tissue. In *EUROGRAPHICS'99*, volume 18, pages C–31–C38. (cited on page 128)

[Bishop and Kelliher, 2001] Bishop, B. and Kelliher, T. (2001). Hardware acceleration for physical modeling of deformable objects. In *Technical Sketches at SIGGRAPH 2001 (Los Angeles, California)*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM Press. (cited on page xxviii, 135)

[Bonomi and Lutton, 1988] Bonomi, E. and Lutton, J.-L. (1988). Le recuit simulé. *Pour la science*, pages 68–77. (cited on page 88)

[Boux de Casson, 2000] Boux de Casson, F. (2000). *Simulation dynamique de corps biologiques et changements de topologie interactifs.* Thèse de doctorat, Université de Savoie, Chambéry (FR). (cited on page 113)

[Boux de Casson et al., 2000] Boux de Casson, F., d'Aulignac, D., and Laugier, C. (2000). An interactive model of the liver. In *Proc. of the Int. Symp. on Experimental Robotics*, Honolulu, HI (US). (cited on page 19, 111)

[Boux de Casson and Laugier, 1999] Boux de Casson, F. and Laugier, C. (1999). Modelling the dynamics of a human liver for a minimally invasive simulator. In *Proc. of the Int. Conf. on Medical Image Computer-Assisted Intervention*, Cambridge (GB). (cited on page 111)

[Boux de Casson and Laugier, 2000] Boux de Casson, F. and Laugier, C. (2000). Simulating 2D tearing phenomena for interactive medical surgery simulators. In *Proc. of Computer Animation*, Philadelphia, PA (US). (cited on page 128)

[Bro-Nielsen and Cotin, 1996] Bro-Nielsen, M. and Cotin, S. (1996). Real-time volumetric deformable models for surgery simulation using finite elements and condensation. In *Proceedings of Eurographics*, volume 15, pages 57–66. (cited on page 33, 84)

[Brown et al., 2001] Brown, J., Sorkin, S., Bruyns, C., Latombe, J.-C., Montgomery, K., and Stephanides, M. (2001). Real-time simulation of deformable objects: Tools and applications. In *Computer Animation*, Seoul, Korea. (cited on page 118)

[Cameron, 1997] Cameron, S. (1997). Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In *IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 3112–3117. (cited on page 63)

[Carter et al., 2000] Carter, F., Frank, T., Davies, P., McLean, D., and Cushieri, A. (2000). Biomechanical testing of intra-abdominal soft tissues. In *Medical Image Analysis.* (cited on page 109, 118)

[Cavusoglu and Tendick, 2000] Cavusoglu, M. and Tendick, F. (2000). Multirate simulation for high fidelity haptic interaction with deformable objects in virtual environments. In *IEEE International Conference on Robotics and Automation, ICRA 2000*, San Francisco. (cited on page 73, 101)

[Cook et al., 1989] Cook, R., Malkus, D., and Plesha, M. (1989). *Concepts and Applications of Finite Element Analysis*. John Wiley and Sons, New York. (cited on page 24)

[Costa Ferreira and Balaniuk, 2001] Costa Ferreira, I. and Balaniuk, R. (2001). Static solution for real time deformable objects with fluid inside. *ERCIM News*, 44:44–45. (cited on page 29)

[Cotin, 1997] Cotin, S. (1997). *Modèles anatomiques déformables en temps-réel*. PhD thesis, Epidaure-Sophia Antipolis. (cited on page xxvii, 43)

[Cotin et al., 1999] Cotin, S., Delingette, H., and Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions On Visualization and Computer Graphics*, 5(1):62–73. (cited on page 33, 57)

[Cotin et al., 2000] Cotin, S., Delingette, H., and Ayache, N. (2000). A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452. (cited on page 23, 118)

[Cover et al., 1993] Cover, S., Ezquerra, N., O'Brien, J., Rowe, R., Gadacz, T., and Palm, E. (1993). Interactively deformable models for surgery simulation. In *Proceedings of ICRA*, pages 68–75. (cited on page 62)

[Craig, 1989] Craig, J. J. (1989). *Introduction to Robotics, Mechanics and Control*. Addison-Wesley, second edition. (cited on page 68)

[Dan, 1999] Dan, D. (1999). *Caracterisation mechanique du foie humain en situation de choc*. PhD thesis, Universite Paris 7 - Denis Diderot. in french. (cited on page 109)

[d'Aulignac and Balaniuk, 1999] d'Aulignac, D. and Balaniuk, R. (1999). Providing reliable force-feedback for a virtual, echographic exam of the human thigh. In *In Proc. of the PHANToM Users Group Workshop*, Boston, MA (US). (cited on page 97)

[d'Aulignac et al., 2000] d'Aulignac, D., Balaniuk, R., and Laugier, C. (2000). A haptic interface for a virtual exam of the human thigh. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 3, pages 2452–2457, San Francisco, CA (US). (cited on page 99)

[d'Aulignac et al., 1999a] d'Aulignac, D., Cavusoglu, M. C., and Laugier, C. (1999a). Modelling the dynamics of a human thigh for a realistic echographic simulator with force feedback. In *Proc. of the Int. Conf. on Medical Image Computer-Assisted Intervention*, Cambridge (UK). (cited on page 16, 87)

[d'Aulignac et al., 1999b] d'Aulignac, D., Laugier, C., and Cavusoglu, M. C. (1999b). Towards a realistic echographic simulator with force feedback. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 727–732, Kyongju (KR). (cited on page 92, 118) [d'Aulignac et al., 1997] d'Aulignac, D., Moschovinos, A., and Lucas, S. (1997). Virtual table tennis and the design of neural network players. In *Int'l Conf. on Artificial Neural Networks and Genetic Algorithms.* (cited on page 57)

[Dawson and Kaufman, 1998] Dawson, S. and Kaufman, J. (1998). The imperative for medical simulation. *Proceedinds of the IEEE*, 86(3):479–483. (cited on page 2)

[Debunne, 2000] Debunne, G. (2000). *Animation multirésolution d'objets déformables en temps réel.* PhD thesis, Institut National Polytechnique, Grenoble. (cited on page 45)

[Debunne et al., 1999] Debunne, G., Desbrun, M., Barr, A., and Cani, M.-P. (1999). Interactive multiresolution animation of deformable models. In *10th Eurographics Workshop on Computer Animation and Simulation, Milano, Italy.* (cited on page 20)

[Debunne et al., 2001] Debunne, G., Desbrun, M., Barr, A., and Cani, M.-P. (2001). Dynamic real-time deformations using space & time adaptive sampling. In *Proc. of Computer Graphics.* (cited on page 25, 51)

[Deguet et al., 1998] Deguet, A., Joukhadar, A., and Laugier, C. (1998). A collision model for deformable bodies. In *IEEE Int. Conf. on Intelligent Robots and Systems.* (cited on page 72)

[Delingette, 1998] Delingette, H. (1998). Towards realistic soft-tissue modeling in medical simulation. *Proceedinds of the IEEE*, 86(3):512–523. (cited on page 3)

[Delingette et al., 1999] Delingette, H., Cotin, S., and Ayache, N. (1999). A hybrid elastic model allowing real-time cutting, deformations and force-feedback for surgery training and simulation. In *Computer Animation*, Geneva Switzerland. (cited on page 21, 24)

[Desbrun, 1997] Desbrun, M. (1997). *Modélisation et animation d'objets hautement déformables à l'aide de surfaces implicites.* PhD thesis, Institut National Polytechnique, Grenoble. (cited on page 27, 137)

[Desbrun et al., 1999] Desbrun, M., Schröder, P., and Barr, A. (1999). Interactive animation of structured deformable objects. In *Graphics Interface '99 Proceedings*, pages 1–8. (cited on page 51)

[Featherstone, 1983] Featherstone, R. (1983). The calculation of robot using articulated-body inertias. *International Journal of Robotics Research*, 2(1):13–30. (cited on page 28)

[Featherstone, 1987] Featherstone, R. (1987). *Robot Dynamics Algorithms.* Kluwer Academic Publishers. (cited on page 68)

[Fung, 1994] Fung, Y. (1994). *A first course in Continuum Mechanics.* Prentice-Hall, 3rd edition. (cited on page 10, 14)

[Gascuel and Gascuel, 1992] Gascuel, J. and Gascuel, M. (1992). Displacement constraints: A new method for interactive dynamic animation of articulated solids. In *Third Eurographics Workshop on Animation and Simulation.* To appear in The Visual Computer, 1993. (cited on page 28)

[George and Borouchaki, 1998] George, P. and Borouchaki, H. (1998). *Delaunay Triangulation and Meshing - Applications to Finite Elements*. Éditions Hermès. (cited on page 18, 110)

[Gibson, 1997] Gibson, S. F. (1997). 3d chainmail: a fast algorithm for deforming volumetric objects. In Cohen, M. and Zeltzer, D., editors, *1997 Symposium on interactive 3D graphics*, pages 149–154. ACM SIGGRAPH. (cited on page 28)

[Gilbert et al., 1988] Gilbert, E. G., Johnson, D. W., and Keerthi, S. S. (1988). A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203. (cited on page 63)

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley. (cited on page 88)

[Goldsmith, 1960] Goldsmith, W. (1960). *Impact: The Theory and Physical Behaviour of Colliding Solids*. Edward Alrnold, London. (cited on page 68)

[Gottschalk et al., 1996] Gottschalk, S., Lin, M., and Manocha, D. (1996). Obb-tree : A hierarchical structure for rapid inteference detection. In *ACM Siggraph*. (cited on page 67)

[Grzeszczuk et al., 1998] Grzeszczuk, R., Terzopoulos, D., and Hinton, G. (1998). Neuroanimator: Fast neural network emulation and control of physics-based models. In Cohen, M., editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 9–20. ACM SIGGRAPH, Addison Wesley. (cited on page 57)

[Hairer et al., 1987] Hairer, E., Norsett, S. P., and Warnner, G. (1987). *Solving Ordinary Differential Equations, volume 1 : Non-Stiff Problems*. Springer-Verlag. (cited on page 45, 47)

[Hairer and Wanner, 1991] Hairer, E. and Wanner, G. (1991). *Solving Ordinary Differential Equations II*. Springer-Verlag, Berlin. (cited on page 46, 48, 138)

[Hauth and Etzmuß, 2001] Hauth, M. and Etzmuß, O. (2001). A high performance solver for the animation of deformable objects using advanced numerical methods. In *Proc. Eurographics 2001*. (cited on page 50)

[Haykin, 1992] Haykin, S. (1992). *Neural Networks: A Comprehensive Foundation*. Macmillan. (cited on page 58)

[Henry, 1997] Henry, D. (1997). *Outils pour la modélisation de structure et la simulation d'examens échographiques*. PhD thesis, Université Joseph Fourier, Grenoble (FR). (in french). (cited on page xxiii, 83, 103)

[Hilde et al., 2001] Hilde, L., Meseure, P., and Chaillou, C. (2001). A fast implicit integration method for solving dynamic equations of movement. In *ACM Symposium on Virtual Reality Software & Technology 2001*, Calgary, Alberta. (cited on page 133)

[Hoff et al., 2001] Hoff, K., Zaferakis, A., Lin, M., and Manocha, D. (2001). Fast and simple geometric proximity queries using graphics hardware. In *Proc. ACM Symposium on Interactive 3D Graphics*. (cited on page 67)

[Hunt and Crossley, 1975] Hunt, K. H. and Crossley, F. R. E. (1975). Coefficient of restitution interpreted as damping in vibroimpact. *Journal of Applied Mechanics*, pages 440–445. (cited on page 70)

[Jambon et al., 2000] Jambon, A., Querleu, D., Dubois, P., Chaillou, C., Meseure, P., Karpf, S., and Géron, C. (2000). Spic : Pedagogical simulator for gynecologic laparoscopy. In *Proceedings of the 8th Medecine Meets Virtual Reality Conference*, pages 139–145, Newport Beach. (cited on page 3)

[James and Pai, 1999] James, D. and Pai, D. (1999). Art defo - accurate real time deformable objects. In *Proceedings of SIGGRAPH '99 (Los Angeles, California, August 8–13, 1999)*, Computer Graphics Proceedings, Annual Conference Series, pages 65–72. ACM SIGGRAPH, ACM Press. (cited on page 33)

[Joukhadar, 1996] Joukhadar, A. (1996). Adaptive time step for fast converging dynamic simulation system. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, volume 2, pages 418–424. (cited on page 47)

[Joukhadar et al., 1997] Joukhadar, A., Garat, F., and Laugier, C. (1997). Constraint-based identification of a dynamic model. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Grenoble (FR). (cited on page 88)

[Joukhadar and Laugier, 1996] Joukhadar, A. and Laugier, C. (1996). Dynamic simulation: Model, basic algorithms, and optimization. In *Proc. of the Workshop on the Algorithmic Foundations of Robotics*, Toulouse (FR). (cited on page 19)

[Joukhadar et al., 1996] Joukhadar, A., Wabbi, A., and Laugier, C. (1996). Fast contact localisation between deformable polyhedra in motion. In *Proc. of the IEEE Computer Animation Conf.*, pages 126–135, Geneva (CH). (cited on page 66)

[Kotoku et al., 1992] Kotoku, T., Komoriya, K., and Tanie, K. (1992). A force display system for virtual environments and its evaluation. In *IEEE Int. Workshop on Robot and Human Communication (RoMan'92)*, Tokyo (JP). (cited on page 63)

[Krysl et al., 2001] Krysl, P., Lall, S., and Marsden, J. (2001). Dimensional model reduction in nonlinear finite-element dynamics of solids and structures. *International Journal for Numerical Methods in Engineering.* (cited on page 52)

[Kuhnapfel et al., 1995] Kuhnapfel, U., Krumm, H., Kuhn, C., Hubner, M., and Neisius, B. (1995). Endosurgery simulations with kismet: a flexible tool for surgical instrument design, operation room planning, and vr technology based abdominal surgery training. In *Proc. VR '95 WORLD Conference.* (cited on page 3)

[Laffont, 1997] Laffont, P. (1997). Simulation dynamique pour le diagnostic de thromboses veineuses. Mémoire de diplôme d'etudes approfondies, Université de Savoie, Chambéry (FR). (cited on page 84)

[Lee et al., 1995] Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic facial modeling for animation. In *Computer Graphics* Proceedings, Annual Conference Series, Proc. SIGGRAPH '95 (Los Angeles, CA), pages 55–62. ACM SIGGRAPH. (cited on page 20, 84, 87)

[Lin and Canny, 1991] Lin, M. C. and Canny, J. F. (1991). A fast algorithm for incremental distance calculation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, volume 4, pages 1008–1014, Sacramento, CA (US). (cited on page 63)

[Lombardo et al., 1999] Lombardo, J.-C., Cani, M.-P., and F.Neyret (1999). Real-time collision detection for virtual surgery. In *Computer Animation*, Geneva Switzerland. (cited on page 67, 92, 96, 127)

[Louchet et al., 1995] Louchet, J., Provot, X., and Crochemore, D. (1995). Evolutionary identification of cloth animation models. In *Computer Animation and Simulation. Proceedings of the Eurographics Workshop.* (cited on page 88)

[Luciani et al., 1991] Luciani, A., Jimenez, S., Florens, J.-L., Cadoz, C., and Raoult, O. (1991). Computational physics: a modeler simulator for animated physical objects. In *Eurographics'91*, Vienna, Austria. (cited on page 27)

[Marhefka and Orin, 1996] Marhefka, D. W. and Orin, D. E. (1996). Simulation of contact using a nonlinear damping model. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Minneapolis, MN (US). (cited on page 72)

[Mark et al., 1996] Mark, W., Randolph, S., Finch, M., Van Verth, J., and Taylor, R. (1996). Adding force feedback to graphical systems: Issues and solutions. In *Computer Graphics* Proceedings, Annual Conference Series, Proc. SIGGRAPH '96. ACM SIGGRAPH. (cited on page 73)

[Maurel et al., 1998] Maurel, W., Wu, Y., Thalmann, N. M., and Thalmann, D. (1998). *Biomechanical Models for Soft Tissue Simulation.* Springer. (cited on page 9)

[Mendoza and Laugier, 2000] Mendoza, C. A. and Laugier, C. (2000). A solution for the difference rate sampling between haptic devices and deformable virtual objects. In *Proc. of the Int. Symp. on Robotics and Automation*, Monterrey (MX). (cited on page 76)

[Miller and Pearce, 89] Miller, G. and Pearce, A. (89). Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309. This paper also appeared in SIGGRAPH'89 Course notes number 30. (cited on page 25)

[Minoux, 1983] Minoux, M. (1983). *Programmation mathématique - Théorie et algorithme.* Collection Technique et Scientifique des Télécommunications. Dunod. (cited on page 88)

[Mirtich, 1996] Mirtich, B. V. (1996). *Impulsed-based Dynamic Simulation of Rigid Body Systems.* PhD thesis, Univ. of California at Berkeley. (cited on page 68, 69)

[Nebel, 2001] Nebel, J.-C. (2001). Soft tissue modelling from 3d scanned data. In Magnenat-Thalmann, N. and Thalmann, D., editors, *Deformable Avatars*, pages 85–97. Kluwer. (cited on page 87)

[O'Brien and Hodgins, 1999] O'Brien, J. and Hodgins, J. (1999). Graphical models and animation of brittle fracture. In *SIGGRAPH 99 Conference Proceedings*, pages 137–146. (cited on page 12, 21)

[Ottensmeyer and Salisbury, 2001] Ottensmeyer, M. P. and Salisbury, J. K. (2001). In vivo data acquisition instrument for solid organ mechanical property measurement. In *MICCAI*, pages 975–982. (cited on page 109)

[Palazzi, 1993] Palazzi, L. (1993). Deformable models using displacement constraints. Master's thesis, University of British Columbia. (cited on page 28)

[Pars, 1965] Pars, L. A. (1965). *A Treatise on Analytical Dynamics.* Heinemann, London. (cited on page 68)

[Payan et al., 2000] Payan, Y., Collado, R., and Chabanas, M. (2000). A 3d biomechanical model of the human face. In *Proceedings of the 3rd Australian and New Zealand Society of Biomechanics Conference*, Queensland, Australia. (cited on page 133)

[Pentland and Williams, 1989] Pentland, A. and Williams, J. (1989). Good vibrations: Modal dynamics for graphics and animation. *Computer Graphics*, 23(3):215–222. Proceedings of SIGGRAPH'89 (Boston, MA, July 1989). (cited on page 52)

[Picinbono, 2001] Picinbono, G. (2001). *Modèles géométriques et physiques pour la simulation d'interventions chirurgicales.* PhD thesis, université de Nice Sophia-Antipolis. (cited on page 127)

[Picinbono et al., 2001] Picinbono, G., Delingette, H., and Ayache, N. (2001). Nonlinear and anisotropic elastic soft tissue models for medical simulation. In *ICRA2001: IEEE International Conference Robotics and Automation*, Seoul Korea. Best conference paper award. (cited on page 25, 29)

[Picinbono and Lombardo, 1999] Picinbono, G. and Lombardo, J. C. (1999). Extrapolation: a solution for force feedback? In *Actes du Colloque Scientifique International Realite Virtuelle et Prototypage*, pages 117–125, Laval (FR). (in french). (cited on page 73)

[Press et al., 1992] Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1992). *Numerical Recipes in C, second edition.* Cambridge University Press, New York, USA. (cited on page 53)

[Provot, 1995] Provot, X. (1995). Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, pages 147–154. (cited on page 28)

[Richard et al., 1996] Richard, P., Birbent, G., Coiffet, P., Burdea, G., Gomez, D., and Langrana, N. (1996). Effect of frame rate and force feedback on virtual object manipulation. In *Presence.* (cited on page 73)

[Routh, 1905] Routh, E. J. (1905). *Dynamics of a System of Rigid Bodies.* Dover, sixth edition. (cited on page 68)

[Ruspini et al., 1997] Ruspini, D. C., Kolarov, K., and Khatib, O. (1997). Haptic interaction in virtual environments. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, pages 128–133, Grenoble (FR). (cited on page 74, 77, 78)

[Saad and van der Vorst, 2000] Saad, Y. and van der Vorst, H. (2000). Iterative solution of linear systems in the 20-th century. *JCAM.* (cited on page 56)

[Sagar et al., 1994] Sagar, M., Bullivant, D., Mallinson, G., Hunter, P., and Hunter, I. (1994). A virtual environment and model of the eye for surgical simulation. Proc. SIGGRAPH '94 (Los Angeles, CA), pages 205–212. ACM SIGGRAPH. (cited on page xii, 3)

[Shewchuk, 1994] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. Technical Report 94-125, Pittsburgh, PA. (cited on page 54)

[Sundaraj et al., 2000] Sundaraj, K., d'Aulignac, D., and Mazer, E. (2000). A new algorithm for computing minimum distance. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems*, Takamatsu (JP). (cited on page 63)

[Sundaraj and Laugier, 2000] Sundaraj, K. and Laugier, C. (2000). Fast contact localisation of moving deformable polyhedras. In *IEEE Int. Conf. on Automation, Robotics, Control and Vision*. (cited on page 70)

[Szilas and Ronco, 1995] Szilas, N. and Ronco, E. (1995). Action for learning in non-symbolic systems. In *European Conference on Cognitive Science*, Saint Malo, France. (cited on page 88)

[Tendick et al., 2000] Tendick, F., Downes, M., Goktekin, T., Cavusoglu, M., Feygin, D., Wu, X., Eyal, R., Hegarty, M., and Way, L. (2000). A virtual environment testbed for training laparoscopic surgical skills. *Presence*, 9(3):236–255. (cited on page xii, 3)

[Terzopoulos and Waters, 1990] Terzopoulos, D. and Waters, K. (1990). Physically-based facial modelling, analysis, and animation. In *The journal of visualization and computer animation*. (cited on page 16, 20)

[Troccaz et al., 2000] Troccaz, J., Henry, D., Laieb, N., Champleboux, G., Bosson, J. L., and Pichot, O. (2000). Simulators for medical training: application to vascular ultrasound imaging. *Journal of Visualization and Computer Animation*. (cited on page 102, 103)

[Van Gelder, 1998] Van Gelder, A. (1998). Approximate simulation of elastic membranes by triangulated spring meshes. *Journal of Graphics Tools*. (cited on page 19)

[van Overveld and van Loon, 1992] van Overveld, C. and van Loon, E. (1992). Hanging cloth and dangling rods: A unified approach to constraints in computer animation. *The Visual Computer 3*, pages 45–79. (cited on page 28)

[Vieira, 2000] Vieira, S. (2000). Intégration de modéles déformables avec un générateur d'images échographiques. Technical report, Diplome d'Ingenieur, Conservatoire National des Arts et Metiers. in french. (cited on page 102)

[Volino and Thalmann, 1994] Volino, P. and Thalmann, N. (1994). Efficient self-collision detection on smoothly discretised surface animations using geometrical shape regularity. In *EuroGraphics, Computer Graphics Forum*. (cited on page 66)

[Wibaux et al., 1997] Wibaux, L., Varlet, E., and Chaillou, C. (1997). Contruction d'images échographique pour des simulateurs medicaux. In *5eme Journées AFIG 97*. (cited on page 83)

[Witkin, 1999] Witkin, A. (1999). Constrained dynamics. *Physically Based Modeling (SIGGRAPH'99 course notes).* (cited on page 28)

[Wu et al., 2001] Wu, X., Downes, M., Goktekin, T., and Tendick, F. (2001). Adaptive nonlinear finite elements for deformable body simulation using dynamic progressive meshes. In *Proc. Eurographics 2001.* (cited on page 29)

[Young, 1971] Young, D. (1971). *Iterative Solution of Large Linear Systems.* Academic Press, New York. (cited on page 53, 55)

[Zhuang and Canny, 2000] Zhuang, Y. and Canny, J. (2000). Haptic interaction with global deformations. In *IEEE International Conference on Robotics and Automation, ICRA 2000*, San Francisco. (cited on page 29)

[Zilles and Salisbury, 1994] Zilles, C. and Salisbury, J. K. (1994). A constraint-based god-object method for haptic display. In *Proceedings of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems*, volume 1, pages 149–150, Chicago, IL (US). (cited on page 74, 77)

Modélisation de l'interaction avec objets déformables en temps-réel pour des simulateurs médicaux

La qualité du traitement médical des patients s'améliore de jour en jour. Cette amélioration est souvent accompagnée d'une augmentation de la complexité des interventions et les procédures médicales. Cependant, la formation des praticiens est principalement assurée par l'expérience sur un patient réel, avec des dangers évidents pour des procédures mal maîtrisées. D'où l'objectif de notre travail : explorer les possibilités de fournir une formation virtuelle par la simulation sur ordinateur. Souvent, ces procédures médicales impliquent les tissus mous du corps. C'est pourquoi, nous étudions différents modèles d'objets déformables, tels que des masse-ressorts et des éléments finis, et examinons leur adéquation avec les applications médicales en termes d'exactitude et de complexité de calcul. Puisque ces modèles font partie d'un simulateur pédagogique, il est essentiel qu'ils fournissent des solutions en temps réel. Par conséquent, nous examinons la résolution des systèmes résultant des modèles déformables, et discutons les non-linéarités liées à la géométrie et au matériau. De plus, pour pouvoir apprendre, le stagiaire doit pouvoir interagir avec la simulation. Par conséquent, des collisions doivent être détectées et une réponse adéquate calculée. Particulièrement pour des dispositifs de retour d'efforts, la cadence de cette réponse est de grande importance pour une interaction haptique réaliste. Nous appliquons ces méthodes dans le cadre d'un simulateur échographique de la cuisse humaine et d'une simulation laparoscopique du foie humain. Dans ce contexte, nous abordons les questions importantes telles que l'identification et l'évaluation des paramètres du modèle correspondant à des propriétés physiques du tissu.

Mots Clés : simulation médicale, modèles déformables, retour d'efforts, temps-réel.

Modeling interaction with deformable objects in real-time for medical simulators

Medical procedures have much increased in complexity to improve the treatment of patients. However, training is mainly provided by hands-on experience, with obvious dangers to the patient. Hence, it is our aim to explore the possibilities of providing virtual training through computer simulation. Many, if not most, medical procedures involve soft tissues of the body; thus, we study different models for deformable objects, such as mass-springs and finite elements, and examine their suitability for medical applications in terms of their accuracy and computational complexity. Since these models are part of a pedagogical simulator, it is empirical that they provide solutions in real-time; hence, we examine the resolution of the systems arising from the deformable models, and discuss their geometrical and material non-linearities. Further, to be useful, the trainee must be able to interact with the simulation. Therefore, collisions must be detected and adequate response computed; especially when using force-feedback devices, the rate of this response is of great importance for a realistic haptic interaction. We apply these methods in the framework of an echographic simulator of the human thigh and a laparoscopic simulation of the human liver. In this context, we tackle important issues such as the identification of the material properties of the tissue.

Keywords : medical simulation, deformable models, force-feedback, real-time.