Algorithmes de résolution des équations du mouvement pour l'animation basée sur la physique

Présentée et soutenue publiquement le 20 décembre 2002

Pour l'obtention du

Doctorat de l'université des Sciences et Technologies de Lille

(spécialité informatique)

par

Laurent HILDE

Composition du jury

Président :	Michel PETITOT	LIFL, Université de Lille 1
Rapporteurs :	Yannick RÉMION	LERI, Université de Reims
	François FAURE	EVASION, Université Joseph Fourier, Grenoble
	s/c Marie-Paule CANI	EVASION, Institut National Polytechnique de Grenoble
Examinateurs :	Pascal VOLINO	MIRAlab, Université de Genève
	Mathieu DESBRUN	GRAIL, University of Southern California, Los Angeles
Directeurs :	Christophe CHAILLOU	LIFL, Ecole Polytechnique Universitaire de Lille
	Philippe MESEURE	LIFL, ENIC Telecom Lille 1

UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE

Laboratoire d'Informatique Fondamentale de Lille – UPRESA 8022

U.F.R. d'I.E.E.A. – Bât. M3 – 59655 VILLENEUVE D'ASCQ CEDEX

Tél. : +33 (0)3 20 43 47 24 – Télécopie +33 (0)3 20 43 65 66 – email : direction@lifl.fr

Les Remerciements

Merci à Christophe Chaillou de m'avoir donné la possibilité d'effectuer cette thèse.

Merci à Philippe Meseure d'avoir co-encadré ma thèse. Il a été le correspondant privilégié durant ces longues années de labeur.

Un grand merci à Yannick Rémion et à François Faure pour avoir accepté de rapporter ma thèse.

Merci à Mathieu Desbrun et à Pascal Volino pour avoir accepté d'être les examinateurs de ma thèse.

Merci à Michel Petitot qui m'a fait l'honneur de présider mon jury.

Cette page de remerciements serait incomplète si je ne faisais pas mention des membres (ou ex-membre) de l'équipe GRAPHIX que j'ai cotoyé : RoDGeR, Dual, Fab, Luigi, Sam, Jp, Juju, Greg, Hollowman, l'infâme, Jeremy, Luc, Alex et Doug. Je les félicite d'avoir supporter ma logorrhée verbale, ma puissance sonore ainsi que ma tendance à venir embêter un peu tout le monde. Un clin d'œil à Goyan et à Mike pour leur présence dans nos bureaux pour une partie de Quake.

Merci à ma grand-mère, mes parents, beaux-parents, proches et amis qui m'ont apporté leur soutien.

Enfin, je dédie cette thèse à Graziella qui non seulement m'a apporté le soutien nécessaire à l'écriture de ce document mais a surtout réussi à supporter les interférences (nombreuses) de ces mois de rédactions dans notre vie de tous les jours.

Un dernier mot pour embrasser très fort Iléana et sa future sœur.

50/50 ça me paraît raisonnable (bis)

Table des Matières

Introduction

Grâce aux différentes techniques imaginées ces 30 dernières années couplées à un matériel de plus en plus performant, la production d'images de synthèses a énormément gagné en souplesse et qualité. En contemplant celles-ci, qui n'a pas envie de les voir « bouger », de regarder évoluer les phénomènes naturels mis en jeu, de voir interagir les différents éléments du décor, de tester d'autres configurations des éléments de la scène ?

Les techniques d'animation par ordinateur héritent naturellement des techniques traditionnellement utilisées avec le dessin papier ; ainsi une animation sera une juxtaposition d'images représentant les objets dans des positions légèrement différentes. Par contre, l'usage de l'informatique apporte de nombreux avantages comme la possibilité de générer un positionnement de caméra quelconque.

De plus, il est une catégorie d'animation qu'il est difficile à réaliser par les techniques manuelles : la simulation interactive. Il s'agit de réaliser l'affichage d'un environnement le plus proche possible de la réalité tout en permettant à un opérateur externe d'agir sur certain éléments de la scène. La difficulté est que les décisions de cet opérateur sont imprévisibles et qu'il faut penser à prendre en compte tous les phénomènes qui peuvent survenir. L'exemple de simulation interactive qui montre toute l'étendue des possibilités de ce genre d'animation est la simulation de vol. Les pilotes passent en effet de nombreuses heures sur ces appareils d'apprentissages avant de prendre en main un avion réel. C'est ce concept de simulateurs pédagogiques que nous voulons voir étendre à la chirurgie et qui a conduit l'équipe GRAPHIX du LIFL à orienter ses recherches vers la simulation dans des contextes médicaux.

Afin de rendre compte de façon la plus exacte possible le comportement de telles simulations, on tente de copier la « réalité ». On peut se contenter d'un seul mimétisme visuel couplé avec des mécanismes plus ou moins complexes de déclenchement de l'animation à utiliser ; l'autre solution consiste à se servir des modèles générateurs qui décrivent mathématiquement l'évolution du comportement d'une certaine entité. Ainsi une animation basée sur la physique comporte une phase qui consiste en la résolution des équations impliquées dans la description du modèle.

C'est dans le cadre de cette résolution que s'inscrit cette thèse. Nos travaux se basent essentiellement sur la simulation du comportement mécanique de corps (solides rigides, corps déformables, fluides,...). Nous voulons utiliser les différents formalismes mécaniques et pouvoir résoudre tous les types d'équations décrivant ces modèles, qui sont en général des équations différentielles relatives au temps.

Nous devons faire face tout d'abord aux problèmes inhérents aux méthodes employées. Il est en effet primordial de rester dans le cadre des conditions de convergences afin de toujours garder une visualisation crédible. De plus, notre objectif étant d'obtenir une animation interactive, il faut que ces méthodes soient suffisamment rapides. Pour améliorer cette interaction entre la simulation et l'opérateur, nous comptons employer des dispositifs à retour d'effort. Ceux-ci sont plus délicats à utiliser que de simples outils de positionnement : ils imposent un fonctionnement rapide et synchronisé.

Dans le 1er chapitre, nous présenterons le cadre de cette étude, c'est-à-dire la simulation mécanique temps réel et notre déclinaison vers les simulateurs pédagogiques médicaux. Ensuite, dans une 2ème partie, nous ferons un bilan des méthodes de résolution des équations différentielles ordinaires. Le 3ème chapitre mettra en avant les limites des méthodes dites explicites. La 4ème partie décrit la méthode de Broyden de résolution de systèmes non-linéaires par laquelle nous arrivons à simuler des scènes en un temps interactif avec des méthodes implicites. Le 5ème chapitre présente notre approche de séparation du système mécanique global en sous-systèmes indépendants.

Chapitre 1 : Contexte

1.1 Simulation Mécanique Temps Réel

1.1.1 Simulation Temps Réel

De nombreux procédés permettent d'obtenir des animations d'objets très variés. De manière générale, ces procédés sont pilotés par un but ; par exemple, cela peut-être des effets spéciaux comme la transformation d'une forme en une autre [TO99] ou encore la reproduction d'un phénomène naturel comme une coulée de lave [SACN99]. Pour évaluer la pertinence de l'animation, on peut se contenter d'une appréciation empirique : l'important est que cela paraisse plaisant et naturel pour notre perception visuelle. On parle alors de simulation plausible : elle semble correcte mais n'est pas directement basée sur des mesures expérimentales. Lorsque l'on désire une meilleure adéquation avec celles-ci, on procède alors à une identification des paramètres : on détermine les valeurs des paramètres du modèle pour faire en sorte qu'il reproduise à l'identique les comportements observés lors des expériences.

On peut distinguer deux façons d'aboutir à une animation d'un phénomène réel ; La première est d'utiliser des techniques habituelles d'animation. Par exemple, pour réaliser le mouvement d'une planète autour d'une autre, la trajectoire peut être facilement déduite grâce à la résolution formelle des équations de la mécanique. Connaissant le chemin que doit suivre la planète, il suffit d'ajouter une fonction de régulation du temps qui contrôle par exemple la distance parcourue le long de la trajectoire. Cette re-création du mouvement orbital est conforme au réel mais limitée aux paramètres physiques des 2 planètes (masses, positions et vitesses initiales). Ainsi il faudra recalculer la trajectoire si l'on change un seul de ces paramètres. Il apparaît alors clairement qu'il serait plus souple de faire calculer ces trajectoires à la volée.

On rejoint alors le domaine de la seconde technique, i.e. la simulation : on fournit des valeurs d'entrée à une boîte noire et celle-ci calcule l'évolution des variables du système étudié. La simulation est un procédé couramment utilisé et ce dans de nombreux domaines. Le but est de pouvoir reproduire par calcul un phénomène ou le fonctionnement d'une machinerie complexe. La première étape consiste en la modélisation de ce que l'on veut observer. Cette modélisation se doit d'être la plus conforme au comportement observé dans la réalité. Vient ensuite le calcul de l'évolution des entités modélisées. Enfin la suite de valeurs générée peut servir à divers traitements ; dans notre cas cela servira essentiellement à générer un affichage de la scène et le retour de l'effort.

L'usage de la simulation à des fins de production d'une animation en image de synthèse est généralement appelée animation basée sur la physique. Ces 2 termes sont souvent employés avec le même sens. Mais on peut tout de même faire une distinction entre ces 2 notions. [AD92] montre qu'une animation peut être basée sur la physique sans toutefois avoir la précision d'une simulation « complète ». De manière générale, une telle simulation est très complexe et des simplifications sont souvent nécessaires afin de permettre leur calcul. On peut dire que la frontière entre animation basée sur la physique et la simulation est floue : il s'agit d'une appréciation sur les simplifications apportées au modèle le plus proche possible de l'objet simulé.

Ces simplifications sont motivées soit par le fait que les équations mises en jeu sont quasiment impossibles à résoudre dans le cas général, soit par le fait qu'on veuille générer une animation interactive. Dans ce cas, il faut réussir à faire que le calcul de l'évolution de la scène soit suffisamment simple pour s'effectuer dans un temps raisonnable. C'est ici que l'on peut faire la distinction entre simulation et animation : dans une animation, on va simplifier le modèle sans se soucier de coller au modèle physique. Dans l'exemple de [AD92], la voiture tournera dès que l'utilisateur aura manipulé le volant alors que la simulation fera tourner les roues grâce au couple appliqué sur le volant et ce sont les interactions entre les roues et le sol d'une part et les roues et le reste de la voiture d'autre part qui la fera avancer.

Par contre, la complexité du modèle de simulation de la voiture peut empêcher son interactivité, i.e. la réaction aux stimuli de l'utilisateur. On peut alors simplifier ce modèle : par exemple en simplifiant la transmission du couple appliqué au volant. Ainsi on reste beaucoup plus proche de la simulation originelle.

Nous nous plaçons dans le domaine de l'animation basée sur la physique, c'est-à-dire que l'on utilise les principes de la simulation mais que l'on s'autorise à simplifier le modèle de simulation plus proche de la réalité. Cette simplification peut être gênante si l'on veut faire une étude précise du comportement de chaque partie (par exemple connaître les efforts reçus, etc..) ; par contre, cela peut être suffisant pour une animation qui aura un comportement très proche de la réalité. Dans la suite de ce document, nous utiliserons le terme simulation pour désigner aussi ce type d'animation.

Une dernière catégorie de simulation peut être définie : la simulation temps réel. Une animation est une juxtaposition d'images représentant la scène à des instants différents et réguliers. Ce mode de représentation induit un échantillonnage temporel. Cela signifie que l'on devra calculer les différentes variables du modèle de façon périodique. Le terme « temps réel » signifie que le temps passé dans le calcul d'un nouvel état ne doit pas excéder le temps simulé. Ainsi on peut produire un affichage qui s'effectuera à la même vitesse que le phénomène représenté.

C'est un aspect important pour une simulation qui veut pouvoir interagir avec un opérateur humain : il faut que le comportement de ce qu'il voit soit conforme à ses attentes. Il faut donc pouvoir assurer une synchronicité entre le temps de calcul et le temps simulé par l'ajout éventuel d'une attente. C'est cette simulation temps réel que nous cherchons à réaliser dans le cadre d'une simulation mécanique. Pour cela, nous allons présenter le type d'équations que nous allons devoir résoudre.

1.1.2 Équations Mécaniques

De nombreux formalismes existent pour décrire le comportement mécanique d'un objet. Notre but n'est pas de décrire de manière exhaustive l'ensemble des modèles mécaniques mais de présenter ceux qui sont représentatifs de ce que nous voulons pouvoir simuler. La description de ces modèles permet de mieux connaître le type d'équations que l'on doit résoudre.

1.1.2.1 Principes de la mécanique classique

Elle trouve ses fondements dans l'énoncé des trois lois de Newton :

❶ l'existence de référentiels galiléens

❷ dans un tel référentiel, l'accélération d'une particule de masse m soumise à la force $\vec{f}$ est donnée par $\vec{a} = \vec{f} / m$ (Relation Fondamentale de la Dynamique)

❸ principe de l'action et de la réaction ; pour 2 particules P_1 et P_2 : $\vec{f}_{12} + \vec{f}_{21} = 0$ avec $P_1P_2 \wedge \vec{f}_{12} = 0$.

Afin de calculer la trajectoire d'un point matériel, il suffit donc de résoudre l'équation suivante :

$$\vec{a} = \frac{d^2\vec{x}}{dt^2} = \frac{\vec{f}}{m} \tag{1.1}$$

$\vec{f}$ est la somme des différentes forces appliquées à la masse ponctuelle et $\vec{x}$ représente son vecteur position (il contient les trois coordonnées x, y et z nécessaires à sa caractérisation). Nous obtenons une EDO, équation différentielle ordinaire (ne faisant pas intervenir des dérivées partielles) du second ordre (on différencie « 2 fois »).

La mécanique du solide

Un objet a généralement besoin d'autres variables pour décrire son positionnement dans l'espace. On appelle ces différentes variables des degrés de libertés (DDL). Ainsi, un solide rigide (un corps non ponctuel qui ne se déforme pas) emploie trois DDLs qui représentent la position de son centre de gravité et trois DDLs nécessaire à la description de son orientation. Le centre de gravité G obéit à l'équation de Newton (1.1) en considérant la masse de l'objet concentrée en ce point et en appliquant les forces extérieures en ce point (La somme de ces vecteurs force s'appelle la résultante). Le mouvement en rotation suit une loi similaire à la RFD :

$$\frac{d\left(I\vec{\Omega}\right)}{dt} = \vec{\mathcal{M}} \tag{1.2}$$

$\vec{\Omega}$ est le vecteur rotation instantanée, $\vec{\mathcal{M}}$ est la somme des moments[1] appliqués sur le solide et I la matrice d'inertie. Il faut ensuite obtenir l'orientation du solide à partir de $\vec{\Omega}$. Si on utilise un quaternion q pour représenter l'orientation du solide, l'équation permettant d'obtenir l'évolution du quaternion est :

[1] Si une force F_A s'applique en A, alors le moment $\vec{\mathcal{M}}_A = \overrightarrow{GA} \wedge \overrightarrow{F_A}$.

$$\dot{q}(t) = \frac{1}{2}\vec{\Omega}.q \tag{1.3}$$

A nouveau, nous aboutissons à des EDO.

1.1.2.2 Les corps déformables

Le but de ce paragraphe n'est pas de faire un état de l'art sur les modèles de corps déformables. Il s'agit d'exposer quelques grandes catégories pour montrer quel type d'équations sont mises en jeu.

Modèles discrets

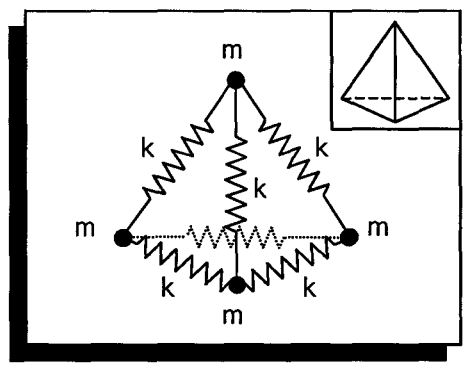


Figure 1.1 : exemple de modélisation masses/ressorts

Ces modèles se basent sur une discrétisation de la forme du corps. Celle-ci est constituée de points massiques munis d'une loi d'interaction. C'est cette dernière qui définit le comportement de l'objet. Il s'agit donc d'appliquer la mécanique du point (voir 1.1.2.1) pour chacune de ces particules massiques. Nous aboutissons aux mêmes équations ; la différence est que celles-ci portent sur un grand nombre de variables.

Parmi ces modèles, le plus répandu est le maillage masses/ressorts. Il s'agit de constituer un maillage dont chaque sommet est relié à un certain nombre de voisins par des ressorts (Figure 1.1) qui sont régis par l'équation suivante :

$$\vec{f} = -kx \tag{1.4}$$

Cette représentation est très populaire par sa simplicité de mise en œuvre mais elle pose des problèmes quant à l'identification des paramètres du modèle à un corps réel.

Modèles continus à résolution discrète

L'intérêt de ces méthodes est de s'appuyer sur des lois de comportement directement issues du domaine de la mécanique. En effet, il existe de nombreuses études sur la manière dont un corps se déforme, notamment en fonction du matériau qui la compose. Ces lois d'élasticité reposent sur une définition continue de l'objet déformable.

Cependant, afin de résoudre les équations gouvernant le comportement de l'objet, on opère une discrétisation spatiale qui permet la résolution du problème. Il existe 2 principales méthodes qui diffèrent par l'usage qu'elles font de cette discrétisation : les différences finies et les éléments finis.

Les différences finies

C'est une méthode classique de résolution d'équations aux dérivées partielles, notamment celles qui mettent en jeu des différentiations spatiales et temporelles. Cette méthode a été introduite en animation par [TPBF87] ; le comportement d'un objet déformable est défini par l'équation suivante :

$$\frac{\partial}{\partial t}\left(\mu\frac{\partial}{\partial t}\vec{r}\right)+\gamma\frac{\partial}{\partial t}\vec{r}=\vec{f}_{ext}+\vec{f}_{int} \tag{1.5}$$

Le vecteur $\vec{r}$ représente un point de l'objet dans le repère global, $\mu(\vec{r})$ la densité de l'objet au point $\vec{r}$, $\gamma(\vec{r})$ la densité de frottements ; dans $\vec{f}_{ext}$, on trouve l'ensemble de forces externes appliquées au corps et $\vec{f}_{int}$ représente les forces de déformations de l'objet. Ces forces internes peuvent être définies comme dérivant d'un potentiel : $\vec{f}_{int}=grad_{\vec{r}}\left(\varepsilon(\vec{r})\right)$; on modélise alors un objet dit hyperélastique ([TPBF87]). Ce potentiel est défini à l'aide d'un tenseur métrique qui fait intervenir des dérivées partielles spatiales.

Le principe des différences finies est d'établir une grille régulière de l'espace et l'équation (1.5) va être résolue uniquement en ces points. Le fait d'avoir une grille régulière était important pour le calcul des opérateurs différentiels spatiaux. Mais dans [DDBC99], des opérateurs différentiels ont été définis pour être utilisés sur un voisinage quelconque d'un point.

Une fois cette discrétisation effectuée, on aboutit à une EDO d'ordre 2 :

$$M\frac{d^2}{dt^2}\vec{u}+C\frac{d}{dt}\vec{u}+K(\vec{u})\vec{u}=\vec{f}_{ext} \tag{1.6}$$

Les éléments finis

Cette méthode est largement utilisée dans le domaine de la mécanique, notamment dans l'étude du comportement en déformation de structure métallique. Son principal avantage est qu'elle respecte très bien le caractère continu du corps étudié. La présentation faite ici est très succincte, on pourra se reporter à des ouvrages spécialisés (comme par exemple [Bat82]) pour de plus amples détails.

Le principe est de décomposer l'objet en éléments volumiques simples. Ces éléments sont caractérisés par un nombre faible de points appelés nœuds et une formule d'interpolation permettant d'obtenir la position d'un point quelconque de l'élément à partir des nœuds. Le nombre de nœuds est fonction de la continuité désirée pour la solution calculée. Il suffit maintenant de calculer l'évolution des nœuds.

En général, on choisit des fonctions d'interpolation basées sur des compositions linéaires des noeuds ; ceci permet d'exprimer l'équation d'équilibre du corps par un système d'équations linéaires aux nœuds :

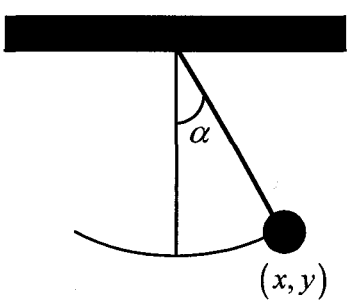
$$K(\vec{U})\vec{U} = \vec{F} \tag{1.7}$$

où $\vec{U}$ rassemble les déplacements des nœuds, $\vec{F}$ les forces externes et où K est une matrice appelée matrice de rigidité. Pour obtenir les déformations au cours du temps, on ajoute des termes correspondants à des accélérations et des vitesses de ces déplacements :

$$M\frac{d^2}{dt^2}\vec{U} + D\frac{d}{dt}\vec{U} + K(\vec{U})\vec{U} = \vec{F} \tag{1.8}$$

On s'aperçoit que cette équation est similaire à (1.6).

Modèles lagrangiens



Certains problèmes de mécaniques s'expriment plus simplement en fonction de leur DDLs que des variables de position dans le repère globale. L'exemple le plus simple est l'étude d'un pendule. Son mouvement plan peut être repéré par l'angle α au lieu du vecteur $(x, y)^t$. Une formulation des équations régissant un mouvement a été construite autour de cette idée de définir un objet par des DDLs : c'est la mécanique lagrangienne. Si l'on nomme $\vec{q}$ le vecteur regroupant tous les degrés de libertés du système, le système vérifie l'équation suivante :

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = Q \tag{1.9}$$

où $L = K - U$, K représentant l'énergie cinétique, U l'énergie potentielle, et où Q représente les forces appliquées au système exprimées en fonction de q et $\dot{q}$. De manière générale, l'énergie potentielle ne dépend que de q et, si l'on choisit une répartition massique ne dépendant pas de q, l'énergie cinétique ne dépend que de $\dot{q}$. Ainsi (1.9) devient :

$$\frac{d}{dt}\left(\frac{\partial K}{\partial \dot{q}}\right) = Q - \frac{\partial U}{\partial q} \tag{1.10}$$

Ce formalisme a été utilisé par [Nou99] pour simuler une maille de tricot en utilisant une spline pour le fil. Les DDLs sont les points de contrôle de la spline et on possède une expression de l'énergie cinétique de la spline. En effectuant formellement, les dérivations de K et de U, on obtient :

$$M\ddot{q} = Q(q,\dot{q}) - U'(q) \tag{1.11}$$

qui est une EDO d'ordre 2. Cette technique sur une spline peut être mise en œuvre de façon identique sur des surfaces ou des volumes à points de contrôle [Noc01].

1.1.3 Résolution Dynamique ou Statique

Nous avons vu dans la section 1.1.2 que, pour obtenir un moyen de calculer l'évolution d'un objet mécanique, on manipule les équations décrivant son comportement et/ou son mouvement afin de se ramener à une EDO d'ordre 2. Il reste à utiliser des méthodes adaptées à de telles équations, ce que nous verrons dans le chapitre 2.

Ces méthodes présentent des problèmes de convergence et il faut tenir compte de l'aspect temporel. En effet, lors d'une simulation dynamique temps réel, il faut assurer la synchronisation entre les positions calculées et le temps écoulé. Si l'on calcule la position future avec les paramètres temporels positionnés pour une période de 10 ms, on doit obtenir le résultat du calcul en au plus 10 ms.

Ceci est d'autant plus important lorsqu'un utilisateur intervient dans la simulation. En effet, si l'on imagine que pour une simulation de 10 ms, il faut 20 ms de temps de calcul, cela veut dire que les mouvements de l'utilisateur vont être acquis tous les 20 ms. Cependant le déplacement mesuré va être interprété comme un déplacement s'étant effectué en 10 ms par le moteur dynamique. Ainsi les vitesses sont multipliés par 2 et les accélérations par 4, ce qui rend le système plus instable car sujet à des déplacements trop rapides.

Pour s'abstraire de ces contraintes temporelles, on peut utiliser une autre façon de déduire l'évolution de la forme de l'objet : le calcul de l'état d'équilibre ou résolution statique. Il s'agit de trouver la configuration d'équilibre de l'objet suivant les forces qui lui sont appliquées. Les équations à résoudre se déduisent des équations de la section 1.1.2 en mettant simplement à zéro tous les termes relatifs au temps, i.e. les vitesses et les accélérations. Dans le cas général, on aboutit à un système d'équations non-linéaires mais qui est souvent simplifié en un système linéaire grâce à certaines approximations sur la modélisation des forces de déformations (élasticité linéaire, petits déplacement : c'est l'équation (1.7) avec la matrice K ne dépendant pas de $\vec{U}$).

Pour que cette technique soit utilisable, le corps doit être suffisamment contraint pour qu'il existe une position d'équilibre. Par exemple, il est impossible de simuler un objet qui se déforme et se déplace en même temps. En effet, dans ce cas, il existe plusieurs configurations d'équilibre : l'objet peut s'être uniquement déformé ou déplacé, la réalité se situant quelque part au milieu. De même, un objet possédant par nature plusieurs configurations stables devient problématique. C'est le cas pour des objets qui ne possède pas de forme de référence : par exemple une ficelle posée sur une table peut prendre plusieurs postures en état d'équilibre.

Une fois que l'on s'est assuré de son unicité, la position d'équilibre représente la position finale (le régime stationnaire) vers laquelle doit tendre l'objet simulé. De cette façon, on évite les problèmes de stabilité ou d'oscillations intempestives dus à l'utilisation d'une résolution

dynamique. Cependant, il manque l'information temporelle qui permet de savoir le chemin suivi par la déformation et le temps qu'elle a mis pour atteindre cette position. Cela rend donc impossible la simulation de phénomènes rapides de transitions vibratoires ou encore les retours à l'équilibre très lent.

Si les forces appliquées au corps varient lentement, cela laisse le temps au système de réagir et donc d'atteindre quasi-instantanément sa position d'équilibre. Par contre, dès qu'un mouvement est trop rapide, on s'expose à voir des objets revenir directement à un état d'équilibre alors que cela ne devrait pas être le cas. Pour juger de la pertinence de la simulation, il faut évaluer la vitesse de réaction des corps mécaniques et faire en sorte que les actions extérieures ne dépassent pas cette vitesse.

Un autre problème de cette absence de vitesses est que l'on ne peut pas incorporer les comportements dynamiques. Ainsi, la viscosité est un phénomène purement dynamique et donc la résolution statique n'en tient absolument pas compte. Par exemple, pour un simulateur d'amniocentèse [Wib00], les différentes couches de la peau ont été modélisées grâce à la méthode des éléments finis [Tur96] afin de rendre compte de l'insertion de l'aiguille de ponction. Or il est impossible de rendre compte des forces de frottements entre l'aiguille et les différentes couches traversées avec une résolution purement statique. Une surcouche dynamique a donc été ajoutée afin de pouvoir rendre compte de ces frottements.

Cette résolution par suite d'états d'équilibre est très utilisée dans les domaines de la mécanique des matériaux qui s'intéressent très souvent à la résistance d'une structure à un effort. Sa première utilisation en animation a été faite par [GTT89] pour simuler la prise en main d'une balle. Elle est aussi utilisée en simulation médicale lorsque celle-ci se place dans un cadre respectant les restrictions précédentes. C'est le cas de la simulation d'intervention cœlioscopique abdominale sur le foie [BC96]. Le foie est en effet un organe qui ne peut se déplacer du fait de ses liens avec le reste du corps et ainsi l'absorption des efforts se fait uniquement par déformation. De plus, comme dans toutes interventions chirurgicales, les gestes des praticiens sont lents et laisse à la dynamique du foie le temps de réagir.

L'affichage successif des positions d'équilibre ne produit pas en général une animation satisfaisante. On peut imaginer d'effectuer une interpolation entre les positions calculées. [Aul01] utilise une méthode itérative pour résoudre le système d'équations et propose de se servir des étapes successives de la résolution pour générer l'animation. On obtient une animation fluide et visuellement acceptable mais dont les variations temporelles ne sont pas liées à la vitesse.

Ces ajouts de dynamique montrent clairement qu'il est des cas où elle est nécessaire. De plus, les contraintes imposées par une résolution statique ne sont pas compatibles avec notre objectif de simulation générique. C'est pourquoi nous avons opté pour la résolution dynamique.

1.2 Mise en œuvre d'une simulation mécanique temps-réel

1.2.1 Architecture logicielle

Les travaux de l'équipe GRAPHIX en ce qui concerne la simulation mécanique temps réel ont commencé avec les travaux de P. Meseure ([Mes97]). Le but de ces travaux était la définition d'un modèle mécanique destiné à simuler des corps élastiques. Une des applications de ceux-ci étaient de simuler des ovaires. Il était alors inévitable de faire des développements spécifiques pour chaque application qui devait utiliser ce modèle.

En plus de ces problèmes de génie logiciel, les travaux de P. Meseure avait mis en évidence des problèmes liés à l'utilisation de modèles à facettes pour la gestion des collisions d'objets déformables. Ainsi, en 1999 a été lancé le projet SPORE : Simulation Physique d'Objets virtuels dédiée eu Retour d'Effort. Le but de ce projet est la définition d'une bibliothèque d'éléments logiciels afin de faciliter le développement d'applications mettant en jeu de l'animation temps réel d'objets mécaniques et utilisant un dispositif à retour d'effort. Un des points de SPORE est de faire coexister des modélisations mécanique variées.

Cet ensemble logiciel s'articule autour de la définition d'une boucle de simulation dynamique (voir Algorithme 1). L'architecture de SPORE se base sur un noyau de simulation dynamique. Celui-ci simule des corps physiques décrit sous forme d'un modèle en trois parties. Enfin, il traite de manière spécifique l'utilisation de dispositifs haptiques. Ces trois aspects sont développés ci-après.

```
Tant que (non fini) faire
   Acquérir position outils
   Calcul collision
   Calcul retour d'effort
   Calcul évolution des corps
Fin
```

Algorithme 1 : une boucle de simulation classique

1.2.1.1 Le noyau

Le noyau concentre les opérations qui sont la base de la simulation dynamique. C'est lui qui a en charge le calcul de l'évolution des corps à simuler, qui effectue la détection de collision et qui gère les contraintes entre corps. Les corps sont classés en trois grandes catégories : les corps inertes, les corps actifs et les corps passifs. La distinction des corps inertes permet de prendre en compte le fait qu'ils sont immobiles afin d'accélérer les traitements des collisions et leur affichage.

Les corps actifs regroupent les corps qui fixent leur positions : ce sont les objets dont la position est indexée sur un quelconque périphérique. On ne calcule donc pas leur évolution mais on récupère la position ou le mouvement relatif de ces objets afin de les replacer dans la

scène dynamique. Les périphériques haptiques manipulent ce type de corps et donc, le calcul d'effort à fournir ne peut se faire qu'au travers de ce type d'objet.

Les corps passifs représentent tous les corps qui ont une représentation mécanique grâce à laquelle on peut calculer leur évolution. Ils sont passifs dans le sens où ils ne font que subir les actions externes. Leur description se trouve dans la section suivante. Le calcul de l'évolution de ces corps passe par une résolution dynamique (nous envisageons de mettre aussi en place une résolution statique des corps). La façon de résoudre les EDO sera largement explicitée dans la suite de ce document.

En ce qui concerne la détection de collision, des travaux précédents (voir [Mes02]) ont étudiés les méthodes par contact : si deux objets sont en collision, il s'agit de mettre en œuvre une stratégie pour imposer la non-interpénétration. On obtient alors une représentation précise des collisions ; cependant les algorithmes mis en œuvre sont généralement coûteux notamment pour la prise en compte d'interactions multiples.

Nous nous sommes alors dirigés vers une méthode à pénalité ([PB88]) : si 2 objets sont en intersection, des forces répulsives sont crées en fonction de la distance d'interpénétration. Pour évaluer celle-ci, nous avons choisi d'utiliser une approximation des objets par un ensemble de sphères. La détermination des sphères en collision est rapide d'autant plus que l'espace est subdivisé en voxels afin de restreindre les calculs à des sphères se situant dans une même zone

Enfin, la gestion de contraintes entre corps est pour l'instant limitée à des contraintes molles, c'est-à-dire la liaison de 2 points par l'intermédiaire d'une interaction élastique. Les contraintes de liaisons fortes sont en cours d'études

1.2.1.2 Les modèles

Un corps mécanique de SPORE aura trois composantes. La gestion des collisions demande une modélisation « en sphères ». Il faut ajouter à cela une partie mécanique qui décrit la façon dont l'objet se comporte et une partie géométrique qui indique comment l'objet s'affiche. La distinction de ces trois composantes est guidée par le souhait de toujours garder des performances optimales. La décomposition en sphères permet d'obtenir des collisions rapides et avec n'importe quel type d'objets (mobile ou immobile, rigide ou déformable, lié à un périphériques ou non). La Figure 1.2 montre la décomposition possible d'un corps rigide en ses composantes

La séparation de la représentation mécanique et géométrique vient du fait que l'on est capable d'afficher rapidement des objets complexes alors qu'une simulation mécanique interactive ne peut prendre en compte qu'un nombre restreint de DDLs. Nous utilisons donc cette notion d'habillage de corps mécanique, illustrée par la thèse de F. Triquet ([Tri01]).

La Figure 1.3 montre la décomposition d'une sphère déformable. Il est à noter que ce n'est qu'un exemple de décomposition. Les ressorts ne sont pas tous présents pour des raisons de lisibilité. Dans cet exemple, la partie géométrique et la partie mécanique sont directement dépendante dans leur forme mais l'intérêt de leur séparation est de pouvoir choisir une autre

représentation comme le montre la Figure 1.4. La liaison entre la composante mécanique et géométrique devient un point auquel il faut porter attention.

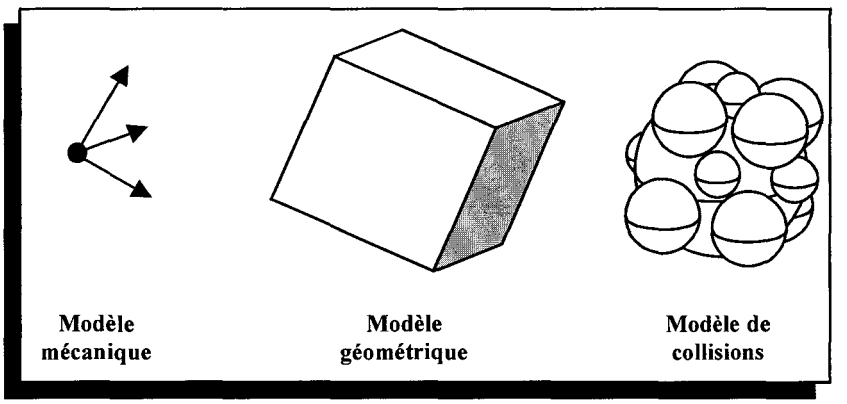


Figure 1.2 : Exemple de décomposition d'un corps rigide

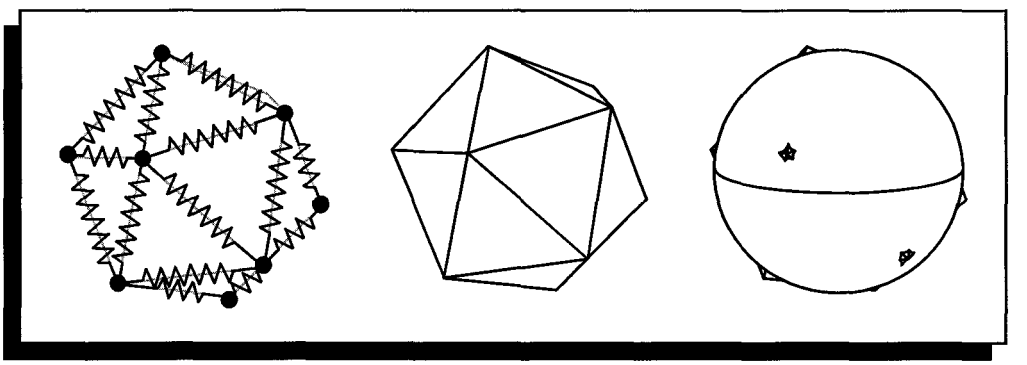


Figure 1.3 : Exemple de décomposition d'un corps déformable

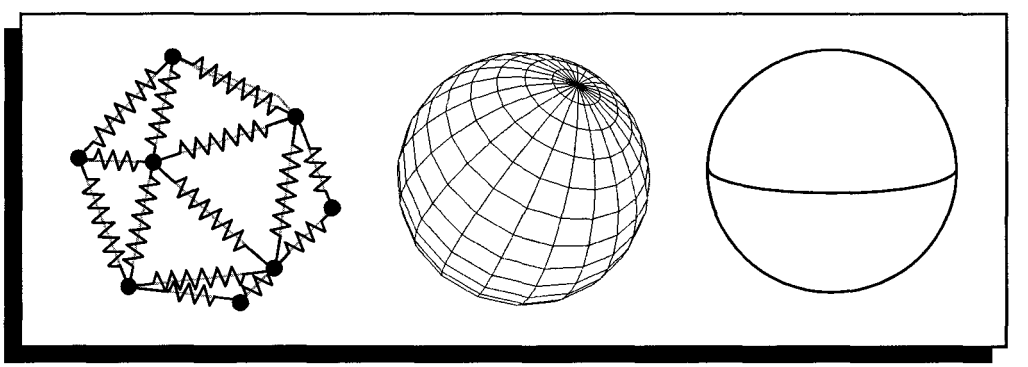


Figure 1.4 : Modification de la forme géométrique

L'approximation par sphères permet d'obtenir une méthode efficace de détection de collision mais sa construction est délicate. Actuellement, la construction se fait par des méthodes simples avec ensuite une intervention manuelle importante. Nous entamons une exploration des travaux déjà effectués dans ce domaine, notamment par l'intermédiaire d'une collaboration avec le LERI ([Pre01]).

SPORE incorpore plusieurs modèles mécaniques : les solides indéformables, particules avec forces de Lennard-Jones et affichage en surfaces implicites ([TMC01]), B-splines dynamiques 1D et 2D ([Len01]), maillages Masses-Ressorts surfaciques pour les tissus ([Dav00]) et

ainsi qu'une version avec composante de référence rigide ([MC00]). Cet ensemble apporte déjà une grande diversité et des études sont menées soit vers d'autres modèles mécaniques (éléments finis, corps rigides articulés), soit vers des extensions des modèles existants (lois de déformations, multirésolution).

1.2.1.3 Le retour d'effort

L'usage de dispositifs haptiques impose de leur fournir des informations de commande à une fréquence suffisante (500Hz à 1kHz [PLDA00]). Il est utopique de vouloir faire fonctionner à une telle vitesse une simulation dynamique comportant plusieurs objets mécaniques dont certains sont déformables. Pour maintenir une fréquence de fonctionnement suffisante pour un dispositif à retour d'effort, nous avons déporté le calcul des efforts à renvoyer dans une boucle indépendante des calculs de simulation.

La seule préoccupation de cette boucle est d'évaluer les collisions du périphérique avec l'environnement. La boucle possède une information du même type que la boucle de la dynamique : une subdivision de l'espace avec les sphères de collisions des corps positionnés. Cette grille est considérée comme inchangée d'un pas de calcul dynamique à l'autre ; la nouvelle position des sphères de collision est communiquée à la fréquence des calcul de la simulation (quelques dizaines de Hz). La mise à jour effective dans la grille de voxels de la boucle haptique est effectuée de manière progressive, à la fréquence du retour d'effort, par une interpolation entre l'ancienne position des sphères et leur nouvelle position dans le but d'assurer une continuité des efforts.

Possédant cette structure, il suffit d'effectuer une détection de collision avec les sphères approximant la forme du périphérique. Cette technique permet un fonctionnement suffisamment rapide [DMC02] pour entretenir la fréquence de fonctionnement d'un dispositif haptique. Cette procédure est utilisée actuellement avec 2 périphériques à retour d'effort : le Phantom de la société Sensable et avec un périphérique dédié à la cœlioscopie gynécologique ([LC99]).

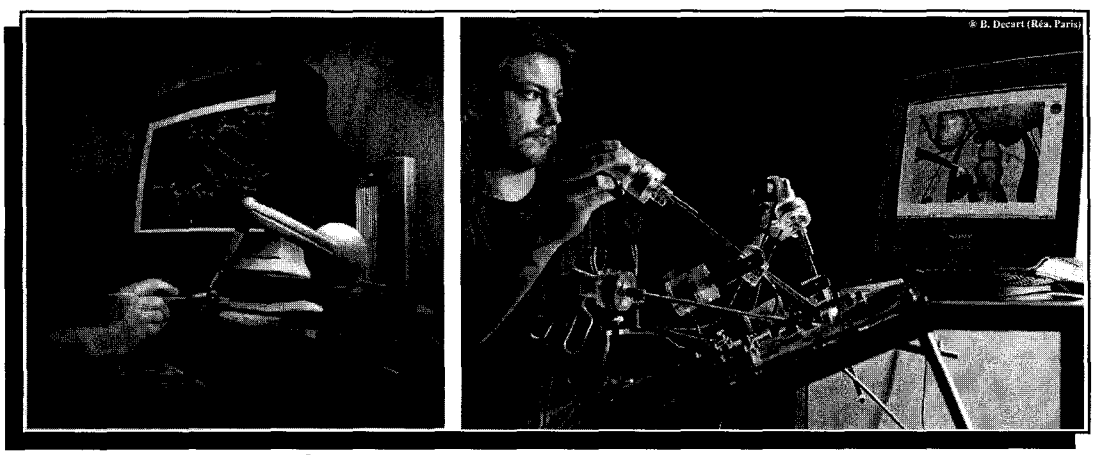


Figure 1.5 : À gauche le Phantom et à droite notre périphérique

1.2.2 Conception d'une application

Le principe d'utilisation de SPORE est classique : on fournit une API qui permet d'échanger les informations nécessaires à une application. Les principaux points de communications sont :

1. La définition des corps : c'est une phase d'initialisation qui indique à SPORE les corps utilisés. En déclarant un corps actif, il devient potentiellement l'objet sur lequel sera calculé des efforts.
2. Le calcul d'un pas de simulation dynamique, la durée à simuler est donnée en initialisation.
3. L'affichage d'un objet de SPORE : il est constitué de commandes OpenGL
4. Le positionnement d'un corps actif dans SPORE et le calcul des efforts à renvoyer

Il est souhaitable de faire fonctionner l'application selon trois boucles indépendantes : une pour la dynamique, une pour le retour d'effort et une pour l'affichage. Ainsi l'application est fractionnée en trois composantes fonctionnant à des fréquences différentes (Figure 1.6). C'est l'application qui a alors en charge de gérer la synchronicité de la dynamique ainsi que celle du retour d'effort.

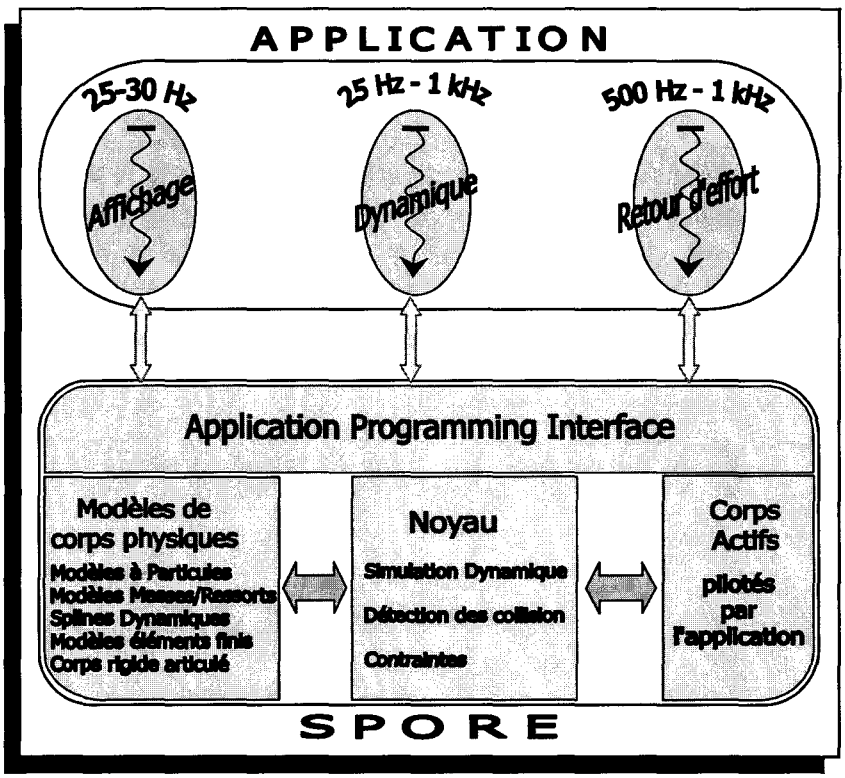


Figure 1.6 : Design d'une application utilisant SPORE

La bibliothèque SPORE peut aussi gérer elle-même la boucle de simulation dynamique, contrôlant alors la synchronicité. Enfin, il est tout à fait possible de constituer une application composée d'un seul fil d'exécution mais cela peut nuire à la qualité de la simulation, notamment quant à la stabilité du dispositif haptique.

Pour l'instant, nous effectuons nos tests sur une machine bi-processeurs sous Windows NT/2000. Il est assez difficile de pouvoir assurer une réelle synchronicité sur un tel système d'exploitation. Nous pensons qu'il serait souhaitable d'exécuter les boucles de la dynamique et de gestion du retour d'effort sur un système d'exploitation (SE) temps réel tout en laissant la boucle d'affichage sur une machine avec un SE gérant les cartes 3D.

L'architecture matérielle idéale est alors un ordinateur bi-processeur avec un SE temps réel et un ordinateur avec un SE avec interface graphique et gérant les cartes 3D (Figure 1.7). Le débit de la liaison entre les 2 machines est assez élevé mais reste en deçà des limites de liaison classique.

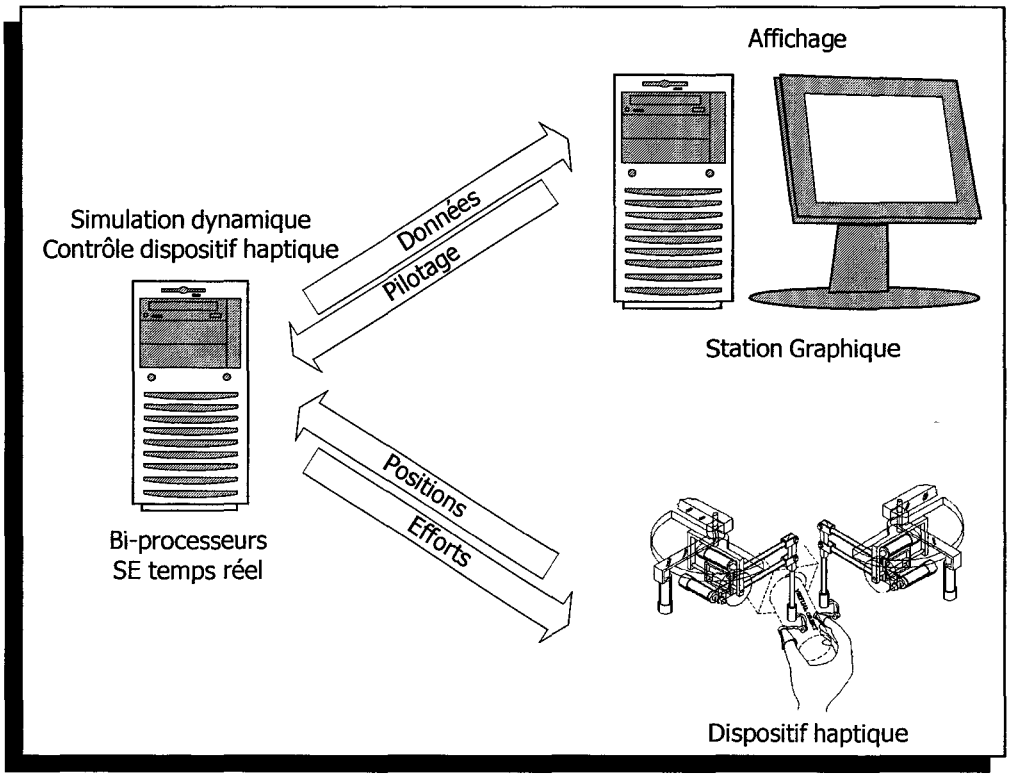


Figure 1.7 : Architecture matérielle souhaitable pour une application utilisant SPORE

1.3 Les simulateurs pédagogiques médicaux

1.3.1 Principe

Le but d'un simulateur pédagogique est de reproduire les conditions réelles d'une manipulation que des « étudiants » doivent apprendre. Ce type de simulateur existe depuis longtemps dans l'aviation et tend à se développer vers d'autres domaines comme la conduite de véhicules spéciaux. Un simulateur comporte de nombreux intérêts pour l'apprentissage de ce type de tâches.

La formation aux techniques de manipulation sur un simulateur permet de réduire les coûts : les erreurs éventuelles ne se solderont que par une catastrophe virtuelle. De plus, lors de la première mise en situation opérationnelle, l'apprenti possède une expérience acquise sur

un simulateur. Si l'on imagine ceci dans le cadre médical, on permet ainsi à l'apprenti chirurgien d'arriver devant son premier patient avec une certaine expérience. Il faut bien insister sur le fait qu'un simulateur pédagogique n'a pas vocation à se substituer à la formation actuelle mais de la faciliter et de la sécuriser.

De toutes les techniques chirurgicales, certaines sont plus adaptées que d'autres à la formation par simulateur. En effet, toutes les interventions faites directement de main humaine sont difficilement reproductibles. Par contre, toutes celles s'effectuant par l'intermédiaire d'outils et par une visualisation indirecte sont beaucoup moins délicates à mettre en œuvre.

En cela, toutes les techniques se basant sur une visualisation par le biais d'une caméra sont adaptées : il n'y a aucune différence entre la visualisation sur une télévision ou un écran d'ordinateur. D'ailleurs de nombreux simulateurs ont pour objet la reproduction des techniques endoscopiques : on introduit une caméra (sonore ou classique) par un orifice (naturel ou non), on visualise sur un écran ce que la caméra perçoit et on pratique l'intervention par l'intermédiaire d'outils, eux aussi introduits par un orifice (le même ou un autre).

Ces techniques sont très utilisées du fait de leurs avantages pour le patient. Elles sont dite d'invasion minimale car elles ne laissent que la trace minime des perforations d'introduction des différents outils, contrairement à une intervention classique. Les traumatismes conséquents à une opération sont ainsi considérablement diminués.

Du côté des chirurgiens, cela demande un apprentissage spécifique. Actuellement il se fait essentiellement par compagnonnage et/ou par entraînement sur des animaux. Un simulateur remplacerait la phase sur animaux et permettrait de diminuer la part du compagnonnage (cette phase sera toujours indispensable car elle représente la mise en situation réelle). Cette partie étant très coûteuse en temps, cela permettrait d'en gagner un peu sur le temps de formation.

De plus, une session de formation sur animaux ne s'organise pas simplement. Leur nombre est donc limité, sans compter le coût de l'élevage de ces animaux et sans parler des problèmes d'éthique. Le simulateur est lui disponible à tout moment, on peut même imaginer laisser le simulateur en libre service, les étudiants ayant toute latitude quant à la répartition de leurs entraînements. Des procédés d'évaluation automatiques peuvent être incorporés permettant à l'étudiant de juger de ses progrès, voire même d'aider l'enseignant dans l'évaluation qu'il aura à fournir pour valider l'acquis d'un étudiant.

Un autre point important est qu'il est possible de simuler des pathologies. Un étudiant doit attendre d'être confronté à un patient atteint d'une pathologie pour mettre en pratique des procédures dont il n'a qu'une connaissance théorique. Il est clair que les pathologies les plus courantes sont « disponibles » mais que certaines moins fréquentes ne permettent pas un entraînement adéquat. Le simulateur pourra proposer toutes les pathologies que l'enseignant jugera important de connaître.

1.3.2 Historique des simulateurs dans l'équipe GRAPHIX

L'activité sur les simulateurs a été initiée en 1992 par l'étude de la photo coagulation de l'œil par laser. L'ophtalmologiste regarde l'œil du patient dans un binoculaire, manipule à l'aide d'un joystick l'orientation de trois miroirs qui permettent de pouvoir atteindre toute la surface du fond de l'œil et appuie sur une pédale pour effectuer un tir de laser (Figure 1.8). On voit bien ici que la transposition en un simulateur apparaît « naturelle ». Un prototype a été conçu sous le nom de SOPHOCLE [MKCR95] et une version commerciale existe depuis 2000, nommé PixEyes et créé par la société SimEdge (Figure 1.9)

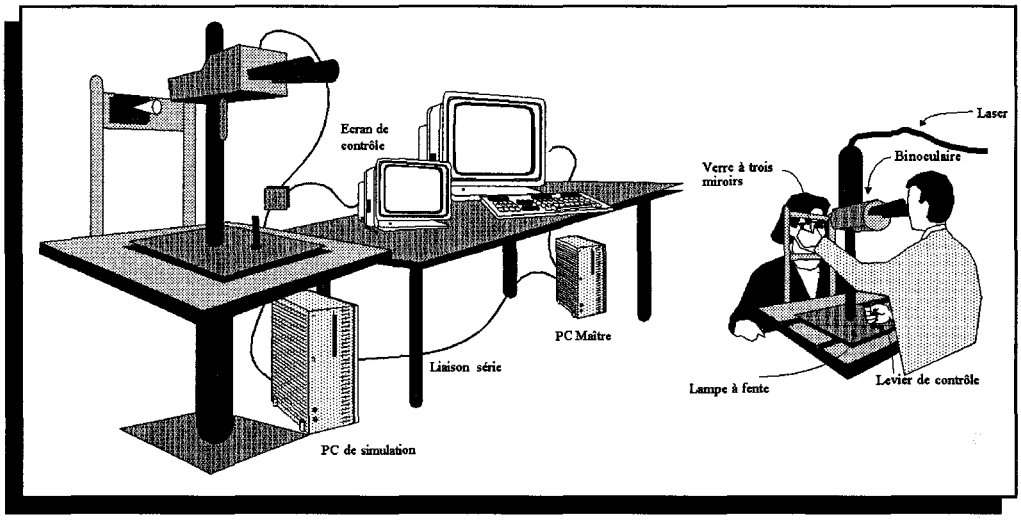


Figure 1.8 : SOPHOCLE

Ce simulateur a subi une phase de test [PDR97] : sur une promotion d'étudiants en ophtalmologie du CHR de Lille, la moitié a suivi la formation traditionnelle, l'autre a utilisé le simulateur. L'évaluateur de l'examen final ne connaissait pas le groupe auquel appartenait chacun des étudiants et n'a pas su différencier les 2 groupes. La population de ce test n'est pas grande mais laisse entendre que le simulateur peut se substituer à la formation traditionnelle.

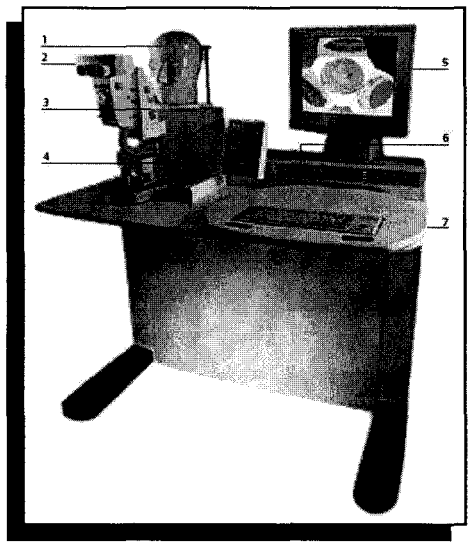


Figure 1.9 : PixEyes

Fort de ce succès, l'équipe GRAPHIX s'est dirigée vers d'autres situations à simuler : l'arthroscopie de l'épaule, echo-endoscopie [VCDM97], amniocentèse [Wib00]. Ces simulateurs ont été la mise en application de recherches sur l'affichage de surfaces implicites [Tri01] ou encore sur le rendu d'images échographiques [Wib00]. Le projet actuel SPIC (Simulateur Pédagogique d'Intervention Cœlioscopique) vise à simuler des interventions chirurgicales, notamment en gynécologie.

La 1ère version de SPIC n'effectuait qu'une visualisation de la cavité abdominale et pouvait servir pour apprendre à se déplacer ou à se repérer. Les chirurgiens qui ont évalué ce prototype ont émis 2 principales critiques : le manque de retour d'effort et le fait que les objets soient statiques. Il convenait donc d'introduire des objets déformables et il fallait que ces déformations soient conformes à la réalité.

1.3.3 Les simulateurs avec utilisation de la mécanique

Pour réaliser cette 2ème version de SPIC, nous utilisons SPORE. Nous avons voulu gardé une certaine généralité dans la construction du simulateur. En effet, d'autres projet de simulateur pédagogique médicaux étaient prévus. Un simulateur générique appelée MISS (Minimally Invasive Surgery Simulation) sert de point de départ. Ce simulateur générique permet de concevoir rapidement le prototype d'un simulateur dédiée. A l'heure actuelle, il est décliné en deux simulateurs : un simulateur d'opthalmologie qui consiste en l'apprentissage du montage d'un hansatome et SPIC. Celui-ci est lui-même spécialisées en deux domaines : la chirurgie gynécologique et viscérale.

La version gynécologique de SPIC est la plus avancée. Elle est complétée par un dispositif à retour d 'effort spécialement conçu pour la manipulation des outils utilisés en cœlioscopie abdominale ([LC99]). SPORE et SPIC sont très liés dans le sens où nous développons en priorité dans SPORE ce dont nous pourrions avoir besoin dans SPIC. Un des apports de SPIC est de valider un modèle introduit dans SPORE et permet de voir son comportement dans des situations d'interactions complexes. La Figure 1.10 montre une séquence de vues de SPIC représentant une intervention complète. Le pavé (solide rigide) représente une adhérence à enlever (1ère image). On l'arrache à la paroi de la cavité abdominale (2ème image) et quand on atteint le point de rupture, une hémorragie se produit (3ème image). Le sang est représenté par des particules reliées par une force de type Lennard-Jones et affichées par une surface implicite. On utilise alors l'outil de cautérisation pour stopper l'hémorragie (4ème image) et enfin l'outil d'aspiration pour évacuer le sang (5ème image).

1.4 BILAN

A travers l'exposé de différents formalismes mécaniques, nous avons montré quel type d'équations il faut traiter lorsque l'on crée une animation basée sur la physique. Il s'agit généralement d'équations différentielles ordinaires du second ordre. On peut envisager un résolution statique qui ramène l'EDO à une équation « normale ». Néanmoins, la suppression de la dynamique n'est utilisable que dans certains problèmes.

Nous avons présenté la bibliothèque SPORE avec laquelle nous construisons des simulateurs pédagogiques médicaux. Son objectif est de simuler des modèles mécaniques variés dans une scène interactive, notamment grâce à l'utilisation de dispositifs à retour d'effort. L'intervention de périphériques manipulés renforce le fait que la simulation se doit d'être temps réel. Nous devons donc effectuer une résolution dynamique qui se doit d'être rapide et fiable numériquement. C'est ce qui a motivé l'étude sur les méthodes de résolutions d'EDO qui est décrite dans le chapitre suivant.

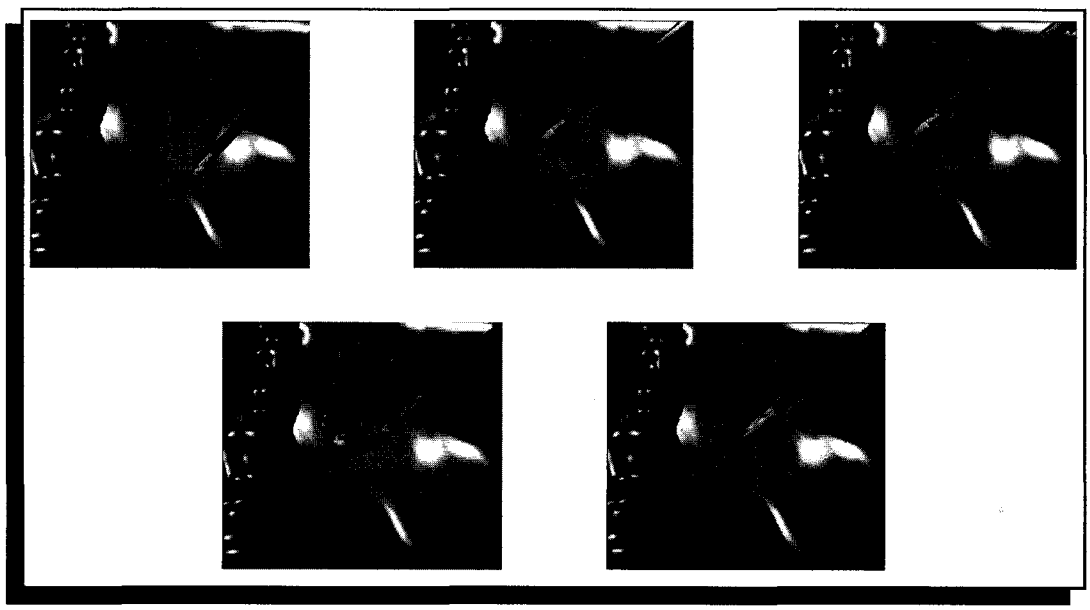


Figure 1.10 : Une intervention dans SPIC

Pour l'instant, nous effectuons nos tests sur une machine bi-processeurs sous Windows NT/2000. Il est assez difficile de pouvoir assurer une réelle synchronicité sur un tel système d'exploitation. Nous pensons qu'il serait souhaitable d'exécuter les boucles de la dynamique et de gestion du retour d'effort sur un système d'exploitation (SE) temps réel tout en laissant la boucle d'affichage sur une machine avec un SE gérant les cartes 3D.

L'architecture matérielle idéale est alors un ordinateur bi-processeur avec un SE temps réel et un ordinateur avec un SE avec interface graphique et gérant les cartes 3D (Figure 1.7). Le débit de la liaison entre les 2 machines est assez élevé mais reste en deçà des limites de liaison classique.

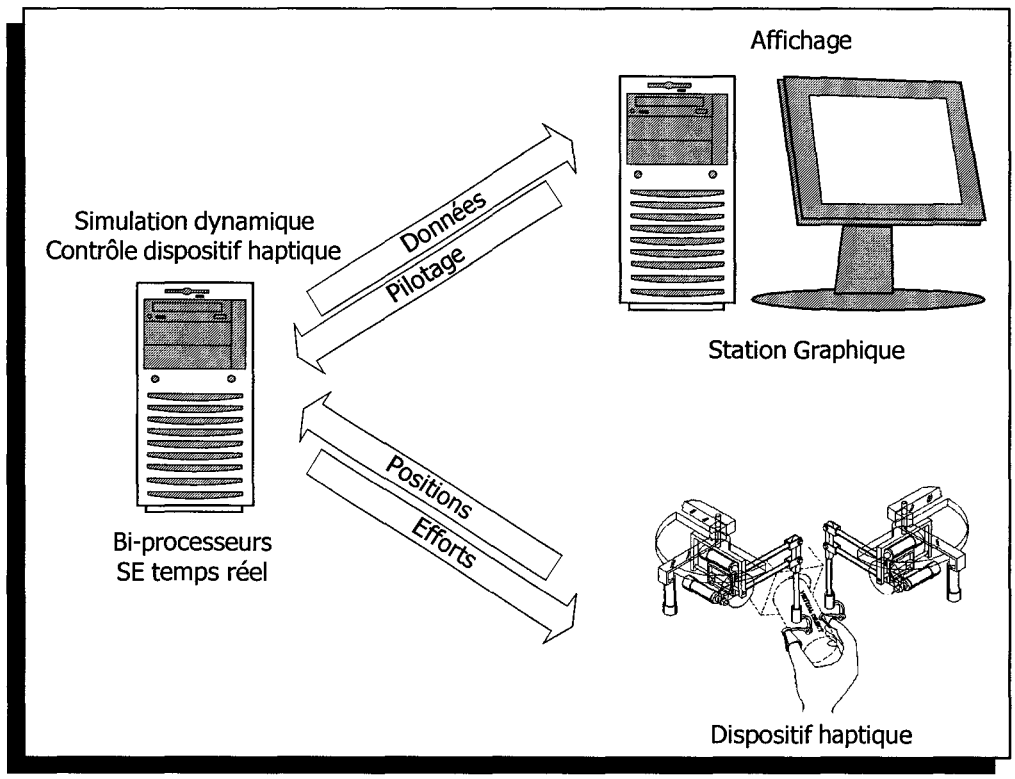


Figure 1.7 : Architecture matérielle souhaitable pour une application utilisant SPORE

1.3 Les simulateurs pédagogiques médicaux

1.3.1 Principe

Le but d'un simulateur pédagogique est de reproduire les conditions réelles d'une manipulation que des « étudiants » doivent apprendre. Ce type de simulateur existe depuis longtemps dans l'aviation et tend à se développer vers d'autres domaines comme la conduite de véhicules spéciaux. Un simulateur comporte de nombreux intérêts pour l'apprentissage de ce type de tâches.

La formation aux techniques de manipulation sur un simulateur permet de réduire les coûts : les erreurs éventuelles ne se solderont que par une catastrophe virtuelle. De plus, lors de la première mise en situation opérationnelle, l'apprenti possède une expérience acquise sur

un simulateur. Si l'on imagine ceci dans le cadre médical, on permet ainsi à l'apprenti chirurgien d'arriver devant son premier patient avec une certaine expérience. Il faut bien insister sur le fait qu'un simulateur pédagogique n'a pas vocation à se substituer à la formation actuelle mais de la faciliter et de la sécuriser.

De toutes les techniques chirurgicales, certaines sont plus adaptées que d'autres à la formation par simulateur. En effet, toutes les interventions faites directement de main humaine sont difficilement reproductibles. Par contre, toutes celles s'effectuant par l'intermédiaire d'outils et par une visualisation indirecte sont beaucoup moins délicates à mettre en œuvre.

En cela, toutes les techniques se basant sur une visualisation par le biais d'une caméra sont adaptées : il n'y a aucune différence entre la visualisation sur une télévision ou un écran d'ordinateur. D'ailleurs de nombreux simulateurs ont pour objet la reproduction des techniques endoscopiques : on introduit une caméra (sonore ou classique) par un orifice (naturel ou non), on visualise sur un écran ce que la caméra perçoit et on pratique l'intervention par l'intermédiaire d'outils, eux aussi introduits par un orifice (le même ou un autre).

Ces techniques sont très utilisées du fait de leurs avantages pour le patient. Elles sont dite d'invasion minimale car elles ne laissent que la trace minime des perforations d'introduction des différents outils, contrairement à une intervention classique. Les traumatismes conséquents à une opération sont ainsi considérablement diminués.

Du côté des chirurgiens, cela demande un apprentissage spécifique. Actuellement il se fait essentiellement par compagnonnage et/ou par entraînement sur des animaux. Un simulateur remplacerait la phase sur animaux et permettrait de diminuer la part du compagnonnage (cette phase sera toujours indispensable car elle représente la mise en situation réelle). Cette partie étant très coûteuse en temps, cela permettrait d'en gagner un peu sur le temps de formation.

De plus, une session de formation sur animaux ne s'organise pas simplement. Leur nombre est donc limité, sans compter le coût de l'élevage de ces animaux et sans parler des problèmes d'éthique. Le simulateur est lui disponible à tout moment, on peut même imaginer laisser le simulateur en libre service, les étudiants ayant toute latitude quant à la répartition de leurs entraînements. Des procédés d'évaluation automatiques peuvent être incorporés permettant à l'étudiant de juger de ses progrès, voire même d'aider l'enseignant dans l'évaluation qu'il aura à fournir pour valider l'acquis d'un étudiant.

Un autre point important est qu'il est possible de simuler des pathologies. Un étudiant doit attendre d'être confronté à un patient atteint d'une pathologie pour mettre en pratique des procédures dont il n'a qu'une connaissance théorique. Il est clair que les pathologies les plus courantes sont « disponibles » mais que certaines moins fréquentes ne permettent pas un entraînement adéquat. Le simulateur pourra proposer toutes les pathologies que l'enseignant jugera important de connaître.

1.3.2 Historique des simulateurs dans l'équipe GRAPHIX

L'activité sur les simulateurs a été initiée en 1992 par l'étude de la photo coagulation de l'œil par laser. L'ophtalmologiste regarde l'œil du patient dans un binoculaire, manipule à l'aide d'un joystick l'orientation de trois miroirs qui permettent de pouvoir atteindre toute la surface du fond de l'œil et appuie sur une pédale pour effectuer un tir de laser (Figure 1.8). On voit bien ici que la transposition en un simulateur apparaît « naturelle ». Un prototype a été conçu sous le nom de SOPHOCLE [MKCR95] et une version commerciale existe depuis 2000, nommé PixEyes et créé par la société SimEdge (Figure 1.9)

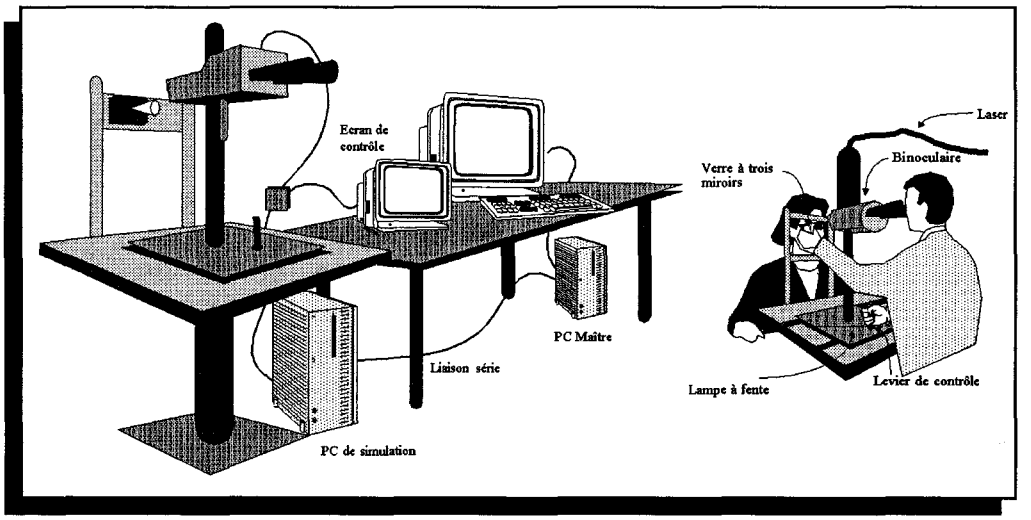


Figure 1.8 : SOPHOCLE

Ce simulateur a subi une phase de test [PDR97] : sur une promotion d'étudiants en ophtalmologie du CHR de Lille, la moitié a suivi la formation traditionnelle, l'autre a utilisé le simulateur. L'évaluateur de l'examen final ne connaissait pas le groupe auquel appartenait chacun des étudiants et n'a pas su différencier les 2 groupes. La population de ce test n'est pas grande mais laisse entendre que le simulateur peut se substituer à la formation traditionnelle.

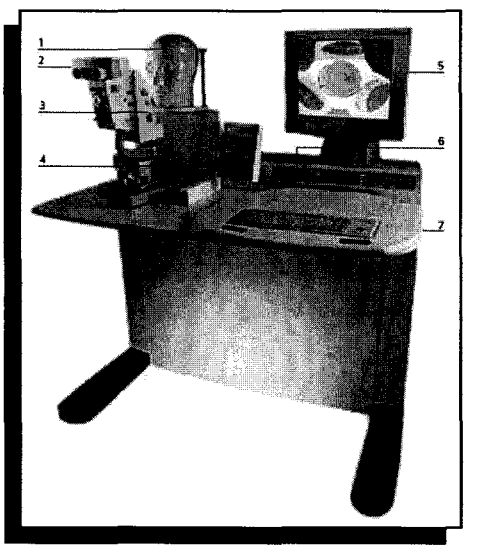


Figure 1.9 : PixEyes

Fort de ce succès, l'équipe GRAPHIX s'est dirigée vers d'autres situations à simuler : l'arthroscopie de l'épaule, echo-endoscopie [VCDM97], amniocentèse [Wib00]. Ces simulateurs ont été la mise en application de recherches sur l'affichage de surfaces implicites [Tri01] ou encore sur le rendu d'images échographiques [Wib00]. Le projet actuel SPIC (Simulateur Pédagogique d'Intervention Cœlioscopique) vise à simuler des interventions chirurgicales, notamment en gynécologie.

La 1ère version de SPIC n'effectuait qu'une visualisation de la cavité abdominale et pouvait servir pour apprendre à se déplacer ou à se repérer. Les chirurgiens qui ont évalué ce prototype ont émis 2 principales critiques : le manque de retour d'effort et le fait que les objets soient statiques. Il convenait donc d'introduire des objets déformables et il fallait que ces déformations soient conformes à la réalité.

1.3.3 Les simulateurs avec utilisation de la mécanique

Pour réaliser cette 2ème version de SPIC, nous utilisons SPORE. Nous avons voulu gardé une certaine généralité dans la construction du simulateur. En effet, d'autres projet de simulateur pédagogique médicaux étaient prévus. Un simulateur générique appelée MISS (Minimally Invasive Surgery Simulation) sert de point de départ. Ce simulateur générique permet de concevoir rapidement le prototype d'un simulateur dédiée. A l'heure actuelle, il est décliné en deux simulateurs : un simulateur d'opthalmologie qui consiste en l'apprentissage du montage d'un hansatome et SPIC. Celui-ci est lui-même spécialisées en deux domaines : la chirurgie gynécologique et viscérale.

La version gynécologique de SPIC est la plus avancée. Elle est complétée par un dispositif à retour d 'effort spécialement conçu pour la manipulation des outils utilisés en cœlioscopie abdominale ([LC99]). SPORE et SPIC sont très liés dans le sens où nous développons en priorité dans SPORE ce dont nous pourrions avoir besoin dans SPIC. Un des apports de SPIC est de valider un modèle introduit dans SPORE et permet de voir son comportement dans des situations d'interactions complexes. La Figure 1.10 montre une séquence de vues de SPIC représentant une intervention complète. Le pavé (solide rigide) représente une adhérence à enlever (1ère image). On l'arrache à la paroi de la cavité abdominale (2ème image) et quand on atteint le point de rupture, une hémorragie se produit (3ème image). Le sang est représenté par des particules reliées par une force de type Lennard-Jones et affichées par une surface implicite. On utilise alors l'outil de cautérisation pour stopper l'hémorragie (4ème image) et enfin l'outil d'aspiration pour évacuer le sang (5ème image).

1.4 BILAN

A travers l'exposé de différents formalismes mécaniques, nous avons montré quel type d'équations il faut traiter lorsque l'on crée une animation basée sur la physique. Il s'agit généralement d'équations différentielles ordinaires du second ordre. On peut envisager un résolution statique qui ramène l'EDO à une équation « normale ». Néanmoins, la suppression de la dynamique n'est utilisable que dans certains problèmes.

Nous avons présenté la bibliothèque SPORE avec laquelle nous construisons des simulateurs pédagogiques médicaux. Son objectif est de simuler des modèles mécaniques variés dans une scène interactive, notamment grâce à l'utilisation de dispositifs à retour d'effort. L'intervention de périphériques manipulés renforce le fait que la simulation se doit d'être temps réel. Nous devons donc effectuer une résolution dynamique qui se doit d'être rapide et fiable numériquement. C'est ce qui a motivé l'étude sur les méthodes de résolutions d'EDO qui est décrite dans le chapitre suivant.

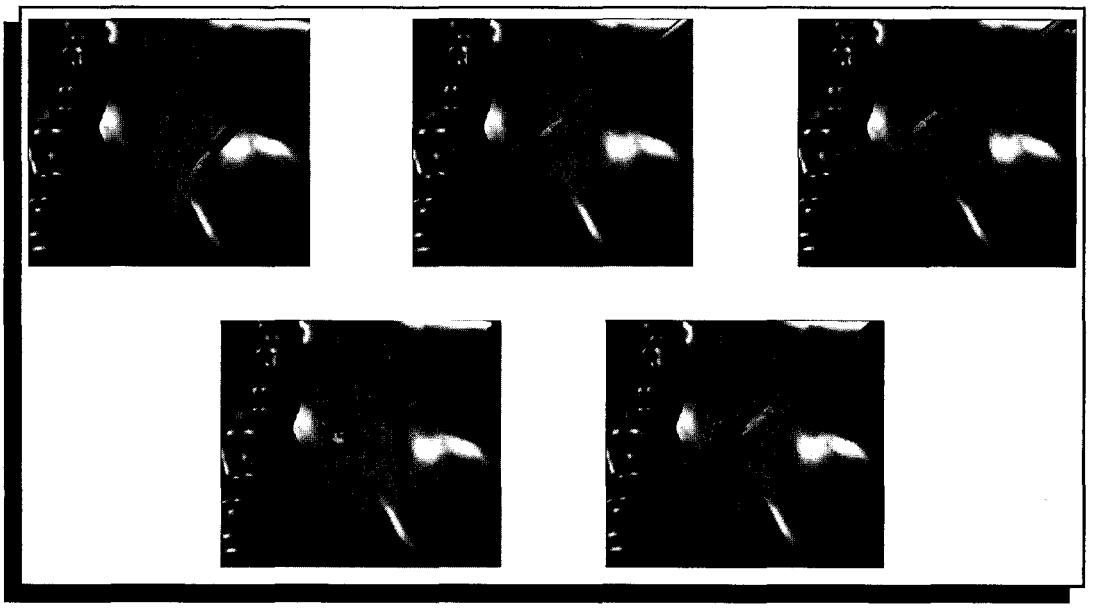


Figure 1.10 : Une intervention dans SPIC

Chapitre 2 : La résolution des équations différentielles ordinaires

2.1 Théorie

2.1.1 Problème de Cauchy

La résolution d'équations différentielles est un problème très ancien. Les 1ères équations différentielles ont animé, au cours du 17ème siècle, les efforts de mathématiciens de l'époque : Euler, Leibniz, les frères Bernoulli, Clairaut,... Chacun cherchait à résoudre les équations différentielles obtenues lors des calculs de la solution à un problème donné. Des méthodes formelles ont été mises au point pour résoudre certaines formes d'équations mais il était clair que certaines resteraient impossibles à résoudre formellement.

Euler, féru de calcul numérique, a le premier imaginé une méthode permettant d'obtenir une suite de valeurs numériques proches de la solution. Beaucoup d'autres lui ont emboîté le pas et diverses méthodes ont été créées. Tous ces calculs ne représentaient toutefois qu'une approximation de la solution et il a fallu attendre Cauchy et le début du 19ème siècle pour que commence une étude théorique des équations différentielles (les conditions d'existence d'une solution et son unicité) et des méthodes numériques imaginées pour calculer leur solution (l'erreur par rapport à la solution, la convergence des séries numériques,...).

Une équation différentielle est une équation qui fait intervenir des dérivées d'une fonction qu'on cherche à calculer. On distingue les équations différentielles ordinaires (EDO) qui ne font intervenir que des dérivations par rapport à une variable et les équations différentielles partielles (EDP) qui font intervenir des dérivations selon plusieurs variables. La résolution des EDP est délicate. En général, on opère une approximation de l'opérateur de différentiation selon certaines variables pour aboutir à une EDO. Dans le chapitre précédent (voir page 7), une EDP est ramené à une EDO temporelle grâce à une discrétisation spatiale. Enfin, on appelle l'ordre d'une équations différentielles l'ordre maximal de dérivation qui apparaît dans l'équation.

Une EDO du premier ordre est donc une équation de la forme[2] :

$$\dot{x} = f(t, x) \tag{2.1}$$

La fonction $x(t)$ est une solution de (2.1) si $\forall t \in I \subset \mathbb{R}$, $\dot{x}(t) = f(t, x(t))$. Il a été très tôt remarqué (Newton, Leibniz et Euler) que la solution avait un paramètre libre et qu'il fallait ajouter une condition initiale :

$$x(t_0) = x_0 \tag{2.2}$$

(2.1) et (2.2) forment ce qu'on appelle le problème de Cauchy ou problème aux conditions initiales. Ce problème correspond à ceux posés par la mécanique : on a une équation décrivant le comportement d'un corps, on connaît les conditions à l'instant t_0 et on veut calculer

[2] On ne s'intéresse pas au problème différentiel général $F(t, x(t), \dot{x}(t)) = 0$. Il est tout de même parfois possible de se ramener à un problème de Cauchy, au moins localement.

l'évolution de ce corps en fonction du temps. On peut imaginer d'autres problèmes en fixant une valeur de la fonction x pour une valeur de t quelconque dans I. Par exemple, on pourrait imaginer un problème de mécanique pour lequel, sachant la position d'arrivée d'un objet on voudrait calculer la trajectoire empruntée.

Les équations de la mécanique font intervenir des EDO d'ordre 2

$$\ddot{x} = f(t, x, \dot{x}) \tag{2.3}$$

on peut ramener cette équation à une EDO d'ordre 1 de la façon suivante

$$\begin{cases} \dot{x} = v \\ \dot{v} = f(t, x, v) \end{cases} \tag{2.4}$$

On a donc un système d'EDO du premier ordre. Les conditions initiales associées sont $x(t_0) = x_0$ et $v(t_0) = v_0$. Dans la suite, quand on évoquera une EDO du 1er ordre (2.1), x représentera un vecteur de $\mathbb{R}^n$ et f une fonction $I \times \mathbb{R}^n \mapsto \mathbb{R}^n$.

Une condition suffisante pour que (2.1) ait une solution unique est que f soit continue sur $I \times \mathbb{R}^n$ et qu'il existe $L \in \mathbb{R}^n$ tel que :

$$\forall (t, x_1) et (t, x_2) \in I \times \mathbb{R}^n, \quad \left| f(t, x_1) - f(t, x_2) \right| \leq L \left| x_1 - x_2 \right| \tag{2.5}$$

Cette condition est appelée condition de Cauchy-Lipschitz. Cette condition est déduite de théorèmes moins contraignants (voir [CM89]). Il est à noter que c'est une condition suffisante et non nécessaire. Il existe des cas où la fonction f n'est pas continue alors qu'il existe une solution au problème.

2.1.2 Résolution numérique

Pour résoudre numériquement une EDO, on va estimer la valeur de la fonction solution en certains points t_i. On va donc se fixer une suite de valeur (t_i) et on calcule la suite de valeurs (x_i) tel que x_i soit une approximation de $x(t_i)$. On définit la suite (h_i) telle que $h_i = t_{i+1} - t_i$ et $h = \max h_i$. h_i représente un pas d'intégration (on utilise souvent le terme de pas de temps quand il représente une durée). Le plus simple est de considérer un pas d'intégration fixe, i.e. une répartition équidistante des (t_i). Nous reviendrons plus tard sur l'utilité d'un pas de temps variable. On notera f_i la valeur de $f(t_i, x_i)$.

Il s'agit maintenant de définir la façon dont on détermine x_{n+1} en fonction des valeurs précédemment calculées. On distingue les méthodes numériques en 2 grandes catégories : les méthodes à un pas et les méthodes multipas

Les méthodes à un pas se basent sur la formulation suivante :

$$x_{n+1} = x_n + h_n \Phi\left(t_n, x_n, x_{n+1}\right) \tag{2.6}$$

C'est Φ qui définit la méthode. Comme le nom et la formule l'indique, elle n'utilise que les informations concernant le pas courant. Leur stratégie est de calculer des valeurs intermédiaires dans l'intervalle $[t_i, t_{i+1}]$[3]. Elles sont décrites dans la section 2.2.2.

Les méthodes multipas se basent sur la formulation suivante :

$$\alpha_0 x_{n+1} + \alpha_1 x_n + \ldots + \alpha_k x_{n+1-k} = h_n\left(\beta_0 f_{n+1} + \ldots + \beta_k f_{n+1-k}\right) \tag{2.7}$$

avec $\alpha_0 \neq 0$. Ici, la valeur inconnue x_{n+1} est calculée à partir des valeurs des k pas précédents. Elles seront détaillées dans la section 2.2.3.

Cette division en deux catégories vient essentiellement du fait que les études théoriques doivent être souvent menées différemment. De manière générale, les principes de bases sont communs mais les formulations sont différentes. Nous allons maintenant exposer les propriétés fondamentales que doit posséder une méthode de résolution numérique.

Ces propriétés se basent sur le fait qu'il faut que l'utilisation d'une telle méthode présente le moins d'erreur possible par rapport à la vraie solution. Cette erreur a deux sources : l'erreur de discrétisation, uniquement dépendante de la méthode employée, et l'erreur d'arrondi due à la troncature des chiffres décimaux. Cette prise en compte de l'arrondi existait bien avant l'utilisation de l'ordinateur car les calculs, étant auparavant faits manuellement, devaient être arrondis pour obtenir des temps de calculs raisonnables.

La convergence

Cette propriété découle du fait que l'on désire que l'erreur de discrétisation diminue lorsque h diminue. Elle s'exprime par la formulation suivante :

$$\lim_{h \to 0} \max_n \left| x\left(t_n\right) - x_n \right| = 0 \tag{2.8}$$

Cela garantit que la méthode peut trouver une bonne approximation de la solution, à la condition de maintenir h suffisamment petit.

La stabilité

Il est important de savoir quelle va être la répercussion sur les pas suivants de l'erreur commise sur un pas. La stabilité représente donc une façon de contrôler le cumul des erreurs. Sa formulation est la suivante : soit (x_n) une suite numérique associée à une méthode

[3] Ces méthodes sont parfois appelées méthodes à étapes internes et les méthodes multipas sont dites à étapes externes

d'intégration, (z_n) une suite basée sur la même méthode mais perturbée par la suite (ε_n). Pour une méthode à un pas, cela s'écrit :

$$\begin{aligned} x_{n+1} &= x_n + h_n \Phi(t_n, x_n) \\ z_{n+1} &= z_n + h_n \Phi(t_n, z_n) + \varepsilon_n \end{aligned} \tag{2.9}$$

On dit que la méthode est stable s'il existe une constante M, indépendante de h, telle que :

$$\max_n |z_n - x_n| \leq M\left[|x_0 - z_0| + \sum |\varepsilon_n|\right] \tag{2.10}$$

Cela représente une notion de stabilité proche de celle des positions d'équilibres en physique : une petite perturbation sur les données n'entraîne qu'une perturbation du même ordre de grandeur sur la solution. Ceci est très important car cela permet de savoir que les erreurs d'arrondis n'auront pas des perturbations qui s'amplifieront outre mesure.

La consistance

Enfin, il est important de vérifier que la méthode permette d'approximer correctement la solution et mesurer l'erreur commise par rapport à la vraie solution. Pour cela, on définit l'erreur de consistance : elle cherche à mesurer l'erreur faite au $n^{ème}$ pas en remplaçant l'équation différentielle par le schéma d'intégration. Cette erreur s'écrit pour une méthode à un pas :

$$e_n = x(t_{n+1}) - x(t_n) - h_n \Phi(t_n, x(t_n)) \tag{2.11}$$

On dit que la méthode est consistante si :

$$\lim_{h \to 0} \sum_n |e_n| = 0 \tag{2.12}$$

Ainsi, en maintenant h assez petit, on fait en sorte que le cumul de ces erreurs reste faible.

Un théorème lie ces trois propriétés : si la méthode est stable et consistante, alors elle est convergente. Il faut préciser que ces propriétés sont des conditions nécessaires pour qu'un schéma numérique soit utilisable.

Une caractéristique importante d'une méthode est son ordre de consistance : il s'agit en fait de caractériser la façon dont la somme de (2.12) tend vers zéro. On dit que la méthode est d'ordre p s'il existe un réel K tel que :

$$\sum_n |e_n| \leq K h^p \tag{2.13}$$

Grâce à certaines propriétés, on peut lier cette définition de l'ordre à l'erreur de discrétisation. Si la méthode est d'ordre p, $|x(t_n) - x_n| = O(h^p)$. On voit que plus l'ordre d'une mé-

thode est élevée, plus l'erreur de discrétisation sera faible. C'est pourquoi il apparaît intéressant de vouloir utiliser des méthodes qui possèdent un ordre élevé.

2.2 Exemples de méthodes

Pour toutes les méthodes décrites ci-après, il n'y aura pas d'études sur les propriétés précédentes : elles sont toutes vérifiées et on pourra consulter [CM89], [HW00] et [HW96] pour des démonstrations. Par contre, l'ordre de consistance sera mentionné car il donne un indicateur sur la précision de la méthode.

2.2.1 Les méthodes d'Euler

La méthode imaginée par Euler se base sur une approximation de la dérivée de la fonction. On utilise la formule suivante :

$$\dot{x}(t_n) = f(t_n, x_n) \approx \frac{x_{n+1} - x_n}{h_n} \quad \Longrightarrow \quad \boxed{x_{n+1} = x_n + h_n f(t_n, x_n)} \tag{2.14}$$

Cela revient à approximer la solution par une succession de droites. La pente de ces droites est une approximation de la pente en $x(t_n)$. En suivant le même principe on peut utiliser la formulation suivante (imaginée par Cauchy) :

$$\dot{x}(t_{n+1}) = f(t_{n+1}, x_{n+1}) \approx \frac{x_{n+1} - x_n}{h_n} \quad \Longrightarrow \quad \boxed{x_{n+1} = x_n + h_n f(t_{n+1}, x_{n+1})} \tag{2.15}$$

La différence entre les deux formulations tient dans l'approximation de la dérivation. Il existe une dénomination plus générale de ces deux méthodes. La première est dite explicite, i.e. la valeur au point se calcule directement à partir de données existantes ; la seconde est dite implicite, i.e. la valeur au point n'est connu que par une équation qu'il faut résoudre. On est alors en présence d'un système d'équations non-linéaires.

On constate donc qu'il existe une grosse différence de coût de calcul entre une méthode explicite et une méthode implicite. La Figure 2.1 montre un exemple de résolution de l'équation $\dot{x} = -x$ pour $x(0) = -1$ avec un pas d'intégration fixe $h = 0,5$ avec les 2 méthodes. On voit que les 2 méthodes ont un comportement différent : Euler explicite a tendance à suivre un peu trop loin la pente locale, tandis qu'Euler implicite « freine un peu trop ». Les 2 méthodes sont d'ordre 1 et d'ailleurs on peut constater sur la figure que l'erreur commise par les 2 méthodes est du même ordre de grandeur.

Néanmoins, la notion d'ordre de consistance ne donne qu'une indication sur l'erreur commise. Celle-ci dépend de la longueur du pas d'intégration utilisée. La Figure 2.2 montre la résolution du problème $\dot{x} = (\cot(t) - 2at)x$ dont la solution est $x = \sin(t)\mathrm{e}^{-at^2}$ avec $a = 0.05$ avec la méthode d'Euler explicite pour différent pas de temps. On constate que chacune des courbes approxime la solution plus ou moins bien.

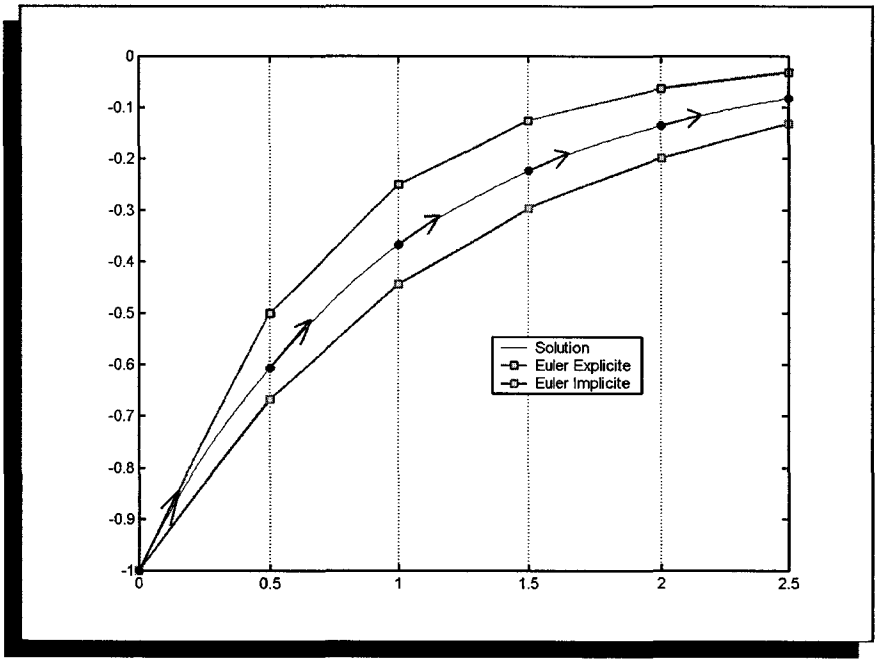


Figure 2.1 : Exemple de résolution du problème $\dot{x} = -x$ avec $x(0) = -1$

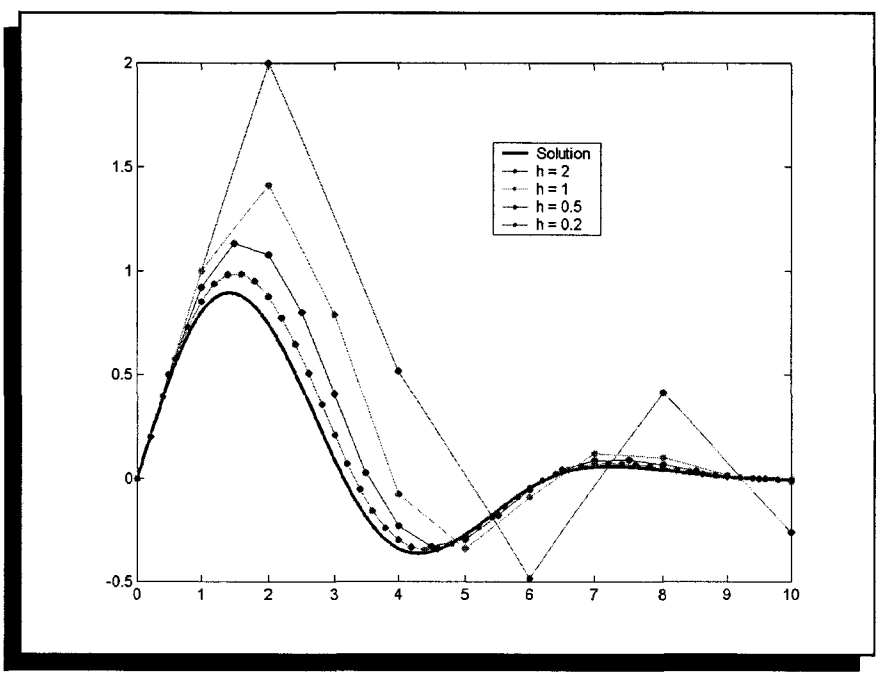


Figure 2.2 : Résolution du problème $\dot{x} = (\cot(t) - 2at)x$ en utilisant Euler explicite avec différents pas de d'intégration.

On peut remarquer aussi que la précision va dépendre de l'allure de la fonction à approcher. Au début, la fonction subit des variations assez importantes et c'est là que se fait sentir le besoin d'un pas d'intégration petit : cela représente le fait qu'il faut pouvoir changer de direction assez souvent pour suivre la courbe. De manière générale, lorsque l'on se fixe une erreur à ne pas dépasser, il faut déterminer le pas d'intégration permettant de réaliser cette condition. Ce pas dépendra de l'ordre de la méthode : on peut dire qu'une méthode avec un ordre plus élevé ne nécessitera pas un pas aussi petit.

Par contre, à partir de $t = 5$, on ne constate plus de grosses différences entre les trois pas les plus petits. Il apparaît alors qu'en utilisant un petit pas, on effectue beaucoup de calcul pour peu ou pas de gain en précision. Il suffirait d'adapter le pas de temps suivant les variations de la fonction suivant la précision désirée. C'est l'intérêt d'utiliser un pas d'intégration variable : adapter la taille du pas pour garder une précision donnée tout en essayant de faire le moins de calcul possible. Il reste à trouver un moyen de savoir quel est le pas optimal à chaque position à calculer. Nous reviendrons sur cette question dans la section suivante.

Ces remarques sur l'influence du pas de temps sur l'erreur de la solution obtenue par rapport à la solution réelle et sur l'intérêt de l'adaptation sont applicables à toutes les méthodes de résolution d'EDO.

Une variante de la méthode d'Euler est très souvent utilisée dans le cadre de résolution d'EDO d'ordre 2 comme celle de la mécanique du point. Suite à la transformation de l'équation en une EDO d'ordre 1 (équation (2.4)) et à l'application du schéma d'Euler on obtient la suite numérique suivante :

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix} + h \begin{pmatrix} v_n \\ f(t_n, x_n, v_n) \end{pmatrix} \tag{2.16}$$

Les calculs de x_{n+1} et de v_{n+1} étant indépendants, ils peuvent être calculé dans n'importe quel ordre. La variante utilise ce fait en calculant v_{n+1} et se servant de cette valeur pour le calcul de x_{n+1} ; ce qui donne le schéma suivant :

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix} + h \begin{pmatrix} v_{n+1} \\ f(t_n, x_n, v_n) \end{pmatrix} \tag{2.17}$$

Ce schéma a été utilisé pour la 1ère fois par Euler lui-même lors d'une étude, réalisée en 1774, sur les conséquences du passage d'une comète. Un de ses atouts est qu'il ne modifie en aucun cas la complexité du calcul mais que l'ordre de consistance est 2. Sur la Figure 2.3 est tracée la résolution d'une EDO d'ordre 2 dont la solution est la même que celle de la Figure 2.2. Pour un pas de temps égal à 0.2, alors qu'Euler donne une solution assez éloignée, celle de la variante d'Euler est quasiment superposée à la courbe solution. Les autres courbes montrent qu'avec des pas de temps plus grands (0.5 et 1), on obtient avec la variante d'Euler des courbes qui restent plus précises qu'Euler avec un pas plus petit.

2.2.2 Les méthodes de Runge-Kutta

Nous avons vu dans la section précédente que l'ordre de consistance est important car cela permet d'utiliser un pas de d'intégration plus grand pour atteindre une précision donnée. L'intérêt est de diminuer les calculs et aussi de parer au problème de troncature de la représentation des nombres flottants dans un ordinateur. En effet, celle-ci fixe une précision maximale que l'on peut atteindre et il est donc illusoire de croire que l'on peut diminuer le pas de temps autant que l'on veut.

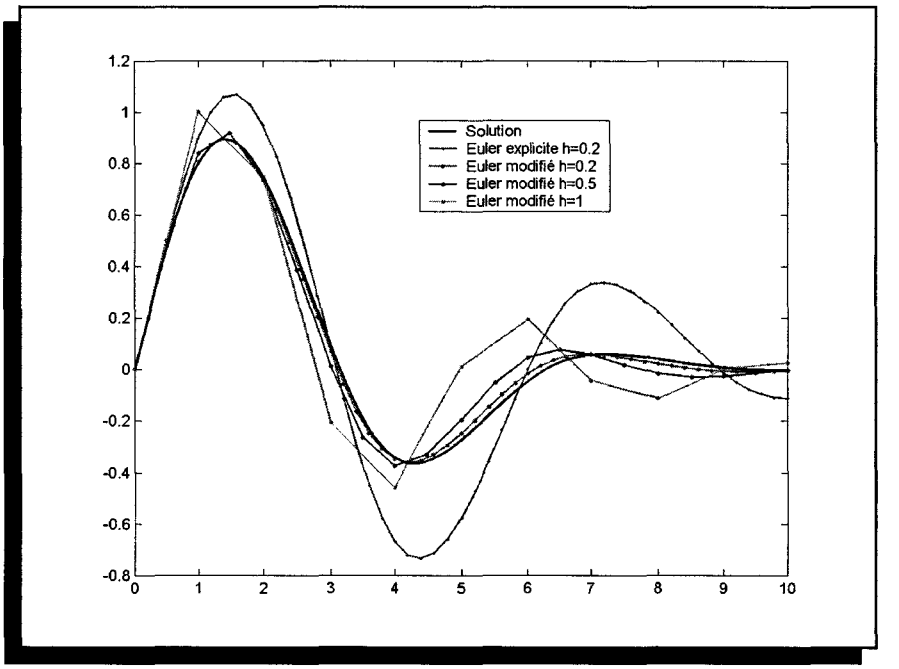


Figure 2.3 : Résolution d'une EDO d'ordre 2avec la variante d'Euler

Runge (1895) a eu l'idée de copier un mécanisme utilisé pour le calcul de l'intégrale d'une fonction. L'approximation du point milieu consiste à utiliser la valeur de la fonction au milieu du segment comme le montre la Figure 2.4. Transposée dans le cadre de la résolution d'une EDO, cela donne le schéma suivant :

$$x_{n+1} = x_n + hf\left(t_n + \frac{h}{2}, x\left(t_n + \frac{h}{2}\right)\right) \tag{2.18}$$

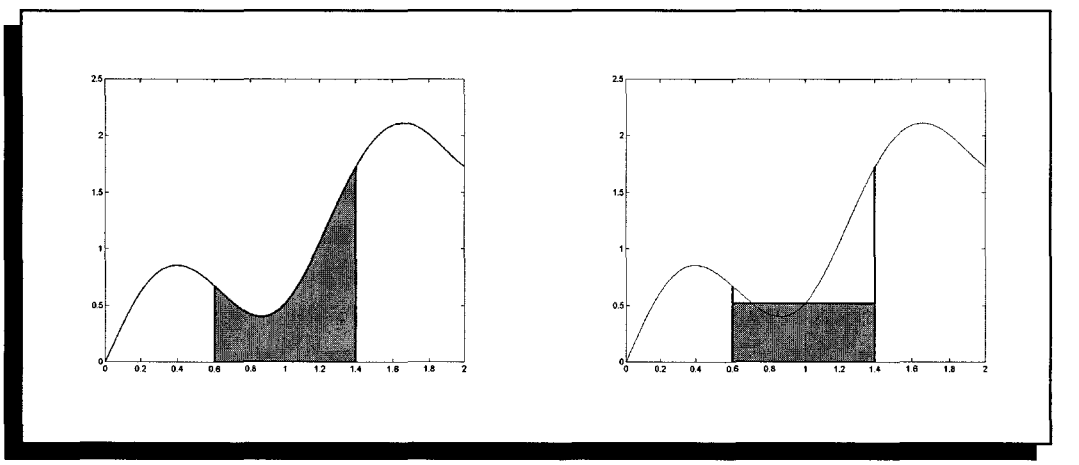


Figure 2.4 : À gauche, l'aire sous la courbe représente un calcul intégral ; à droite, ce calcul est approché par la valeur au point milieu de la fonction.

Il restait à déterminer comment calculer $x\left(t_n + h/2\right)$. Runge eut l'idée supplémentaire d'approcher cette valeur par un pas d'Euler. Au final, on obtient le schéma suivant :

$$
\begin{aligned}
k_1 &= f(t_n, x_n) \\
k_2 &= f(t_n + \frac{h}{2}, x_n + \frac{h}{2} k_1) \\
x_{n+1} &= x_n + h k_2
\end{aligned}
\tag{2.19}
$$

La caractéristique de ce schéma est qu'il est d'ordre 2. Runge puis Heun (1900) ont proposé des méthodes basées sur le même principe d'évaluations intermédiaires pour construire des méthodes d'ordre 3. Enfin, Kutta (1901) a donné un cadre général à ce qui est appelé les méthodes de Runge-Kutta. La formulation donnée ici est la formulation moderne basée sur la représentation de la méthode par le tableau suivant ([But64]) :

$$
\begin{array}{c|ccc}
c_1 & a_{11} & \dots & a_{1s} \\
\vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & \dots & a_{ss} \\
\hline
 & b_1 & \dots & b_s
\end{array}
\tag{2.20}
$$

Le schéma d'intégration est alors défini par les formules :

$$
\begin{aligned}
k_i &= f\left(t_n + c_i h, x_n + h \sum_{j=1}^{s} a_{ij} k_j \right) \quad i \in [\![1, s]\!] \\
x_{n+1} &= x_n + h \sum_{j=1}^{s} b_j k_j
\end{aligned}
\tag{2.21}
$$

Les coefficients c_i représentent les positions intermédiaires où vont être effectuées des évaluations de la solution. Les k_i représentent une évaluation de la pente aux positions intermédiaires ; les coefficients a_{ij} permettent de contrôler cette approximation de la même manière qu'une formule de quadrature approche le calcul d'une intégrale [CM89]. Enfin, les b_i contrôlent l'approximation finale. L'entier s est appelé nombre d'étages de la méthode.

Si la matrice (a_{ij}) est strictement triangulaire inférieure (diagonale nulle), la méthode est explicite (RKE) ; le coût de la méthode dépend uniquement du nombre d'étages s : il faudra effectuer s évaluations de la fonction. Si la matrice (a_{ij}) est triangulaire inférieure, elle est dite semi-implicite (RKSI) ; Il faut alors résoudre un système d'équations non linéaires pour chaque k_i. Dans le cas général, la méthode est dite implicite (RKI) et il faut alors résoudre un système d'équations non linéaires englobant tous les k_i. Des exemples de méthodes RKE se trouvent dans la Table 2.1 et des exemples de méthodes RKI dans la Table 2.2.

Une grande attention a été portée sur la création de méthodes RKE d'ordre élevé. À partir de la définition de l'ordre d'une méthode, on peut expliciter des conditions sur les coefficients d'une méthode RKE : ce sont des relations algébriques. Le gros problème est que le nombre et la complexité de ces conditions augmentent très rapidement avec l'ordre. De plus, il est important de faire en sorte que le nombre d'étages soit minimal.

0	0
	1

Euler, ordre 1

0	0	0
1/2	1/2	0
	0	1

Point milieu, ordre 2

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	2/6	2/6	1/6

Runge-Kutta, ordre 4

Table 2.1 : Exemples de méthodes RKE

1	1
	1

Euler, ordre 1

1/2	1/2
	1

Point milieu, ordre 2

$\frac{1}{2}-\frac{\sqrt{3}}{6}$	$\frac{1}{4}$	$\frac{1}{4}-\frac{\sqrt{3}}{6}$
$\frac{1}{2}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}+\frac{\sqrt{3}}{6}$	$\frac{1}{4}$
	1/2	1/2

Hammer & Hollingsworth, ordre 4

Table 2.2 : Exemples de méthodes RKI

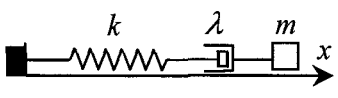
En ce qui concerne le nombre d'étages, Butcher a établi différents résultats :

- Pour une méthode d'ordre $p \geq 5$, il faut au moins $p+1$ étages [But64].
- Pour une méthode d'ordre $p = 7$, il faut au moins $p+2$ étages [But65].
- Pour une méthode d'ordre $p \geq 8$, il faut au moins $p+3$ étages [But85].

Ce sont les seules propriétés théoriques liant le nombre d'étages à l'ordre. Actuellement, la méthode RKE d'ordre le plus élevé est d'ordre 10 avec 17 étages [Hai78]. La méthode la plus employée est la méthode dite RK4 (Runge-Kutta d'ordre 4) présentée dans la Table 2.1 pour son bon rapport précision/coût de calcul. En effet, l'ordre 4 est une charnière : au-delà, le nombre d'étages devient plus grand que l'ordre. Ce nombre d'étages conditionne le coût en calcul de la méthode : une méthode à s étages implique s évaluations de la fonction f, laquelle est en général très complexe ; par exemple, dans le cas d'une simulation mécanique, elle représente le bilan des forces.

Les méthodes RKI ont moins de restrictions en ce qui concerne le nombre d'étages pour un ordre donné (voir la méthode à 2 étages et d'ordre 4 de la Table 2.2). Mais il faut tenir compte des coûts en calcul qui sont beaucoup plus importants. À cause de cela, les méthodes implicites ne sont utilisées que lorsqu'elles apportent autre chose que la seule précision. C'est ce que nous verrons dans la section 2.3.

La Figure 2.5 montre l'évolution de la position d'un système oscillateur amorti. Une masse accrochée à un ressort amorti est lâchée d'une position d'étirement pour le ressort.



Les paramètres physiques utilisés pour l'exemple sont $k = 9Nm^{-1}$, $\lambda = 1.5Nm^{-1}s$ et $m = 1kg$. Les courbes permettent une comparaison d'Euler et de RK4 pour un pas identique égal à 0.1 et pour un coût de calcul identique (pas de 0.1 pour Euler et pas de 0.4 pour RK4) ;

ce coût est mesuré en nombre d'évaluation de la fonction f. En effet c'est elle qui porte la complexité des calculs, une méthode explicite n'ajoutant que des opérations simples et en quantité limitée.

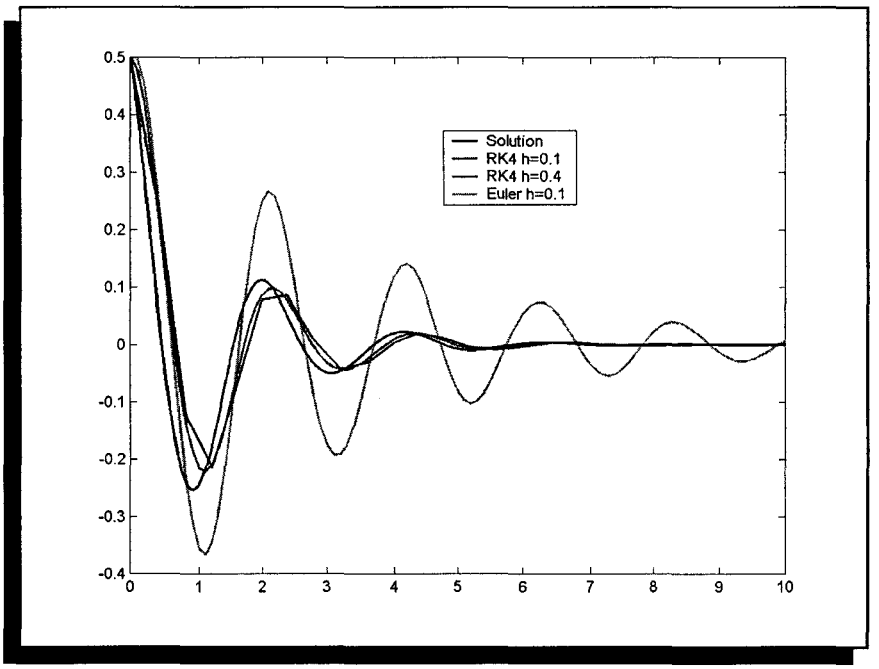


Figure 2.5 : Oscillateur amorti, comparaison Euler RK4

Contrôle du pas

Dans la section 2.2.1, nous avons évoqué l'idée d'adapter le pas d'intégration en fonction des besoins de précision. À cette fin, il faut un moyen d'évaluer l'erreur commise par rapport à la solution. Une manière simple de procéder à cette évaluation est connue sous le nom d'extrapolation de Richardson. Elle consiste en le calcul de x_1 et x_2, valeurs calculées pour 2 pas d'intégration de longueur h à partir de x_0. On considère aussi x^* valeur calculée pour un pas d'intégration de longueur $2h$ à partir de x_0.

En supposant que l'on utilise une méthode RK d'ordre p, l'erreur e_1 commise sur x_1 est égale à :

$$e_1 = x(t_0 + h) - x_1 = Ch^{p+1} + O(h^{p+2}) \tag{2.22}$$

où C est un terme d'erreur dépendant de la méthode et des dérivées de f. L'erreur e_2 sur x_2 se compose d'une erreur locale semblable à e_1 et du report de e_1. On peut écrire :

$$e_2 = x(t_0 + 2h) - x_2 = 2Ch^{p+1} + O(h^{p+2}) \tag{2.23}$$

Enfin, l'erreur commise pour le calcul du pas $2h$:

$$x(t_0 + 2h) - x^* = C(2h)^{p+1} + O(h^{p+2}) \tag{2.24}$$

En utilisant (2.23) et (2.24), on peut aboutir à la relation suivante :

$$x(t_0+2h)-x_2=\frac{x_2-x^*}{2^p-1}+O(h^{p+2}) \tag{2.25}$$

Ainsi, on peut en déduire une approximation de $x(t_0+2h)$ à l'ordre $p+1$

$$\hat{x}_2=x_2+\frac{x_2-x^*}{2^p-1} \tag{2.26}$$

Cette technique permet d'obtenir une solution plus précise mais surtout une évaluation de l'erreur commise par rapport à la solution. L'inconvénient est qu'il faut faire des calculs supplémentaires pour le grand pas (ou pour les 2 petits). Merson (1957) eut l'idée de construire des méthodes RK de telles façons qu'elles puissent générer les valeurs x et $\hat{x}$ sans évaluations supplémentaires de f.

Pour cela, on couple deux méthodes RK en imposant que les valeurs a_{ij} et c_i soient identiques, la différentiation se fait par les coefficients b_i. Cela s'appelle des méthodes emboitées ; on associe à une telle méthode l'ordre des 2 sous-méthodes. La Table 2.3 propose 2 méthodes emboîtées. La construction de méthodes emboîtées avec le même nombre d'étages devient problématique avec l'augmentation de l'ordre ; une parade est d'utiliser les valeurs b_i de x comme étage supplémentaire pour la méthode produisant $\hat{x}$. C'est le cas de la méthode Dormand-Prince 5(4).

0					
$\frac{1}{3}$	$\frac{1}{3}$				
$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$			
$\frac{1}{2}$	$\frac{1}{8}$	0	$\frac{3}{8}$		
1	$\frac{1}{2}$	0	$-\frac{3}{2}$	2	
x	$\frac{1}{6}$	0	0	$\frac{2}{3}$	$\frac{1}{6}$
$\hat{x}$	$\frac{1}{10}$	0	$\frac{3}{10}$	$\frac{2}{5}$	$\frac{1}{5}$

Merson 4 (3)

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$						
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$					
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
x	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
$\hat{x}$	$\frac{5179}{57600}$	0	$\frac{7571}{16695}$	$\frac{393}{640}$	$-\frac{92097}{339200}$	$\frac{187}{2100}$	$\frac{1}{40}$

Dormand-Prince 5(4)

Table 2.3 : Exemples de méthodes emboîtées

L'étape suivante est d'utiliser les deux évaluations x et $\hat{x}$ afin de déterminer le pas d'intégration optimal (le plus grand) afin de satisfaire une précision donnée. En effet, la différence est une bonne mesure de l'erreur commise et en se fixant une certaine tolérance *Tol*, on définit la valeur *Err* telle que :

$$Err = \|v\| \text{ avec } v \text{ tel que } v_i = \frac{x_i - \hat{x}_i}{Tol_i} \tag{2.27}$$

Il suffit de comparer *Err* à 1 : si elle est inférieure à 1, on conserve le pas, sinon on le rejette. Dans les 2 cas, on calcule la valeur du pas suivant : on sait que $Err \approx Ch^{p+1}$ et que *Err* vaut 1 pour le pas optimal h_{opt}. Ce dernier est alors estimé avec la formule :

$$h_{opt} = h\left(\frac{1}{Err}\right)^{\frac{1}{p+1}} \tag{2.28}$$

Dans la pratique, il est courant d'insérer des gardes-fous pour empêcher le pas d'augmenter ou de diminuer trop rapidement. Cette technique permet ainsi d'adapter le pas de temps relativement facilement.

2.2.3 Les méthodes multipas

Avec les méthodes RK, on essaie de gagner en précision grâce à des étapes intermédiaires. Celles-ci alourdissent les calculs en augmentant le nombre d'évaluations de f pour les méthodes explicites, en augmentant la taille du système non-linéaire à résoudre pour les méthodes implicites. Les méthodes multipas utilisent les valeurs des pas précédents pour essayer de mieux prévoir la valeur à calculer.

Les méthodes multipas ont été largement utilisées pour résoudre des problèmes différentiels durant la première partie du 20ème siècle. Celles-ci présentaient l'énorme avantage par rapport aux méthodes RK de faire en sorte que les calculs soient réduits grâce à la réutilisation de calculs précédents. Elles différent aussi des méthodes RK par le fait qu'elles trouvent toutes leurs origines dans la résolution d'un problème concret : forme d'une goutte d'eau pour Bashforth, mécanique céleste pour Darwin,... On pourra trouver un historique des méthodes multipas dans [Tou98].

Les calculateurs électroniques n'existant pas à cette époque, ces méthodes étaient attrayantes du fait qu'il était possible de construire des tables à partir desquelles on pouvait effectuer la résolution de son problème. Et c'est justement l'apparition des premiers ordinateurs qui mit fin au règne des méthodes multipas : elles demandaient trop de stockage. Comme les opérateurs de calculs devenaient par contre plus rapides, les numériciens se sont tournés vers les méthodes RK [Tou98].

Une méthode multipas est définie par la donnée des coefficients (α_i) et (β_i) de l'équation (2.7) :

$$\alpha_0 x_{n+1} + \alpha_1 x_n + \ldots + \alpha_k x_{n+1-k} = h_n \left(\beta_0 f_{n+1} + \ldots + \beta_k f_{n+1-k}\right)$$

Ceux-ci doivent respecter les conditions de convergences définies dans la section 2.1.2. Dans la conception de ce genre de méthodes, on choisit soit de jouer sur les coefficients (α_i) (états

précédents), soit sur les coefficients (β_i) (évaluation de f aux états précédents). Nous allons voir ces 2 cas au travers des méthodes d'Adams et des méthodes BDF.

2.2.3.1 Les méthodes d'Adams

J.C. Adams a imaginé ces méthodes suite à une demande de F. Bashforth qui cherchait à vérifier les modèles théoriques de l'action capillaire. Le principe général des méthodes d'Adams est d'utiliser un polynôme pour approximer f. On peut réécrire le problème de Cauchy sous une forme intégrale :

$$x(t_{n+1}) = x(t_n) + \int_{t_{n+1}}^{t_n} f(t, x(t))dt \tag{2.29}$$

Ensuite, on remplace f par le polynôme calculé dont l'intégration est immédiate et les valeurs $x(t_i)$ par leurs approximations x_i. On classe les méthodes d'Adams selon qu'elles sont explicites ou implicites. Les formulations sont toutes deux dues à Adams mais elles sont toujours nommées en association avec la personne qui a fait connaître ses travaux.

Adams-Bashforth

Le principe est d'utiliser les k dernières valeurs $f_{n-k+1},...,f_n$ afin de trouver un polynôme interpolant ces points (Figure 2.6). Ici le polynôme utilisé est le polynôme de Lagrange de degré $k-1$. Dans la suite, on considérera une répartition équidistante des t_i. En substituant P à f dans (2.29), on obtient l'équation suivante :

$$x_{n+1} = x_n + h\sum_{j=0}^{k-1}\gamma_j \nabla^j f_n \quad avec \quad \gamma_j = (-1)^p \int_0^1 \binom{s}{j} ds \tag{2.30}$$

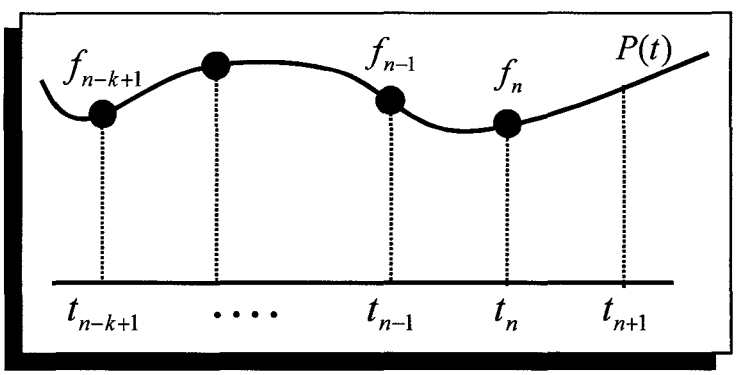


Figure 2.6 : Interpolation de f (Adams Bashforth)

∇ représente l'opérateur aux différences finies rétrogrades définies de la façon suivante : $\nabla^0 f_n = f_n$ et $\nabla^j f_n = \nabla^{j-1} f_n - \nabla^{j-1} f_{n-1}$. La notation sous le signe intégral est celle du coeffient du binôme généralisé :

$$\binom{s}{j} = \frac{s(s-1)...(s-k+1)}{j!}$$

Par l'intermédiaire d'une série génératrice ([HW00]), les coefficients γ_j peuvent être calculés grâce à la relation suivante :

$$\gamma_k + \frac{1}{2}\gamma_{k-1} + \frac{1}{3}\gamma_{k-2} + \ldots + \frac{1}{k+1}\gamma_0 = 1 \tag{2.31}$$

L'ordre d'une telle méthode est fonction du nombre de pas précédents que l'on utilise : pour k pas, la méthode est d'ordre k. Ce qui veut dire que l'on peut obtenir une méthode d'autant plus précise que l'on utilise les pas précédents. Nous verrons dans la section suivante qu'il y a tout de même un gros inconvénient à augmenter le nombre de pas utilisés en ce qui concerne la stabilité numérique. Il reste qu'il y a moyen d'obtenir un ordre de précision élevé sans évaluation supplémentaire de f contrairement aux méthodes RKE.

Les méthodes d'Adams Bashforth sont explicites. La Table 2.4 montre les coefficients pour $k = 1,2,3,4$; on peut remarquer que pour $k = 1$, on retrouve la méthode d'Euler explicite. Nous rappelons que la formulation ci-dessous est valide pour un pas de temps fixe ; utiliser un pas de temps variable nécessite le calcul des coefficients à chaque pas d'intégration. De plus, ce calcul est lui-même un peu plus compliqué dû au fait que l'interpolation ne se fait plus sur des valeurs équidistantes.

$k=1$	$x_{n+1} = x_n + hf_n$
$k=2$	$x_{n+1} = x_n + h\left(\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right)$
$k=3$	$x_{n+1} = x_n + h\left(\frac{23}{12}f_n - \frac{16}{12}f_{n-1} + \frac{5}{12}f_{n-2}\right)$
$k=4$	$x_{n+1} = x_n + h\left(\frac{55}{24}f_n - \frac{59}{24}f_{n-1} + \frac{37}{24}f_{n-2} - \frac{9}{24}f_{n-3}\right)$

Table 2.4 : Coefficients de méthodes d'Adams Bashforth à pas de temps fixes

Adams-Moulton

L'interpolation utilisée pour les méthodes d'Adams Bashforth présente un défaut : on calcule une valeur pour une position en dehors de l'intervalle d'interpolation. Pour éviter cela, il suffit d'utiliser le polynôme qui interpole aussi la position (t_{n+1}, f_{n+1}) (Figure 2.7). Les méthodes obtenues sont alors implicites et nécessitent donc la résolution d'un système non linéaire mais la taille de celui-ci reste égale à la taille du vecteur x, contrairement aux méthodes RKI.

Pour une répartition équidistante des t_i, la formulation de la méthode est :

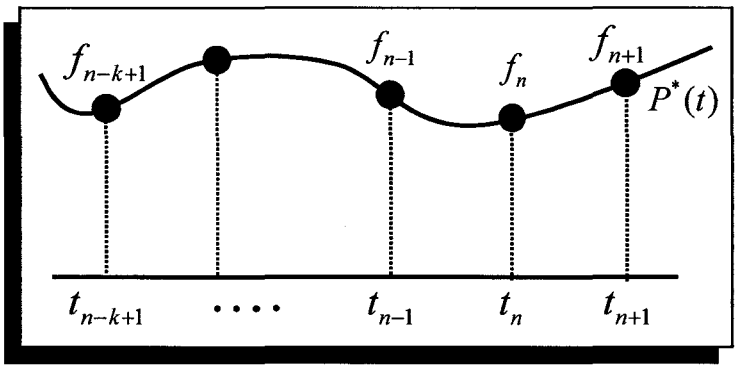


Figure 2.7 : Interpolation de f (Adams Moulton)

$$x_{n+1} = x_n + h\sum_{j=0}^{k} \gamma_j^* \nabla^j f_{n+1} \quad avec \quad \gamma_j^* = (-1)^j \int_0^1 \binom{-s+1}{j} ds \tag{2.32}$$

De la même manière que pour les coefficients des méthodes d'Adams Bashforth, une relation existe entre les coefficients γ_j^* ([HW00]) :

$$\gamma_m^* + \frac{1}{2}\gamma_{m-1}^* + \frac{1}{3}\gamma_{m-2}^* + \ldots + \frac{1}{m+1}\gamma_0^* = 0 \tag{2.33}$$

L'ordre d'une méthode d'Adams Moulton est $k+1$. La Table 2.5 donne des méthodes pour $k = 0,1,2,3$. On remarque pour $k = 0$, on obtient la méthode d'Euler implicite et pour $k = 1$, on obtient la méthode dite des trapèzes.

$k = 0$	$x_{n+1} = x_n + hf_{n+1}$
$k = 1$	$x_{n+1} = x_n + h\left(\frac{1}{2}f_{n+1} + \frac{1}{2}f_n\right)$
$k = 2$	$x_{n+1} = x_n + h\left(\frac{5}{12}f_{n+1} + \frac{8}{12}f_n - \frac{1}{12}f_{n-1}\right)$
$k = 3$	$x_{n+1} = x_n + h\left(\frac{9}{24}f_{n+1} + \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2}\right)$

Table 2.5 : Coefficients de méthodes d'Adams Moulton à pas de temps fixes

Pour éviter la résolution du système non linéaire, il est courant d'utiliser pour les méthodes d'Adams Moulton la prédiction/correction. Il s'agit en fait d'utiliser des itérations du type point fixe. Cette technique se découpe en 4 phases :

1. Prédiction : on utilise à cette fin la méthode d'Adams Bashforth de même ordre pour calculer la valeur $\hat{x}_{n+1}$ qui constitue une approximation de $x(t_{n+1})$.
2. Évaluation : grâce à cette approximation on calcule la valeur $\hat{f}_{n+1} = f(t_{n+1}, \hat{x}_{n+1})$.
3. Correction : on utilise $\hat{f}_{n+1}$ en lieu et place de f_{n+1} pour calculer x_{n+1}.
4. Évaluation : on calcule $f_{n+1} = f(t_{n+1}, x_{n+1})$. Cette valeur servira dans les itérations suivantes.

On dénomme PECE cette résolution. On peut simplifier en omettant la dernière évaluation (méthode PEC) ; dans ce cas, c'est la valeur $\hat{f}_{n+1}$ qui est utilisé dans les itérations suivantes. A l'opposé, il est possible d'itérer plusieurs fois la partie CE (méthode $PE(CE)_N$) pour raffiner le résultat. Cela revient à utiliser une méthode de point fixe (voir § 4.1.1) en limitant arbitrairement le nombre d'itérations.

2.2.3.2 Méthodes BDF (Backward Differentiation Formulas)

Ces méthodes suivent un principe totalement différent des méthodes d'Adams. Ici, c'est la fonction $x(t)$ que l'on va interpoler grâce aux approximations $x_{n-k+1},\ldots,x_n$. On considère donc le polynôme $q(t)$ qui passe par ces points. De plus, on impose la condition suivante :

$$q'(t_{n+1}) = f_{n+1} \tag{2.34}$$

La Figure 2.8 illustre la façon dont est construit $q(t)$.

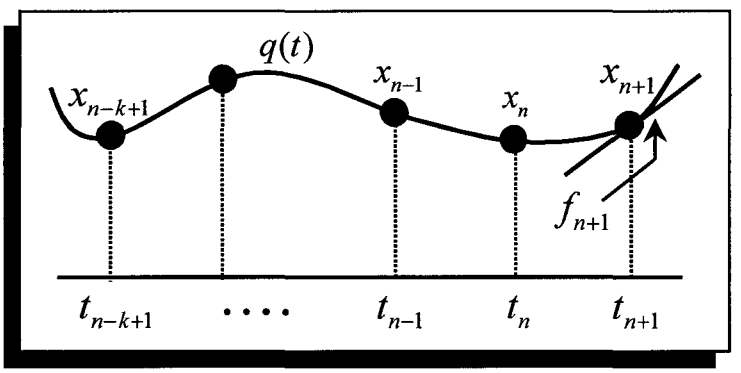


Figure 2.8 : le polynôme q

Pour une répartition équidistante des t_i, on obtient la formule suivante :

$$\sum_{j=0}^{k} \delta_j^* \nabla^j x_{n+1} = h f_{n+1} \quad avec \quad \delta_j^* = (-1)^j \frac{d}{ds}\binom{-s+1}{j}\bigg|_{s=1} \tag{2.35}$$

Les coefficients δ_j^* sont calculables directement :

$$\delta_0^* = 0,\ \delta_j^* = \frac{1}{j}\ pour\ j \geq 1 \tag{2.36}$$

Les méthodes BDF sont implicites et d'ordre k. Il est à noter que les méthodes BDF ne sont stables que pour $k \leq 6$: les 6 méthodes utilisables sont recensées dans la Table 2.6. L'utilisation d'un pas de temps variable complique un peu le calcul comme pour les méthodes d'Adams, mais ce calcul est bien plus simple avec les BDF. Ces méthodes ont un très bon comportement avec les problèmes dits raides dont nous allons parler dans la section 2.3.

$k=1$	$x_{n+1}-x_n=hf_{n+1}$
$k=2$	$\frac{3}{2}x_{n+1}-2x_n+\frac{1}{2}x_{n-1}=hf_{n+1}$
$k=3$	$\frac{11}{6}x_{n+1}-3x_n+\frac{3}{2}x_{n-1}-\frac{1}{3}x_{n-2}=hf_{n+1}$
$k=4$	$\frac{25}{12}x_{n+1}-4x_n+3x_{n-1}-\frac{4}{3}x_{n-2}+\frac{1}{4}x_{n-3}=hf_{n+1}$
$k=5$	$\frac{137}{60}x_{n+1}-5x_n+5x_{n-1}-\frac{10}{3}x_{n-2}+\frac{5}{4}x_{n-3}-\frac{1}{5}x_{n-4}=hf_{n+1}$
$k=6$	$\frac{147}{60}x_{n+1}-6x_n+\frac{15}{2}x_{n-1}-\frac{20}{3}x_{n-2}+\frac{15}{4}x_{n-3}-\frac{6}{5}x_{n-4}+\frac{1}{6}x_{n-6}=hf_{n+1}$

Table 2.6 : Coefficients des méthodes BDF à pas de temps fixes

2.2.4 Méthodes dédiées aux EDO d'ordre 2

Les EDO d'ordre 2 apparaissent fréquemment dès que l'on étudie un problème physique, notamment en mécanique (voir § 1.1.2). C'est pourquoi des méthodes ont été créées pour résoudre ce type d'équations, qui ont la forme suivante :

$$\ddot{x}=f\left(t,x,\dot{x}\right) \tag{2.37}$$

Nous avons vu que l'on peut simplement agrandir le système afin de retrouver un système d'ordre 1 (éq. (2.4)). Il suffit ensuite d'appliquer une des méthodes d'intégration vue précédemment. Cependant, on peut essayer d'exprimer directement l'expression de x sans passer par la variable représentant $\dot{x}$. Nous détaillons tout d'abord une manière d'obtenir une telle expression pour les méthodes RK et ensuite un équivalent pour les méthodes multipas avec les méthodes de Stormer.

2.2.4.1 Adaptation des méthodes RK : méthodes de Nyström

Si on applique une méthode de RK (2.21) à (2.4), on obtient :

$$\begin{aligned}
k_i^x &= v_n+h\sum_{j=1}^{s}a_{ij}k_j^v \\
k_j^v &= f\left(t_n+c_ih,x_n+h\sum_{j=1}^{s}a_{ij}k_j^x,v_n+h\sum_{j=1}^{s}a_{ij}k_j^v\right) \\
x_{n+1} &= x_n+h\sum_{j=1}^{s}b_jk_j^x \qquad v_{n+1}=v_n+h\sum_{j=1}^{s}b_jk_j^v
\end{aligned} \tag{2.38}$$

C'est la manière immédiate d'appliquer une méthode RK à une EDO du second ordre. En réarrangeant les équations de (2.38), on obtient la version suivante :

$$
\begin{aligned}
&k_i^v = f\left(t_n + c_i h, x_n + h c_i v_n + h^2 \sum_{j=1}^{s} \overline{a_{ij}} k_j^v, v_n + h \sum_{j=1}^{s} a_{ij} k_j^v\right) \\
&x_{n+1} = x_n + h v_n + h^2 \sum_{j=1}^{s} \overline{b_j} k_j^v \qquad v_{n+1} = v_n + h \sum_{j=1}^{s} b_j k_j^v \\
&avec\ \overline{A} = A^2\ \ et\ \ \overline{B} = {}^tAB
\end{aligned}
\tag{2.39}
$$

E.J. Nyström a le premier utilisé cette mise en œuvre en créant des méthodes dont les coefficients $\overline{a_{ij}}$ et $\overline{b_j}$ ne sont pas liés aux valeurs a_{ij} et b_j. Les méthodes obtenues n'apportent pas de différences notables avec la formulation (2.39) sauf dans un cas particulier de (2.37), i.e. quand la fonction f ne dépend pas de $\dot{x}$, ce problème devenant :

$$
\ddot{x} = f(t, x) \tag{2.40}
$$

On obtient ce type d'équations, par exemple lorsque l'on considère un système mécanique sans frottement.

Les formules (2.39) deviennent alors

$$
\begin{aligned}
&k_i^v = f\left(t_n + c_i h, x_n + h c_i v_n + h^2 \sum_{j=1}^{s} \overline{a_{ij}} k_j^v\right) \\
&x_{n+1} = x_n + h v_n + h^2 \sum_{j=1}^{s} \overline{b_j} k_j^v \qquad v_{n+1} = v_n + h \sum_{j=1}^{s} b_j k_j^v
\end{aligned}
\tag{2.41}
$$

Nyström obtient alors des méthodes dont l'ordre ne subit plus les contraintes établies par Butcher pour les EDO d'ordre 1 (voir § 2.2.2). Ainsi [HW00] exhibe une méthode d'ordre 4 qui ne nécessite que 3 évaluations de f.

2.2.4.2 Méthodes de Störmer

On peut réécrire les méthodes multipas de la même façon que les méthodes RK :

$$
\begin{aligned}
&\sum_{i=0}^{k} \alpha_i x_{n+1-i} = h \sum_{i=0}^{k} \beta_i v_{n+1-i} \\
&\sum_{i=0}^{k} \alpha_i v_{n+1-i} = h \sum_{i=0}^{k} \beta_i f\left(t_{n+1-i}, x_{n+1-i}, v_{n+1-i}\right)
\end{aligned}
\tag{2.42}
$$

On peut aboutir à une formulation semblable à (2.39) mais on a toujours besoin des deux équations du fait de la présence du terme en vitesse dans f. Par contre, si l'on considère le problème (2.40), la formulation se simplifie en:

$$
\begin{aligned}
&\sum_{i=0}^{2k} \widetilde{\alpha}_i x_{n+1-i} = h^2 \sum_{i=0}^{2k} \widetilde{\beta}_i f\left(t_{n+1-i}, x_{n+1-i}\right) \\
&\widetilde{\alpha}_i = \sum_{j=0}^{i} \alpha_j \alpha_{i-j} \qquad \widetilde{\beta}_i = \sum_{j=0}^{i} \beta_j \beta_{i-j}
\end{aligned}
\tag{2.43}
$$

Cependant, ce n'est pas suivant ce formalisme que les méthodes les plus connues ont été construites. Elles sont appelées méthodes de Störmer ([HW00]). Elles ont été construites pour un problème du type (2.40). Störmer considère le développement limité de $x(t+h)$ et celui de $x(t-h)$. En négligeant des termes dans celui-ci, on obtient :

$$x_{n+1} - 2x_n + x_{n-1} = h^2 f(t_n, x_n) \tag{2.44}$$

Il est possible ensuite de construire des méthodes d'ordre supérieur en prenant en compte les termes du développement limité jusqu'à un certain ordre. Ces approximations sont similaires à celles effectuées par les méthodes d'Adams. La Table 2.7 présente la méthode explicite utilisée par Störmer et une méthode de Störmer implicite.

$$x_{n+1} - 2x_n + x_{n-1} = \frac{h^2}{240}(299 f_n - 176 f_{n-1} + 194 f_{n-2} - 96 f_{n-3} + 19 f_{n-4})$$

$$x_{n+1} - 2x_n + x_{n-1} = \frac{h^2}{240}(19 f_{n+1} + 204 f_n + 14 f_{n-1} + 4 f_{n-2} - f_{n-3})$$

Table 2.7 : Méthodes de Störmer

La formule (2.44) définit la méthode de Störmer-Verlet du fait de sa popularisation par L. Verlet ([Hai99]) pour des calculs de dynamique moléculaire. [Hai99] présente également une formulation à un pas qui utilise une évaluation de la vitesse au milieu du pas de temps :

$$\begin{aligned} v_{n+1/2} &= v_n + \frac{h}{2} f(t_n, x_n) \\ x_{n+1} &= x_n + h v_{n+1/2} \\ v_{n+1} &= v_{n+1/2} + \frac{h}{2} f(t_{n+1}, x_{n+1}) \end{aligned} \tag{2.45}$$

2.3 La Stabilité

2.3.1 Problèmes raides

Nous avons vu dans la section 2.1.2 la notion de stabilité. Cette définition est celle d'une stabilité « théorique » : cela indique que l'on peut trouver un pas d'intégration suffisamment petit pour que le cumul d'erreurs de la méthode soit borné. Dans la pratique, on ne peut pas diminuer énormément le pas de temps pour 2 raisons principales :

- On peut considérer que le coût en calcul reste constant quelque soit la longueur du pas d'intégration[4]. Dans un contexte temps réel, il faut que le temps de calcul soit inférieur ou égal au temps simulé ; ceci impose une limite inférieure à la diminution du pas de temps, liée à la complexité de calcul de f et de la méthode.

[4] C'est vrai pour les méthodes explicites. Les méthodes implicites requérant une méthode itérative, le nombre d'itérations peut être fonction du pas (voir chapitre 4)

⊙ La représentation des nombres flottants dans un ordinateur ajoute une imprécision. Pour des nombres en simple précision, l'addition de 2 nombres dont le rapport est supérieur à 10^{-6} donne comme résultat le plus grand des 2 nombres, considérant le plus petit comme étant égal à 0. Cela conduit aussi à une limite inférieure, liée elle aux ordres de grandeurs de x, f et h.

C'est pourquoi il est utile d'étudier quel va être le comportement d'une méthode suivant le pas de temps choisi. La Figure 2.9 illustre les problèmes rencontrés : le problème traité est $\dot{x} = -10x$ avec $x(0) = -1$. Quatre méthodes sont représentées Euler explicite, RK4, Euler implicite et BDF avec k=3. Il apparaît sur cet exemple que les méthodes explicites ne convergent plus à partir d'une valeur limite (0.2 pour Euler et entre 0.27 et 0.28 pour RK4) et que plus on se rapproche de cette limite, moins la solution converge vite vers la solution. Par contre, les 2 méthodes implicites évoquées ici ne présentent pas ces inconvénients même avec des pas d'intégration assez importants. Dans la suite, nous allons exposer diverses propriétés qui permettent de classer les méthodes suivant leur comportement de stabilité.

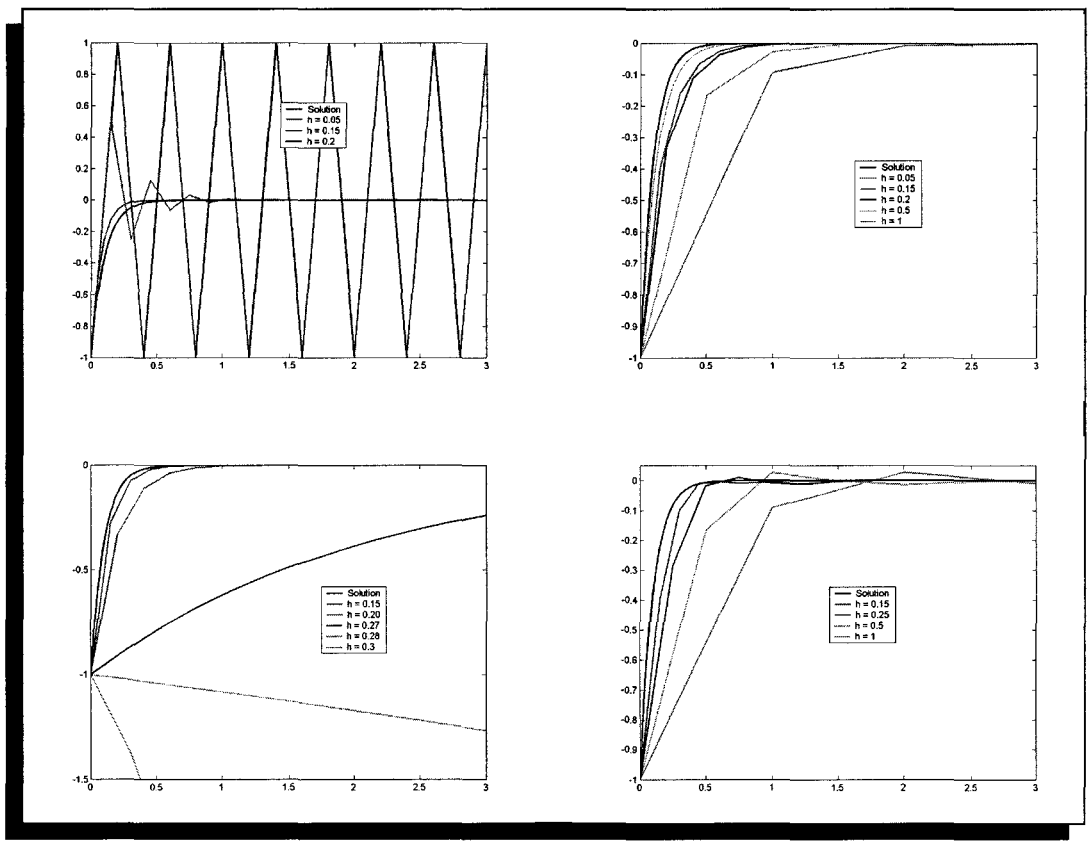


Figure 2.9 : De haut en bas et de gauche à droite : Euler explicite, Euler implicite, RK4 et BDF à l'ordre 3, résolution du problème $\dot{x} = -10x$ avec $x(0) = -1$

Ces problèmes de divergence avec les méthodes explicites obligent à maintenir un pas d'intégration très petit ; tellement petit que , dans certains cas, on fait face aux problèmes suscités. Il en résulte une définition empirique de classification des problèmes à résoudre : on dit qu'un problème est raide si les méthodes usuelles (explicites) ne parviennent pas à une solution en un temps raisonnable ([CM89]).

Cette définition est évasive, les causes de la raideur d'un problème ont de nombreuses origines. On peut dire que celles-ci peuvent provenir des variations trop importantes ou trop rapides de la fonction ou encore de la taille du système (surtout si il met en œuvre des phénomènes de dynamiques différentes). Pour étudier le comportement des méthodes de résolution d'EDO face à ce type de problème, un nouveau concept de stabilité a été défini : la A-stabilité.

2.3.2 La A-stabilité

Afin de mieux appréhender ces problèmes de stabilité des méthodes, il est courant d'étudier la A-stabilité. Celle-ci prend sa définition dans l'approximation (locale) de la fonction à intégrer par sa matrice Jacobienne. Si on suppose $\varphi(t)$ solution de $\dot{x} = f(t,x)$, en considérant la linéarisation de f dans son voisinage et en définissant $\tilde{x} = x - \varphi$, on en déduit [HW96] :

$$\dot{\tilde{x}}(t) = \frac{\partial f}{\partial x}\big(t,\varphi(t)\big).\tilde{x}(t) = J(x)\tilde{x}(t) \tag{2.46}$$

en faisant une approximation au premier ordre (J représente la matrice Jacobienne). De plus, on effectue l'approximation supplémentaire d'une Jacobienne constante.

En appliquant la méthode d'Euler, on obtient la récurrence suivante :

$$x_m = \big(1 + hJ\big)^m x_0 \tag{2.47}$$

Cette suite géométrique ne convergera que si pour chaque valeur propre λ_i de J, le module de $1 + h\lambda_i$ reste inférieur à 1. Cette étude est identique à celle du comportement d'une méthode face au problème suivant :

$$\dot{x} = Ax, \quad x(0) = 1, \quad A \in \mathbb{C} \tag{2.48}$$

pour lequel on étudie le domaine $z = Ah$ tel que la suite (x_i) converge vers zéro[5]. Ce domaine est appelé domaine de stabilité.

Détermination du domaine de stabilité

La détermination du domaine de stabilité prend des formes différentes selon que la méthode est multipas ou non. Pour les méthodes à un pas, on peut ramener la formule de la méthode à

$$x_{n+1} = R(z)x_n \tag{2.49}$$

[5] Ce test est appelé test de Dahlquist

avec $z = Ah$. Le domaine de stabilité est le domaine du plan complexe tel que $R(z) < 1$. Pour les méthodes explicites $R(z)$ est un polynôme en z et pour les méthodes implicites, c'est un quotient de 2 polynômes en z ([HW96]).

Pour les méthodes multipas, la formulation est un peu plus complexe : rechercher le domaine de stabilité revient à étudier les racines du polynôme

$$P(\xi) = (\alpha_k - z\beta_k)\xi^k + \ldots + (\alpha_0 - z\beta_0) \tag{2.50}$$

Le domaine de stabilité est le domaine complexe contenant tous les z tels le polynôme $P(\xi)$ ait toutes ses racines inférieures ou égales à 1 lorsqu'elles sont simples, strictement inférieure à 1 lorsqu'elles sont multiples([HW96]).

La Figure 2.10 présente des domaines de stabilités de méthodes RKE d'ordre 1 à 4 (partie gauche) et des RKI Euler implicite et du point milieu. Pour les RKE, les domaines sont de plus en plus grand à mesure que l'ordre augmente (ainsi que le coût en calcul). Pour les RKI, la relation est moins simple et l'on voit ici qu'une méthode d'ordre 2 a un domaine de stabilité plus petit qu'une méthode d'ordre 1.

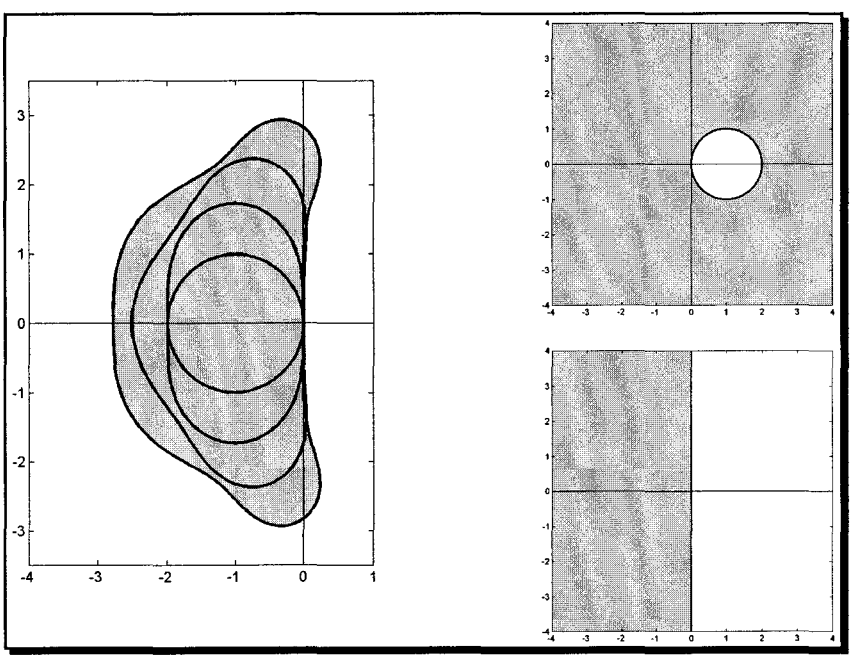


Figure 2.10 : Domaines de stabilité (en gris) : à gauche, RKE d'ordre 1 (Euler explicite, le petit cercle) à 4 (la plus grande forme). A droite, en haut Euler implicite, en bas la méthode du point milieu.

On retrouve les résultats de la Figure 2.9 : pour Euler explicite, on constatait que pour $Ah = -2$, la courbe atteignait une limite et le domaine de stabilité correspondant atteint bien sa limite pour $z = -2$; de même pour RKE d'ordre 4, la limite était atteinte pour une valeur proche de 2,78. Cette valeur qui correspond à la valeur extrémale du domaine de stabilité sur l'axe réel est appelé rayon de stabilité. C'est ce rayon qu'il est utile de connaître dans le cas d'un problème où les valeurs propres de J sont réelles. Enfin, il n'y avait pas de problème avec

la méthode d'Euler implicite et on peut voir que son domaine de stabilité couvre la quasi-totalité du plan complexe.

Une des propriétés que l'on peut attendre d'une méthode est qu'elle converge lorsque le problème (2.48) converge, i.e. le domaine de stabilité doit couvrir le demi-plan complexe à partie réelle négative. On dit alors que la méthode est A-Stable. Les deux méthodes RKI sont A-stables et les RKE ne le sont pas.

Nous allons maintenant voir les domaines de stabilité des méthodes multipas. La Figure 2.11 montre ceux d'Adams-Bashforth pour $k = 2,3,4$; pour $k = 1$, on retrouve Euler explicite. La Figure 2.12 montre ceux d'Adams-Moulton pour $k = 2,3,4$; pour $k = 1,2$, on retrouve respectivement la méthode d'Euler implicite et la méthode des trapèzes qui a le même domaine de stabilité que le point milieu. On s'aperçoit que les domaines diminuent fortement avec l'augmentation de l'ordre. Les seuls méthodes A-stable sont celles d'Adams-Moulton pour $k = 1,2$.

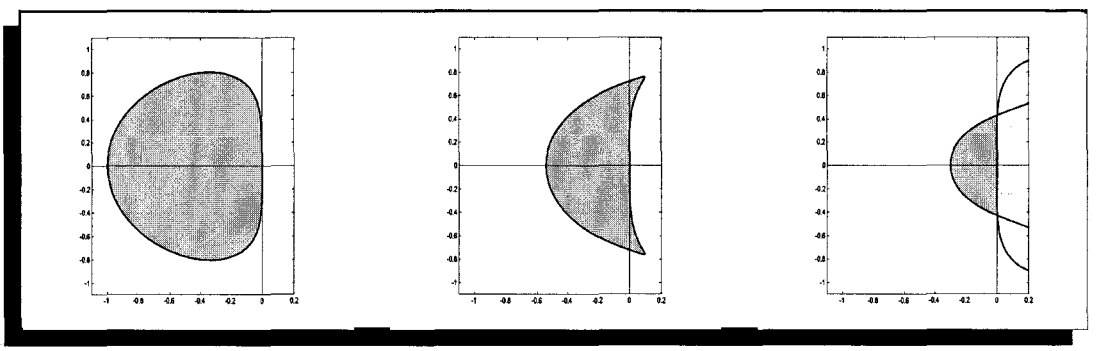


Figure 2.11 : Domaine de stabilité d'Adams-Bashforth pour k=2,3,4.

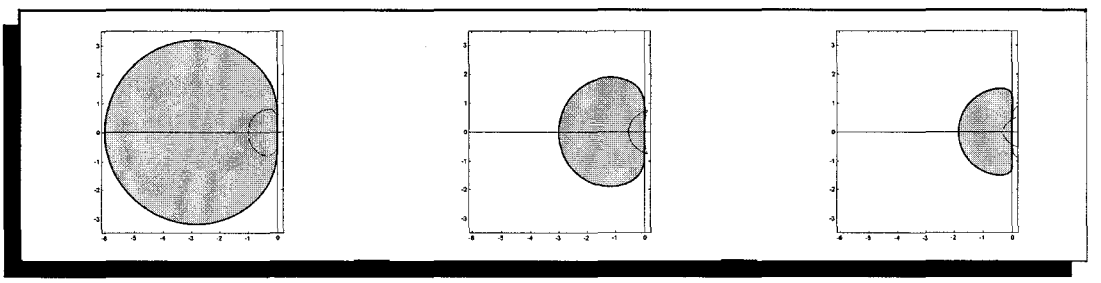


Figure 2.12 : Domaine de stabilité d'Adams-Moulton pour k=2,3,4. En trait fin est dessiné le domaine d'Adams-Bashforth pour la même valeur de k.

En revanche, les BDF possèdent des domaines de stabilité qui couvrent une très grande partie du plan complexe (voir Figure 2.13). Pour $k = 1,2$, les méthodes sont A-stable ; pour les autres valeurs de k, le domaine perd de son ampleur dans une zone proche de l'axe imaginaire.

A(α)-stabilité

Si on reprend la Figure 2.13, on se rend compte que la BDF d'ordre 3 se comporte très bien alors qu'elle n'est pas A-stable. En observant son domaine de stabilité, on peut dire qu'elle

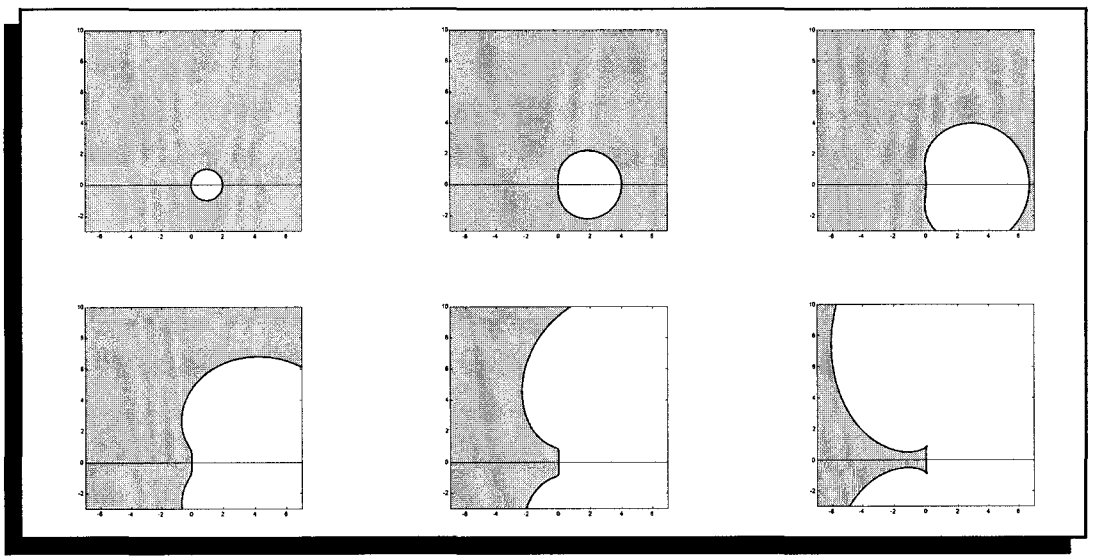


Figure 2.13 : Domaines de stabilité des méthodes BDF, k=1,...,6 de gauche à droite et de haut en bas

est presque A-stable ; pour éviter que la distinction entre les méthodes soit si manichéenne, il a été défini la notion suivante : une méthode est A(α)-stable si le domaine de stabilité D contient le domaine S_α défini par :

$$S_\alpha = \{\mu; |\arg(-\mu)| < \alpha, \mu \neq 0\} \quad avec \quad 0 < \alpha < \pi/2 \tag{2.51}$$

α représente donc une valeur angulaire (en fait, un demi-angle) ; ainsi une méthode A-stable est A(90°)-stable. Les méthodes BDF ont des valeurs de A(α)-stabilité suivantes :

k	1	2	3	4	5	6
α	90°	90°	86,03°	73,35°	51,84°	17,84°

L-stabilité

Il peut paraître que les méthodes optimales soient celles dont le domaine de stabilité est exactement le demi-plan complexe à partie réelle négative. Mais il existe des cas où cette coïncidence des deux ensembles n'est pas avantageuse. Si on prend la formulation (2.49) pour les méthodes RK, on a :

$$|R(iy)| = 1 \quad \forall y \tag{2.52}$$

Or les domaines de stabilités sont des fractions rationnelles en z ([HW96]). Cela implique :

$$\lim_{z\to-\infty} R(z) = \lim_{z\to\infty} R(z) = \lim_{z=iy, y\to\infty} R(z) \tag{2.53}$$

Ce qui veut dire que si une des valeurs propres impliquées dans (2.47) est proche de l'axe réel avec une très grande partie réelle négative, la solution ne sera que « faiblement stabilisée ».

Pour le montrer, nous prenons l'exemple de [HW96] suivant :

$$\dot{x} = -2000\left(x - \cos(t)\right) \quad x(0) = 0 \tag{2.54}$$

La solution passe par une période transitoire qui vise à rejoindre la courbe $x = \cos(t)$. La Figure 2.14 montre que la méthode d'Euler implicite suit correctement la période transitoire tandis que la méthode des trapèzes oscille beaucoup, même en réduisant le pas de temps. Pour garantir une atténuation correcte de la phase transitoire, il faudrait faire en sorte que la limite de $R(z)$ à l'infini soit bien plus éloignée de 1.

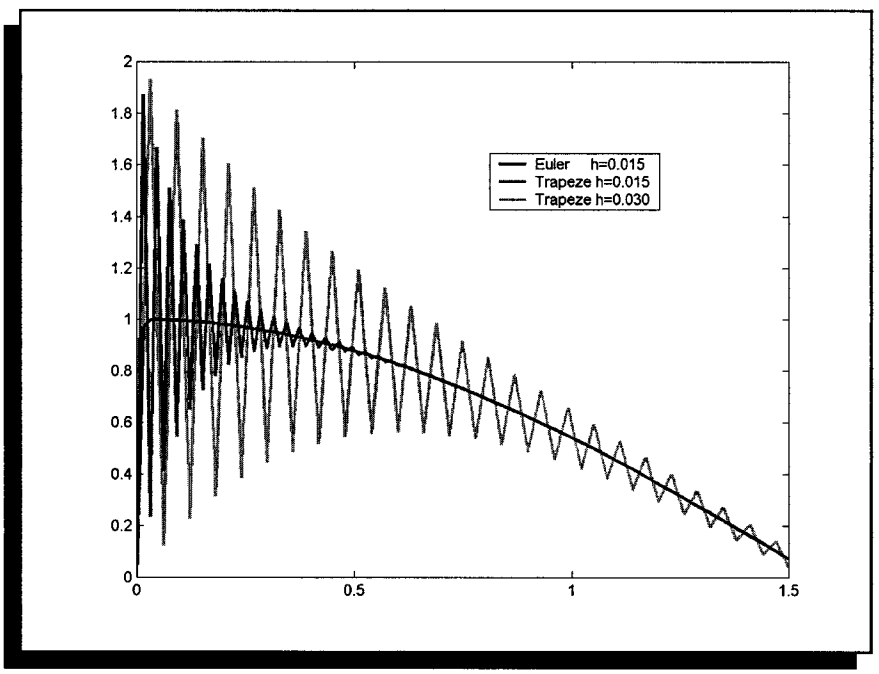


Figure 2.14 : Solution du problème (2.54) pour $x(0) = 0$

Pour caractériser ce fait, on dit qu'une méthode est L-stable si elle est A-stable et que :

$$\lim_{z \to \infty} R(z) = 0 \tag{2.55}$$

Ainsi, une méthode L-stable fonctionne toujours parfaitement avec un problème de ce type. Euler implicite est une méthode L-stable.

2.3.3 Extension au cas général

La A-stabilité est une étude du comportement des méthodes face à un problème linéaire. Cette section décrira la façon dont a été étendu le concept de A-stabilité au problème général.

On considère donc l'EDO $\dot{x} = f(t,x)$; on suppose que f satisfait la condition suivante (appelé condition de Lipschitz unilatérale) :

$$\left\langle f(t,x) - f(t,z), x - z \right\rangle \leq \nu \left\| x - z \right\|^2 \tag{2.56}$$

De plus, si f est continue alors $\forall x(t), z(t)$,solutions de l'EDO, elles vérifient :

$$\|x(t)-z(t)\| \leq \|x(t_0)-z(t_0)\| e^{\nu(t-t_0)} \quad \forall t \geq t_0 \tag{2.57}$$

Ce résultat est important quand $\nu \leq 0$: la distance entre deux solutions est une fonction décroissante de t ([HW96]). Cette propriété est aussi une propriété désirable pour une méthode de résolution d'EDO. On dit qu'une méthode de Runge-Kutta est B-stable lorsque, si f vérifie (2.56) avec $\nu \leq 0$, on a pour tout $h \geq 0$:

$$\|x_1 - \hat{x}_1\| \leq \|x_0 - \hat{x}_0\| \tag{2.58}$$

x_1 et $\hat{x}_1$ représente les valeurs après un pas de la méthode avec les valeurs initiales x_0 et $\hat{x}_0$.

Pour les méthodes multipas, les hypothèses sont les mêmes mais on ne peut utiliser la condition (2.58) car le calcul d'un pas dépend des pas précédents. Une formulation dépendante de ces pas existe[6] et apporte une signification similaire à la B-stabilité pour les RK. On peut trouver les détails dans [HW96].

Cette propriété de stabilité est très forte et d'ailleurs elle implique la A-stabilité. Par exemple, Euler implicite est une méthode B-stable ainsi que la méthode BDF d'ordre 2. Cette notion de B-stabilité garantit une stabilité « à toute épreuve ».

2.3.4 Bilan

Il n'a été présenté ici que les notions de base des différentes notions de stabilité que les numériciens ont définis. Toutefois, les autres définitions sont bien souvent des réductions de conditions comme la A(α)-stabilité. En effet, bien souvent la A-stabilité et la B-stabilité sont des conditions très fortes et peu de méthodes possèdent ces propriétés. De plus, ce ne sont que des indications sur le comportement des méthodes et il se peut qu'une méthode puisse tout de même convenir à un problème donné sans se conformer à ces définitions de stabilité.

	Ordre	A-stabilité	A(α)-Stabilité	L-Stabilité	B-stabilité
Euler explicite	1	-	-	-	-
Runge-Kutte 4	4	-	-	-	-
Adams-Bashforth	k	-	-	-	-
Euler implicite	1	Oui	90°	Oui	Oui
Trapèze	2	Oui	90°	Non	Non
Adams-Moulton k>1	k+1	Non	-	-	-
BDF k=2	2	Oui	90°	Oui	Oui
BDF k=3	3	Non	86,03°	-	-
BDF k=4	4	Non	73,35°	-	-

Table 2.8 :Récapitulatifs des méthodes selon leurs stabilités

La Table 2.8 fait un récapitulatif des stabilités des différentes méthodes qui ont été exposées. Il reste qu'il est difficile d'établir pour un problème donné ce qui va fonctionner. On

peut simplement dire d'ores et déjà que la méthode d'Euler implicite est l'une des rares à satisfaire toutes ces notions. A l'opposé, il n'y a pas de méthodes explicites qui satisfassent une seule de ces propriétés.

2.4 Conservation d'invariant

Cette section a pour but de décrire que la stabilité n'est pas la seule finalité et que d'autres propriétés sont importantes, notamment pour la résolution d'équations liées à certains problèmes de mécanique.

2.4.1 Notion d'invariant et d'index

2.4.1.1 Des quantités à conserver

Pour illustrer la notion d'invariant, nous allons utiliser l'expérience suivante : la simulation de l'orbite de Mars. Celle-ci sera simplifiée dans le sens où il ne sera tenu compte que de l'effet gravitationnel du soleil. Les valeurs initiales sont celles du 1er janvier 2000 à 0h00 GMT, obtenues sur le site du Bureau des Longitudes [BDL]. La Figure 2.15 montre le résultat de cette simulation avec trois méthodes : Euler explicite, Euler implicite et Runge-Kutta d'ordre 4. Le pas de temps est égal à un jour et la simulation est effectuée sur 2100 pas (un peu plus de trois ans de temps simulé).

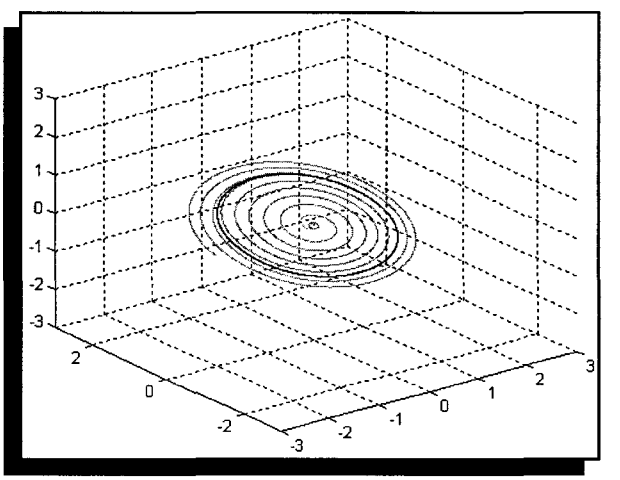


Figure 2.15 : Simulation de l'orbite de Mars : Euler explicite en rouge (orbite extérieure), RK4 en vert et Euler implicite en bleu (orbite intérieure).

On constate que la méthode d'Euler explicite fait sortir Mars de son orbite, que celle d'Euler implicite la fait rentrer et que RK4 arrive à maintenir une orbite fermée pour cette simulation. Cette orbite fermée représente un invariant du système simulé. Tout se passe comme si la méthode d'Euler explicite ajoutait de l'énergie au système et que celle d'Euler implicite amenait une déperdition d'énergie.

[6] pour les méthodes multipas, on parle de G-stabilité

La Figure 2.16 montre une simulation similaire : celle du système solaire lointain (à partir de Jupiter). Les simulations sont faites à pas de temps égal et à coût de calcul égal (RK4 demande quatre évaluations de la fonction contre une pour Euler). On constate encore la supériorité de RK4 alors qu'elle utilise un pas de temps beaucoup plus grand.

Ces invariants sont partie intégrante du problème et donc une certaine attention doit leur être portée. Il existe certains résultats généraux ; les invariants linéaires (la masse totale des réactifs dans des réactions chimiques ou la quantité de mouvement en mécanique par exemple) sont conservés par les méthodes de Runge-Kutta et les méthodes multipas. Une méthode de Runge-Kutta conserve des invariants quadratiques si les coefficients vérifient ([Hai99]) :

$$b_i a_{ij} + b_j a_{ji} = b_i b_j \quad \forall i, j = 1,...,s \tag{2.59}$$

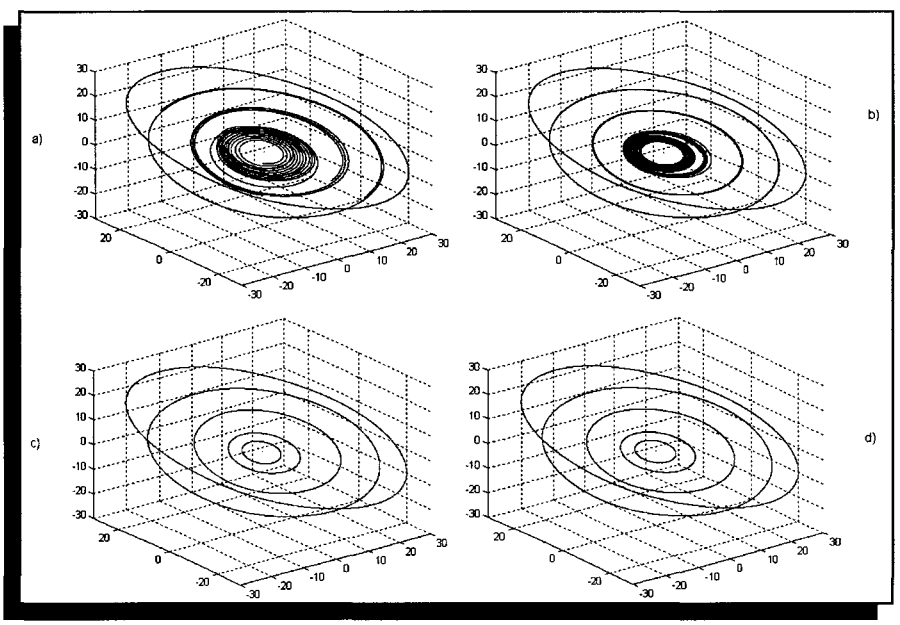


Figure 2.16 : Simulation du système solaire lointain. a) Euler explicite avec h=12 jours, b) Euler explicite avec h=3 jours, c) RK4 avec h=12 jours, d) RK4 avec h=48 jours.

Les autres résultats généraux sont essentiellement des résultats négatifs comme le fait qu'il n'existe pas de méthodes RK qui conservent des invariants polynomiaux de degré supérieur ou égal à 3. Certaines techniques et/ou propriétés permettent de conserver des invariants spécifiques, nous les verrons dans la section suivante.

2.4.1.2 Les systèmes algébro-différentiels

Il existe une 2ème catégorie de quantités que l'on veut voir conserver : les contraintes. Par exemple, il est courant dans un système mécanique, tel des solides rigides articulés, d'imposer des liaisons entre différentes parties et donc d'imposer des limites dans les déplacements possibles. Ces systèmes sont appelés algébro-différentiels car ils regroupent dans le même système des EDO et des équations algébriques.

Pour illustrer ce type de système, nous allons prendre l'exemple de contraintes appliquées à un système en utilisant les multiplicateurs de Lagrange[7]. On utilise le formalisme Lagrangien (voir section 1.1.2.2) et on se fixe des contraintes, i.e. des équations $g_i(q)=0$. On modifie alors l'expression du Lagrangien de la façon suivante :

$$L = K - U - \lambda_1 g_1(q) - \ldots - \lambda_m g_m(q) \tag{2.60}$$

Les λ_i sont prises en compte comme des degrés de liberté supplémentaires. En utilisant cette nouvelle définition de L dans (1.9), on obtient le système à résoudre. Dans l'exemple du pendule, on peut imposer que la corde soit toujours tendue, i.e. $x^2+y^2=l^2$; cela donne, en coordonnées cartésiennes :

$$L = \frac{m}{2}\left(\dot{x}^2+\dot{y}^2\right) - mgy - \lambda\left(x^2+y^2-l^2\right) \tag{2.61}$$

ce qui donne le système :

$$\begin{aligned} m\ddot{x} &= -2x\lambda \\ m\ddot{y} &= -mg - 2y\lambda \\ 0 &= x^2+y^2-l^2 \end{aligned} \tag{2.62}$$

La forme générale d'un tel problème est (il faut évidemment réduire l'ordre des éventuelles EDO d'ordre supérieur à 1) :

$$F\left(\dot{x},x\right)=0 \tag{2.63}$$

Une caractéristique de ce type de système est ce que l'on appelle l'index de différentiation : c'est le nombre m minimal de différentiations qu'il faut opérer sur F de telle façon qu'après manipulation sur l'ensemble des équations

$$F\left(\dot{x},x\right)=0,\ dF(\dot{x},x)/dt=0,\ \ldots,\ d^mF(\dot{x},x)/dt^m=0$$

on puisse extraire une EDO du type $\dot{x}=\varphi(x)$. Si on reprend l'exemple (2.62), c'est un système d'index 3 : il faut en effet différencier 3 fois la contrainte et substituer les valeurs des autres équations pour obtenir une EDO du 1er ordre en $\dot{\lambda}$ (moyennant quelque condition d'inversibilité sur les jacobiennes des fonctions). La technique employée pour résoudre ce genre de problème rejoint une technique pour la conservation d'invariant que nous allons détailler dans la section suivante.

[7] On trouvera dans [Rem00] une étude complète de l'utilisation de ces multipicateurs.

2.4.2 Méthodes appropriées

2.4.2.1 Méthodes Symplectiques

Celles-ci ont été étudiées dans le cadre de la résolution des systèmes Hamiltoniens, i.e. où toutes les forces sont conservatives. Ce formalisme mécanique introduit par Hamilton est basé sur celui de Lagrange ; comme les forces dérivent d'un potentiel, elles sont toutes regroupées dans l'énergie potentiel U et il n'y a plus de terme Q dans l'équation (1.9). On définit de nouvelles variables $p_i = \partial L / \partial \dot{q}_i$ qui sont appelés moments généralisés. Ensuite, il faut considérer la grandeur, appelée Hamiltonien, $H(p,q) = {}^t p\dot{q} - L(q,\dot{q})$. Alors l'équation (1.10) est équivalent à :

$$\dot{p}_k = -\frac{\partial H}{\partial q_k}(p,q),\ \dot{q}_k = -\frac{\partial H}{\partial p_k}(p,q),\ k = 1,...,d \tag{2.64}$$

Cela représente une réécriture des équations de départ dans le but de les rendre plus « symétriques » ; en fait, le Hamiltonien peut être défini comme l'énergie totale du système :

$$H = K + U \tag{2.65}$$

Cette énergie totale peut être considérée comme une valeur invariante ; celle-ci est quadratique puisqu'elle représente une énergie. Mais le Hamiltonien possède une propriété plus remarquable : c'est une transformation symplectique. Une application différentiable est symplectique ([HW00]) si sa jacobienne $R = g'(p,q)$ satisfait :

$${}^tRJR = J \ \ avec\ J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \tag{2.66}$$

La symplecticité est une caractéristique intéressante liée à la préservation de volume : en dimension 2, l'aire d'une figure qui bouge selon une application symplectique conserve son aire.

$H(p,q)$ est une transformation symplectique. Il semble donc intéressant d'utiliser une méthode qui va satisfaire cette propriété. Elle est en effet assez rarement conservée. L'étude de méthodes symplectiques n'existe que pour les méthodes de Runge-Kutta : il suffit qu'elle vérifie la condition (2.59). Ce sont des méthodes implicites, la Table 2.9 en présente une.

$\frac{4-\sqrt{6}}{10}$	$\frac{16-\sqrt{6}}{72}$	$\frac{328-167\sqrt{6}}{1800}$	$\frac{-2+3\sqrt{6}}{450}$
$\frac{4+\sqrt{6}}{10}$	$\frac{328+167\sqrt{6}}{1800}$	$\frac{16+\sqrt{6}}{72}$	$\frac{-2-3\sqrt{6}}{450}$
1	$\frac{85-10\sqrt{6}}{180}$	$\frac{85+10\sqrt{6}}{180}$	$\frac{1}{18}$
	$\frac{16-\sqrt{6}}{36}$	$\frac{16+\sqrt{6}}{36}$	$\frac{1}{9}$

Table 2.9 : une méthode symplectique [HW00]

On a vu sur la Figure 2.15 que la méthode RKE d'ordre 4 présente un bon comportement alors qu'elle ne peut pas être symplectique. En fait, elle est symplectique jusqu'à un certain ordre ([Aub97]), c'est-à-dire que le caractère symplectique est conservé pour une certaine précision et que cette méthode convient à la simulation présentée.

L'utilisation de ces méthodes symplectiques est très importantes pour la simulation d'un système lui-même symplectique. Or, un système comportant des forces non-conservatives (forces de frottements par exemple) ne peut produire un flot symplectique. On peut tout de même dire qu'une telle méthode aura un effet bénéfique dans le sens où elle sera la plus neutre possible vis à vis du système simulé.

2.4.2.2 Méthodes symétriques

Pour définir l'utilité des méthodes symétriques, il faut à nouveau partir d'une constatation des propriétés de la solution de l'équation. Ici on va s'intéresser à son caractère « réversible » : si on suit la solution pendant un temps t puis on la suit du point final à rebours, i.e. pendant un temps $-t$, on peut s'attendre à ce qu'elle revienne au même endroit (Cela n'est pas toujours le cas mais constitue une bonne part de problèmes possibles).

Une méthode à un pas est une fonction $\Phi_h : x_0 \mapsto x_1$. Celle-ci est aussi définie pour des valeurs de h négatives. Une méthode qui préserve la propriété précédente est dite symétrique et vérifie :

$$\Phi_h \circ \Phi_{-h} = Id \text{ ou } \Phi_h = \Phi_{-h}^{-1} \tag{2.67}$$

On appelle la fonction $\Phi^* = \Phi_{-h}^{-1}$, la méthode adjointe à Φ. L'opérateur « adjoint » satisfait les propriétés habituelles de ces opérateurs comme l'involution[8].

Pour déterminer la méthode adjointe, il suffit d'effectuer les transformations qui consistent en les deux échanges $x_0 \leftrightarrow x_1$ et $h \leftrightarrow -h$. Par exemple, la méthode adjointe d'Euler explicite est Euler implicite et donc ni l'une ni l'autre ne sont symétriques. Par contre, les méthodes du point milieu et du trapèze le sont. Les méthodes symétriques garantissent souvent un bon comportement avec les systèmes qui ont un comportement cyclique.

2.4.2.3 Méthodes par projection

Le principe de ces méthodes est simple : un invariant (ou une contrainte) va réduire le sous-ensemble dans lequel évolue les solutions. Une méthode peut produire un résultat qui se trouve en dehors de ce sous-ensemble : il suffit de le ramener dans ce domaine de validité ; cette opération consiste donc en la projection du résultat sur le sous-ensemble défini par l'invariant.

---

[8] $\left(\Phi^*\right)^* = \Phi$

Cette projection peut prendre n'importe quel forme (orthogonale, Groupe de Lie [Laz95]) et elle n'altère pas la précision de la méthode utilisée [Hai99], étant effectuée après chaque pas d'intégration. Cette technique est très intuitive : on corrige une estimation produite par la méthode de résolution de l'EDO. Il faut tout de même prendre certaines précautions : il est important de recenser tous les invariants du système. L'exemple de [Hai99] sur un système à deux invariants utilise deux méthodes de résolution : une qui ne conserve aucun des deux invariants et une autre qui conserve les deux. Pour la 1ère, la projection selon un invariant améliore le comportement de la solution obtenue et la projection selon les deux invariants procure un résultat correct. Pour la 2nde, le résultat est correct dès le départ grâce au fait que la méthode est symplectique et de ce fait conserve cette propriété partagée par la solution réelle. Cependant, lorsque l'on applique une projection selon un des deux invariants, cela détruit le bon comportement initial de la méthode. Il faut alors la projection selon l'autre invariant pour retrouver une solution correcte.

Il faut donc manier avec précaution la projection selon un invariant. Il reste que son utilisation avec une méthode qui ne présente aucune propriété de conservation est parfaitement adaptée.

2.4.2.4 Réduction d'index

Cette technique s'adresse aux systèmes algébro-différentiels en général. Afin de résoudre ceux-ci, il faut intégrer à la méthode de résolution de l'EDO les équations définissant les contraintes. Celles-ci influent sur la méthode et modifie les conditions de convergence et de stabilité. Cela implique aussi la résolution d'un système non-linéaire. Des études existent concernant les méthodes de résolution de problème d'index 2 mais on préfère généralement revenir à un problème d'index 1, pour lequel on peut utiliser toutes les méthodes classiques.

Pour illustrer ce propos, nous reprenons le formalisme Lagrangien augmenté des multiplicateur de Lagrange (2.4.1.2). On obtient alors le système suivant :

$$\begin{aligned} \dot{q} &= u \\ M(q)\dot{u} &= f(q,u) - {}^{t}G(q)\lambda \\ 0 &= g(q) \end{aligned} \tag{2.68}$$

avec $q = {}^{t}(q_1,...,q_n)$, $u = {}^{t}(\dot{q}_1,...,\dot{q}_n)$, $\lambda = {}^{t}(\lambda_1,...,\lambda_n)$ et $G = \partial g / \partial q$. C'est un système d'index 3. Si on dérive 2 fois la 3ème équation, on peut ramener les deux dernières équations au système suivant :

$$\begin{pmatrix} M(q) & {}^{t}G(q) \\ G(q) & 0 \end{pmatrix} \begin{pmatrix} \dot{u} \\ \lambda \end{pmatrix} = \begin{pmatrix} f(q,u) \\ -g_{qq}(u,u) \end{pmatrix} \tag{2.69}$$

De cette façon, on peut calculer $\dot{u}$ et λ en fonction de q et u. Il ne reste plus qu'à intégrer l'EDO en u. C'est là, l'intérêt d'une réduction de l'index du problème de départ.

Cette diminution de l'index présente le problème suivant : le système (2.69) est valide quelque soit les valeurs de q et u, notamment en dehors de l'ensemble défini par la contrainte

et généralement, la solution obtenue a tendance à ne pas respecter la contrainte exactement. Afin de revenir à la contrainte, il faut projeter la solution obtenue sur le sous-ensemble défini par les contraintes. Mais il faut tenir compte de contraintes liées à l'index du problème.

Pour un problème d'index 3 tel que (2.68), il faut dériver 2 fois pour obtenir (2.69). D'une contrainte en position, on est passé à une contrainte en accélération qui, elle, est respectée. Par ces dérivations, on introduit un nouvel invariant : une contrainte en vitesse, obtenue en dérivant une fois la contrainte en position. Pour ce type de problème, il faut donc en toute logique effectuer une projection selon les deux contraintes qui ne seront pas respectées. Selon [A094], il apparaît que la projection selon la contrainte en vitesse est primordiale et que celle en position n'apporte que peu de précision supplémentaire. On pourrait même se passer de cette dernière.

2.5 Utilisation de ces méthodes en animation

Dans cette partie, nous allons expliciter les méthodes utilisées dans les travaux précédents et comment elles ont été utilisées. Nous commençons par les méthodes explicites car ce sont celles qui ont été utilisées au début de l'animation basée sur la physique. Ensuite nous aborderons les méthodes implicites, utilisées en animation bien plus récemment.

2.5.1 Méthodes Explicites

Ces méthodes ont été les premières mises en œuvre car elles ne représentent qu'un faible coût de calcul et leur implantation est relativement simple. Parmi toutes les méthodes existantes, sont utilisées la méthode d'Euler ([Pro95]), d'Euler modifié ([LJFC91]), de Störmer-Verlet ([DG96], [BC00], [PLDA00]), des méthodes RKE (RK4 pour [Bar95], [Jou95], point milieu explicite pour [VCM95]). Dès lors qu'elles servent à l'animation d'objets déformables, elles font face à des problèmes de stabilité. Pour garder une simulation qui ne diverge pas, différentes solutions existent.

Une 1ère catégorie cherche à influer sur les paramètres physiques d'une scène. La solution la plus simple consiste en l'augmentation artificielle de l'amortissement des forces ; cet amortissement n'a pas de réalité physique et s'indexe directement sur la vitesse de tous les objets simulés. Cela peut d'une certaine manière modéliser grossièrement les frottements fluides du milieu ambiant. De cette façon, on crée une force allant à l'encontre du déplacement permettant de garder une simulation correcte. Cette technique présente un défaut majeur : en augmentant la valeur de cet amortissement, on finit par obtenir une simulation dont le milieu ambiant semble être de l'huile. En outre, un trop gros amortissement peut générer des forces immenses qui généreront à leur tour de l'instabilité. Il s'agit de jongler correctement avec ces valeurs.

Une autre solution est celle proposée par X. Provot ([Pro95]). Lors de la modélisation d'un objet déformable, il utilise un maillage masses/ressorts. Pour contourner les problèmes de divergence, il limite l'élongation des ressorts et l'effort généré par l'amortissement. Ces limitations permettent de limiter l'amplitude des forces générées et donc de rester dans le domaine de validité. Ainsi cette association permet de retrouver une simulation d'un ressort qui ne di-

verge pas. La limite de l'élongation d'un ressort dépend de sa raideur, de telle sorte que cela gène la simulation de raideurs importantes. En ajoutant la limitation des forces d'amortissement, on peut dire qu'il devient difficile de trouver des valeurs permettant de représenter un corps précis.

La 2nde catégorie de solutions pour assurer la stabilité utilise l'adaptation du pas d'intégration. La difficulté de cette technique est de trouver un critère qui permette de déterminer l'évolution que doit suivre le pas de temps. A. Joukhadar [Jou95] propose de calculer l'énergie du système simulé. Celle-ci doit rester constante. Elle est évaluée à la nouvelle position : si l'énergie a varié, le pas de temps est divisé par 2 et le pas est recalculé ; sinon, le pas de temps est multiplié par 1,5. Il obtient ainsi une simulation d'une balle en chute libre qui rebondit sur une surface plane qui reste stable grâce à une diminution très forte du pas de temps pour gérer les moments autour de l'impact sur le sol. Une technique similaire à cette solution est employée dans le moteur dynamique commercial [Havok].

Cette utilisation de l'énergie du système est valable pour une telle simulation car l'énergie du système simulée (la balle) est simple à évaluer. L'évaluation de l'énergie est délicate lors de la présence de forces de frottements (notamment de frottements secs). De même, l'intervention d'un objet manipulé par un opérateur humain rend difficile cette évaluation et complique le critère puisqu'il faut tenir compte de la possibilité d'augmentation ou de diminution de cette énergie.

D'autres critères existent comme celui de [DG96], lié à la vitesse de propagation d'une déformation dans le matériau simulé. Cependant, ils sont généralement liés à un modèle mécanique précis et ne se prêtent pas à une utilisation générique.

Pour terminer, nous avons voulu détailler une méthode qui est à mi-chemin vers les méthodes implicites. Cet article ([TPBF87]) est fondateur en ce qui concerne l'utilisation de la mécanique en animation. La modélisation mécanique suit un modèle de milieu continu grâce à laquelle il obtient une équation aux dérivées partielles dans le temps et dans l'espace. Celle-ci est discrétisée par la méthode des différences finies dans le domaine spatial afin d'obtenir une EDO relativement au temps (voir 1.1.2.2). Il décrit sa méthode comme étant explicite spatialement et implicite temporellement mais il faut apporter une précision quant à l'utilisation de ces termes.

Le terme « explicite » est utilisé dans le sens où les différences finies ont rendu directement calculable les dérivations spatiales. Le terme « implicite » est utilisé pour décrire le fait que l'EDO obtenue ne donne pas une formulation tout à fait directe de la dérivée seconde de la position. En effet, l'équation est de la forme :

$$Ax_{t+\Delta t} = g\left(\Delta t, x_t, v_t\right) \tag{2.70}$$

où x représente les coordonnées de l'ensemble des points simulés (on peut retrouver cette formulation dans la section 1.1.2.2). A est une matrice qu'il faut inverser[9] et qui rend donc le calcul un peu moins direct. Les auteurs aboutissent à une telle formule parce que le tenseur des efforts (calculé à l'instant t) ést appliqué à la position à l'instant $t+\Delta t$. C'est là que réside la différence avec une intégration implicite où tout doit être pris en compte à l'instant $t+\Delta t$.

Il reste que la méthode de résolution de l'EDO est la méthode d'Euler explicite. On peut constater que la part faite à l'explication de cette méthode est très restreinte. Cela peut s'expliquer par le fait que le principal effort à faire à l'époque était de montrer qu'une telle simulation était possible et que la principale difficulté était de proposer un modèle mécanique exploitable pour une simulation. Euler représente aussi une manière naturelle de traiter une EDO du fait qu'une approximation du premier ordre était déjà effectuée par les différences finies.

Il y a peu de détail sur les conditions dans lesquelles ont été faites les simulations. En effet, il est clair que des problèmes de stabilité ont dû être rencontrés, bien que probablement amoindris par la formulation (2.70), et qu'ils ont dû être résolus afin de rester dans le domaine de valeur que la méthode d'Euler peut supporter.

Au final, toutes les techniques pour assurer une certaine stabilité exposées précédemment imposent d'être utilisées avec précaution et sont limitées dans la gamme de modèles qu'elles peuvent simuler. C'est ce qui a conduit à l'utilisation des méthodes implicites (ainsi que l'augmentation de la puissance de calcul disponible).

2.5.2 Méthodes Implicites

Pour éviter les problèmes de stabilité avec les méthodes explicites, nous avons vu qu'il faut introduire des corrections plus ou moins « artificielles ». Nous avons vu que du point de vue de la stabilité (§ 2.3), les méthodes implicites montrent des aptitudes meilleures. C'est ce qui a mené à l'utilisation de méthodes implicites. Nous commençons par la méthode d'Euler parce que cela a été la 1ère introduite.

2.5.2.1 Euler

L'utilisation des méthodes implicites est très récente : l'article fondateur est celui de Baraff et Witkin [BW98]. Le but recherché est l'animation de tissu. Le formalisme mécanique utilisé aboutit à l'équation suivante :

$$\ddot{x} = M^{-1}\left(-\frac{\partial E}{\partial x} + F\right) \Leftrightarrow \begin{pmatrix}\dot{x}\\ \dot{v}\end{pmatrix} = \begin{pmatrix} v \\ f(x,v)\end{pmatrix} \tag{2.71}$$

[9] Quand la matrice n'est pas inversible, il existe des techniques pour tenter de résoudre le problème, proche des techniques liées aux problèmes avec index (voir [HW96]).

où E représente l'énergie interne du tissu et F les autres forces. Le tissu est discrétisé dans l'espace afin d'expliciter le calcul de la dérivation de E. Ainsi on aboutit à une EDO sur la variable temporelle. En partant du constat que les méthodes explicites traditionnellement utilisées ne convenaient pas pour les raisons exposées précédemment, ils ont choisi d'utiliser la méthode d'Euler implicite.

Le problème essentiel d'une méthode implicite est la résolution du système non-linéaire. Ici le système d'équations à résoudre à chaque pas d'intégration est le suivant :

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} x_n \\ v_n \end{pmatrix} + h \begin{pmatrix} v_{n+1} \\ f(x_{n+1}, v_{n+1}) \end{pmatrix} \tag{2.72}$$

Pour résoudre cette équation, les auteurs effectuent une approximation linéaire de la fonction à annuler (développement de Taylor tronqué au 1er ordre). De cette façon, on aboutit au système linéaire suivant :

$$\left(I - hM^{-1}\frac{\partial f}{\partial v} - h^2 M^{-1}\frac{\partial f}{\partial v} \right) \Delta v = hM^{-1}\left(f_n + h\frac{\partial f}{\partial x} v_n \right) \tag{2.73}$$

Δv représente la différence entre la vitesse présente et la vitesse future. On obtient une unique équation du fait que l'on peut exprimer directement x_{n+1} en fonction de v_{n+1}. Il est à noter que la matrice à inverser utilise les matrices jacobiennes de f au point (x_n, v_n). Cette formulation correspond en fait à la méthode de Newton pour la résolution de système non linéaire (voir chapitre 4) dans lequel on ne ferait qu'une seule itération du processus complet.

Le calcul des matrices jacobiennes de f est un problème délicat : ici, le modèle mécanique choisi permet d'avoir une formulation analytique de la dérivée. L'autre problème est le stockage de la matrice : elle est de taille 3nx3n avec n le nombre de points. Ici aussi, le modèle mécanique est simplifié de telle façon que l'influence d'un point sur l'énergie interne reste local : ainsi la matrice est creuse et permet un stockage efficace. Il reste maintenant à utiliser une méthode de résolution de système linéaire.

Les méthodes directes (Gauss, LU, ...) sont exactes : les seuls écarts par rapport à la solution réelles proviennent des erreurs d'arrondis dus au stockage des nombres flottants. Malheureusement, celles-ci sont des méthodes en n^3 (donc peu efficaces) et de plus elles détruisent en général le caractère creux de la matrice initiale. Les auteurs se sont tournés vers une méthode qui exploite cette caractéristique et qui est plus efficace : le Gradient Conjugué. C'est une méthode itérative qui n'exploite que des produits matrices vecteurs, pour lesquels des algorithmes dédiés à des matrices creuses existent. Il reste que cette méthode ne fonctionne que pour une matrice symétrique définie positive.

Pour rendre la matrice symétrique, l'équation (2.73) est multipliée par M. De plus, une simplification est opérée dans le calcul des dérivations de f : le calcul est tronqué par la suppression des termes qui apportent la dissymétrie. Les auteurs admettent qu'il n'y a aucune jus-

tification physique à cette approximation. Cependant, ils ont tout de même constaté que les simulations effectuées sans ce terme garde un aspect satisfaisant.

Ces différentes propositions permettent de diminuer le coût de l'utilisation d'une méthode implicite. Conjugué avec des pas de temps relativement important et variables, le résultat final est une simulation qui reste correcte même avec des raideurs importantes et qui présente des temps de calcul acceptables mais ceux-ci restent trop élevés dans le cadre d'une animation temps réel.

Les points importants liés à l'utilisation d'une méthode implicite sont :

- La linéarisation du problème
- La formulation du système linéaire
- La résolution du système linéaire

L'ensemble bénéficie fortement du système mécanique utilisé : la forme de la matrice en est directement dépendante, ce qui rend possible l'utilisation de méthodes dédiées (comme le gradient conjugué). Il reste que ces travaux ont montré que ces méthodes étaient utilisables et ont ouvert la voie.

Un autre travail s'est directement inspiré de [BW98]. Schröder et alter [SDB99] choisissent Euler implicite pour animer un maillage masses/ressorts. Pour linéariser le problème, ils éliminent du modèle mécanique la partie qui apportent la non-linéarité ; de plus, la partie linéaire produit une matrice qui est constante et permet l'application en précalcul d'un algorithme de résolution de système linéaire exacte. Il est à noter que ce précalcul interdit une éventuelle modification dynamique du modèle ainsi qu'un pas de temps adaptatif.

Ici, la simplification sur le modèle est beaucoup plus forte que [BW98] et les auteurs ont dû ajouter une phase de correction. Celle-ci est liée au modèle mécanique : la partie non-linéaire correspond aux mouvements de « rotation » et donc la correction correspond à une conservation du moment angulaire. L'ensemble ne constitue pas à proprement parlé une méthode implicite mais elle donne de bons comportements, avec une vitesse de traitement supérieure.

La supériorité des méthodes implicites sur les méthodes explicites quant à la stabilité du processus est indéniablement montrée par ces travaux mais elles restent coûteuses ; deux étapes rassemblent l'essentiel de ce coût : la formulation du système linéaire et sa résolution. Il est à noter que les techniques présentées pour réduire ce coût sont partiellement ou totalement dédiées aux modèles mécaniques utilisés.

2.5.2.2 Autres Méthodes

L'introduction d'autres méthodes que celle d'Euler suit deux buts différents : la diminution de la complexité du système non-linéaire à résoudre et l'étude du comportement de la solution obtenue.

Diminution du coût de calcul

[EEH00] réduisent les calculs en intervenant sur le modèle physique par un mécanisme de multirésolution. La solution proposée propose aussi de séparer le modèle mécanique en deux parties : une partie dite « non raide » et une autre dite « raide ». En effet, cette dernière est intégrée grâce à une méthode implicite et l'autre partie par une méthode explicite. La théorie (convergence, etc) est tirée d'un article de Asher et alter ([ARW95]), lequel établit les résultats sur les méthodes multipas.

Les auteurs formalisent cette séparation sur un maillage masses/ressorts 2D pour, là encore, la simulation de tissu. Cette séparation permet de réduire le coût de calcul de la fonction à intégrer implicitement qui constitue une partie importante du coût total. Pour la résolution du système non-linéaire, c'est la même technique que [BW98].

Ils arrivent ainsi à une simulation convaincante pouvant traiter des raideurs importantes et ce pour un coût moindre que [BW98]. Il faut tout de même pouvoir opérer au préalable la séparation entre partie raide et non-raide, ce qui n'est peut-être pas aussi facile que sur un maillage masses/ressorts. Il est à noter que les auteurs ne précisent pas quelle méthode multipas a été utilisée dans leurs tests. Ils semblent suggérer l'emploi des BDFs comme méthodes implicites mais rien en ce qui concerne la méthode explicite associée.

Amélioration du comportement

Nous avons vu que la méthode implicite principalement employée était celle d'Euler. Nous savons que son ordre de précision est 1 et que son domaine de stabilité couvre la quasi-totalité du plan complexe. Cette bonne propriété peut être vue comme un défaut dans certains cas. Si on prend, par exemple, une valeur du domaine de stabilité qui a sa partie réelle positive, la solution produite par la méthode d'Euler implicite converge donc vers 0. Par contre, la solution de l'équation test associée diverge.

Au-delà de cette « surconvergence », cela influe sur les problèmes par un ajout d'une sorte d'amortissement qui lutte contre la divergence. Ceci est souligné comme une qualité par [SDB99] pour assurer la stabilité mais est dénoncé comme un défaut de réalisme par [VM00]. En effet, certains phénomènes de plis qui apparaissent sur les tissus peuvent être diminués voir absents lors d'une simulation. [VM00] propose d'utiliser la méthode du point milieu ; nous avons vu que celle-ci était strictement A-Stable et ainsi ne fait plus converger un problème qui diverge. Bien évidemment, cet amélioration se fait au détriment de la stabilité. On peut aussi ajouter que cette méthode, étant d'ordre 2, est aussi plus précise.

2.6 BILAN

Nous avons découvert tout au long de ce chapitre, les principes fondamentaux des méthodes de résolution des équations différentielles ordinaires ainsi que différentes propriétés relatives à leur comportement face aux solutions exactes : la précision, la stabilité, la possibilité de conserver certaines quantités ou de maintenir des contraintes. Nous avons donné des exem-

ples de méthodes mais cela ne représente qu'un faible éventail de toutes les méthodes existantes, notamment en ce qui concerne les méthodes RK.

Ces propriétés peuvent servir à mener vers la méthode à utiliser pour résoudre son problème. Il faut voir les diverses classifications et propriétés comme une aide mais cela ne remplace pas l'expérimentation relative à un problème donné. En effet, on ne peut pas affirmer que telle ou telle méthode est « la » meilleure. Il faut toujours mettre en balance les propriétés du problème (Est-on en face d'un problème raide ? Le problème exhibe-t-il des invariants ?) et les priorités que l'on accorde à la résolution (Est-ce qu'il faut que la solution obtenue soit précise ? Existe-t-il des contraintes de temps de calcul ? Est-il important de respecter les invariants éventuels ?).

C'est pourquoi nous avons mené des expérimentations sur le type d'équations que nous avons à résoudre. Nous avons découpé cette étude en deux : d'abord les méthodes explicites puis les méthodes implicites. Ce découpage a été motivé par le fait que notre principale priorité est la rapidité d'exécution. De plus, les méthodes explicites sont les plus simples à mettre en œuvre et facilitent l'insertion des mécanismes liés à la mise en œuvre de la résolution dynamique dans la bibliothèque SPORE.

Chapitre 3 : Tentative de résolution explicite dans SPORE

Dans ce chapitre, nous voulons tout d'abord mettre l'accent sur les principes de mise en œuvre de diverses méthodes explicites de résolution des EDO. Ceux-ci sont définis afin de garder une grande généricité dans la gamme d'équations pouvant être résolue. Ensuite, nous relaterons les résultats obtenus sur des expériences représentant des cas typiques de ce que SPORE doit pouvoir simuler.

3.1 Choix de mise en œuvre dans SPORE

3.1.1 Structure de donnée

Dans la présentation de méthodes de résolution d'EDO (§ 2.5), nous avons vu qu'elles étaient destinées à un modèle mécanique précis. Notre objectif est d'insérer dans SPORE les mécanismes permettant de gérer l'évolution de corps mécaniques, sans a priori quant à leur forme. Dans la section 1.1.2, nous avons constaté que les équations régissant le comportement mécanique d'un corps pouvaient être ramené à une EDO d'ordre 2.

Celles-ci doivent être décomposées en EDO d'ordre 1 (comme l'équation (2.3) en le système (2.4)). Nous avons présenté un seul exemple qui était directement mis sous la forme d'un système d'EDO d'ordre 1 : le système composé de (1.2) et de (1.3) qui décrit la mécanique d'un solide. Quoiqu'il en soit, nous pouvons en tirer des éléments de rassemblement ; en effet, nous pouvons caractériser l'état d'un système mécanique par des variables en positions et des variables en vitesses. Quand on utilise la mécanique du point, chacun de ceux-ci est caractérisé par 3 variables en position – sa position dans l'espace – et les 3 variables en vitesse qui en découlent. Pour un solide, il nous faut 7 variables en position – la position de son centre de gravité et le quaternion qui définit son orientation – et 6 variables en vitesse – la vitesse du centre de gravité et le vecteur rotation instantanée. De plus, ce formalisme de variable en position et en vitesse n'est pas limité à la mécanique : des parallèles peuvent être faits dans d'autres domaines.

Nous avons donc décidé de représenter l'état du système par un vecteur d'état qui comporte deux parties : une partie avec les variables en position et une autre avec les variables en vitesse. Toutes les manipulations se feront sur ce vecteur d'état. Chaque corps mécanique agira sur une partie de ce vecteur d'état, laquelle constitue les variables qui décrivent le corps.

Nous avons implanté ces parties positions et vitesses par des tableaux dynamiques. Chaque corps connaît les indices dans le tableau des variables qui le caractérisent. Ainsi les corps ont chacun leur zone propre dans le vecteur d'état. Maintenant il reste à voir ce dont une méthode de résolution d'EDO a besoin pour produire son résultat.

3.1.2 Définition des EDO

Le système que nous avons à résoudre est de la forme :

$$\begin{aligned} \dot{x} &= f_1(x,v) \\ \dot{v} &= f_2(x,v) \end{aligned} \tag{3.1}$$

Les fonctions f_1 et f_2 dépendent de l'équation à résoudre, donc du(des) modèle(s) mécanique(s) employé(s). Ce sont donc les corps mécaniques qui doivent fournir les méthodes de calculs de ces fonctions. Chacun de nos objets représentant un corps mécanique possède une méthode *CalculVitesse* (qui correspond à f_1) et une méthode *CalculAcceleration* (qui correspond à f_2). Ces méthodes permettent de calculer les valeurs pour un vecteur état donné. Notre implémentation en C++ est basée sur la classe abstraite ci-dessous, dont tous les corps physiques doivent hériter.

```
class CorpsPhysique
{
  public:
    virtual ~CorpsPhysique(void){}

//Interface pour la resolution d'EDO
    virtual void CalculVitesse(VecteurEtat &,VecteurEtat &)=0;
    virtual void CalculForces(VecteurEtat &,VecteurEtat &)=0;

//Positions des coordonnées dans le vecteur d'etat
    int DebX;
    int FinX;
    int DebV;
    int FinV;
};
```

Pour une méthode de résolution d'EDO, il faut pouvoir appeler ces méthodes pour tous les corps physiques à simuler. Il existe déjà une centralisation de ces corps pour pouvoir mener différentes tâches comme par exemple la détection de collision. C'est à cet objet centralisateur que l'on a ajouté aussi les mêmes méthodes *Calcul** qui se chargent de faire le parcours sur tous les corps. Nous avons implémenté une classe abstraite (voir ci-dessous) pour donner un cadre à cette centralisation. La méthode *AjouteCorpsPhysique* sert à enregistrer les corps qui doivent être simulés.

```
class SystemeMecanique
{
  public:
    virtual ~SystemeMecanique(void) {}
    virtual void CalculVitesse(VecteurEtat &,VecteurEtat &)=0;
    virtual void CalculAcceleration(VecteurEtat &,VecteurEtat &)=0;

    virtual void AjouteCorpsPhysique(CorpsPhysique &)=0;
};
```

Cette utilisation d'un vecteur d'état avec des variables en position et en vitesse ainsi que la spécialisation des fonctions de calcul des efforts et des vitesses déportés dans les objets représentants les corps physiques est la clé de notre système générique de traitement des EDO. Nous ne l'avons appliqué qu'à des systèmes mécaniques mais d'autres systèmes peuvent être envisagés sans modification. Enfin, nous avons également défini ce que doit être une méthode

de résolution d'EDO (voir ci-après). Le membre *Mecasys* sert à obtenir les fonctions à intégrer et la méthode *PasInteg* calcule le pas suivant.

```
class Integrateur
{
  public:
    Integrateur(SystemeMecanique *mecasys,double pas)
      :Pas(pas){MecaSys = mecasys;}
    virtual ~Integrateur(void) {}
    virtual void PasInteg(VecteurEtat &, VecteurEtat &)=0;

  protected:
    double Pas;  // Pas d'intégration
    SystemeMecanique *MecaSys;
};
```

3.1.3 Gestion des collisions

Nous avons décrit comment SPORE gère ses collisions (voir § 1.2.1.1) par l'intermédiaire d'une méthode à pénalité, basée sur une approximation par sphères des corps. L'évaluation des collisions est une phase coûteuse : il faut placer les corps dans la grille de voxels, tout en évaluant les couples de sphères en collisions. Ces couples servent à déterminer les forces de collision.

Certaines méthodes explicites nécessitent des évaluations intermédiaires. Il se pose alors la question de la gestion des collisions : doit-on les réévaluer ou peut-on conserver l'état initial ? Pour que les collisions soient réévaluées, il faut reprendre le processus de placement des objets dans la grille. Cela représente un temps de calcul assez long, surtout pour notre objectif d'animation interactive.

C'est pourquoi nous voulons ne pas déterminer à nouveau les couples de sphères de collisions. Cette liste est construite au début du pas de temps et les forces de collisions seront évaluées à partir de celle-ci. Les forces seront donc mises à jour dans les évaluations intermédiaires mais les couples en collisions n'évolueront pas.

Ceci a été baptisé « Roadrunner physics » dans [Bar95] , en hommage au personnage de dessin animé qui ne prend en compte l'absence de sol qu'à l'instant suivant. Le fait de figer l'état des collisions semble gênant ; cependant, nous simulons avec un pas de temps de l'ordre de la dizaine de ms et donc l'erreur commise passe rapidement inaperçue.

Nous garderons donc l'état des couples en collisions constant sur un pas, seule l'intensité des forces sera modifiée dans les évaluations intermédiaires. Il en sera de même lorsque nous utiliserons les méthodes implicites (chapitre 4) qui nécessitent elles aussi des évaluations internes.

3.2 Expérimentation avec des méthodes explicites

Nous avons voulu faire une étude sur les méthodes explicites parce qu'elles sont peu coûteuses en calcul et sont donc attractives dans une utilisation temps réel. Il nous faut donc vérifier qu'elles sont adaptées à nos simulations. Nous avons basé nos tests sur les méthodes d'Euler (le normal et le modifié), de Runge-Kutta d'ordre 4 et les méthodes d'Adams (Bashforth et Moulton PECE) à différents ordres de précision. On peut souligner que la méthode de Runge-Kutta nécessite 4 évaluations de f, les méthodes PECE 2 et toutes les autres 1 seule.

Nous avons choisis ces méthodes pour les raisons suivantes. Les méthodes d'Euler sont les plus simples et sont couramment utilisées. La méthode RK4 se situe à une charnière des RKE : c'est la méthode d'ordre le plus élevé avec un nombre d'étages égal à cet ordre. L'emploi de méthodes plus précises augmentent alors beaucoup le coût de calcul. Enfin, les méthodes multipas offre l'avantage de conserver le même nombre d'évaluations de f quelque soit la précision voulue.

Nous rappelons que nous gérons les collisions par pénalité et dans les exemples qui utilisent l'environnement gynécologique, la cavité et les différents corps (autres que ceux expressément indiqués comme étant simulés) sont des objets immobiles qui n'interviennent que dans la détection de collisions.

3.2.1 Solide Rigide

Figure 3.1 : Manipulation d'un pavé

Ce premier test est la mise en œuvre de la mécanique des solides. Nous avons simplement laissé un parallélépipède rectangle tomber dans la cavité abdominale féminine que nous utilisons pour SPIC (voir Figure 3.1). Cet environnement est intéressant pour les interactions de collisions qu'il procure. Une fois le pavé stabilisé sur le fond de la cavité, nous avons utilisé un outil pour interagir avec lui. Cet outil est une pince telle qu'utilisent les gynécologues lors de leurs opérations. On se sert de ces pinces pour déplacer le pavé dans la cavité en le poussant et pour tenter de le saisir.

La simulation d'un objet rigide se passe assez bien pour toutes les méthodes. Les problèmes de divergence peuvent apparaître lors de collisions. Dans SPORE, elles sont gérées par pénalités et donc il y a introduction d'une force semblable à un ressort qui possède une raideur assez grande (suffisamment pour assurer une interpénétration minime), source d'instabilité. De plus, les interactions avec les objets étant gérées de la même manière (voir §1.2.1.3), l'utilisation de l'outil peut apporter les mêmes problèmes.

Des problèmes peuvent survenir si les objets percutent à trop grande vitesse un obstacle. Pour caractériser l'ordre de grandeur de la vitesse limite, on peut prendre le problème suivant : la chute d'un objet sur un plan horizontal. Si on considère que le déplacement sur un pas de temps est uniquement linéaire et constitue la pénétration dans l'obstacle, alors la force générée pour contrer la collision est égale à $-k_{collis}vh$ (h est le pas de temps de la méthode d'intégration). Pour que la force de collision reste dans le domaine de stabilité, il suffit donc que :

$$\left(\frac{k_{collis}vh}{m}-g\right)h<R \Leftrightarrow \boxed{v<\frac{m}{k_{collis}h}\left(g+\frac{R}{h}\right)} \tag{3.2}$$

R représente le rayon de stabilité (voir § 2.3.2). On voit clairement que le rapport k_{colli}/m conditionne la raideur du problème et qu'un pas de temps faible sert à diminuer ce facteur d'instabilité. On peut déterminer une valeur adéquate pour k_{collis} en fonction de l'interpénétration que l'on autorise quand l'objet est à l'équilibre :

$$mg=k_{collis}d \Leftrightarrow \boxed{k_{collis}=\frac{m}{d}g} \tag{3.3}$$

En insérant les caractéristiques liées à la simulation que l'on veut effectuer, on obtient les valeurs limites. Par exemple, dans le cas de SPIC, on manipule essentiellement les ovaires : un ovaire pèse de l'ordre de quelques grammes, nous allons prendre 5g pour l'exemple. Dans notre simulation, on peut estimer que l'interpénétration ne doit pas excéder 0.1mm (un ovaire moyen fait environ 2 cm dans sa plus grande largeur). On obtient alors que $k_{collis}=500$; en choisissant un pas de temps de 10ms, on arrive à $v<0.1R+0.01$. Cela donne pour Euler explicite $v<0.21ms^{-1}$. On voit que la limitation en vitesse est assez forte. Cela reste raisonnable pour une simulation chirurgicale mais peut néanmoins poser des problèmes par ailleurs.

Lorsque le pavé est lâché dans la cavité, la simulation se comporte bien quelque soit la méthode. Les problèmes surviennent surtout lors de l'interaction avec l'outil. Il convient de respecter les mêmes limitations de vitesse or il est difficile d'imposer cela à l'utilisateur. Il est vrai qu'un chirurgien ne fera pas de mouvement rapide mais un étudiant ne sera pas forcément aussi précautionneux dans sa phase d'apprentissage.

Par contre, il y a divergence systématique lors de la saisie d'un objet : il est possible de fermer les pinces totalement sur l'objet par manque de retour d'effort. Il est évident que cette situation n'est pas « physiquement réaliste » mais elle peut se produire dans la simulation. Une

solution possible est l'utilisation d'une technique appelé God Object ([DZ93]). Celle-ci consiste en la gestion d'une copie de l'outil réel : il y a donc l'outil dont la position est dictée par l'utilisateur et une version simulée de cet outil. Ainsi, la version simulée va subir les interactions de la scène : nos pinces ne pourront plus se refermer sur un objet.

Les deux versions de l'outil doivent être liées de telle sorte que la position de l'outil simulé tende vers la position de l'outil réel. On peut voir cette liaison comme un asservissement de position, problème bien connu en automatique. De plus, le God Object peut être mis à profit pour calculer l'effort à renvoyer à un dispositif haptique : c'est la différence de position entre ces outils qui sert à définir le retour d'effort ([ZS95]). Au final, cette technique est utile avec ou sans présence de retour d'effort et surtout permet d'éviter des situations délicates pour la résolution des EDO.

3.2.2 Modèle Déformables Discrets

Les différents tests qui suivent constituent une frange importante de ce que SPORE doit pouvoir simuler : des objets déformables. Tous les modèles présentés ici sont des modèles discrets : un système particulaire et des maillages masses/ressorts. Ces derniers représentent la manière la plus simple de modéliser un objet déformable et ils fournissent des comportements relativement acceptables.

Modèles Particulaires

Le principe est de simuler un ensemble de points qui subissent les interactions des autres. Nous utilisons un tel modèle pour simuler un flux sanguin : chaque particule représente une « goutte » de sang. Des forces de Lennard-Jones régissent le comportement des particules entre elles et l'affichage est effectué par une surface implicite permettant une fusion visuelle des « gouttes ».

Une source de particules est activée lors du contact de l'outil avec un quelconque endroit de la cavité. Dès lors, les particules sont éjectées : leur vitesse initiale est non-nulle et les chocs sont nombreux. Ces conditions font que seule la méthode de RK4 donne des résultats satisfaisants ainsi que la méthode d'Euler modifiée dans une moindre mesure. Les autres méthodes ne permettent même pas un affichage temporaire puisque dès le départ les particules sont envoyées à l'infini.

Il faut aussi ajouter que même la méthode RK4 n'est pas exempte de divergence. Certaines particules « disparaissent » : celles qui sont soumises à de fortes pressions, coincées au milieu de plusieurs de leurs congénères, etc. Cependant le nombre de celles qui sortent du champ de vision reste faible par rapport à l'ensemble. Comme le nombre de particules que nous simulons est fini, il peut être gênant de perdre un grand nombre de particules. De plus, la gestion de celles qui quittent l'espace utile occupe du temps de calcul. Pour remédier à ces deux problèmes, nous arrêtons la simulation de toute particule qui sort de cet espace utile et elle est remise dans le tas des particules à émettre.

Grâce à ce garde-fou, on peut dire que la méthode de RK4 suffit pour ce problème. Il est à noter que cette modélisation est loin d'être réaliste et qu'une simulation approximative suffit amplement. Par contre, les autres méthodes sont inutilisables. Cet état de fait s'explique par la vitesse initiale des particules qui est trop élevée. De plus, par la configuration initiale, une particule émise se trouve déjà en interaction avec l'environnement : cette mise en action en subissant immédiatement des efforts aggrave l'instabilité provoquée par la vitesse.

Maillages Masses/Ressorts

Parmi les modèles d'objets déformables, le maillage masses/ressort est probablement le plus populaire : il est relativement simple à mettre en œuvre. Nous avons entrepris deux tests simples : la chute d'un maillage 2D et d'un maillage 3D sur un plan (voir Figure 3.2). Le but est de voir l'influence de la taille du maillage sur la stabilité de la simulation. Le contact avec le plan permet de voir la stabilité de ce modèle face à une collision.

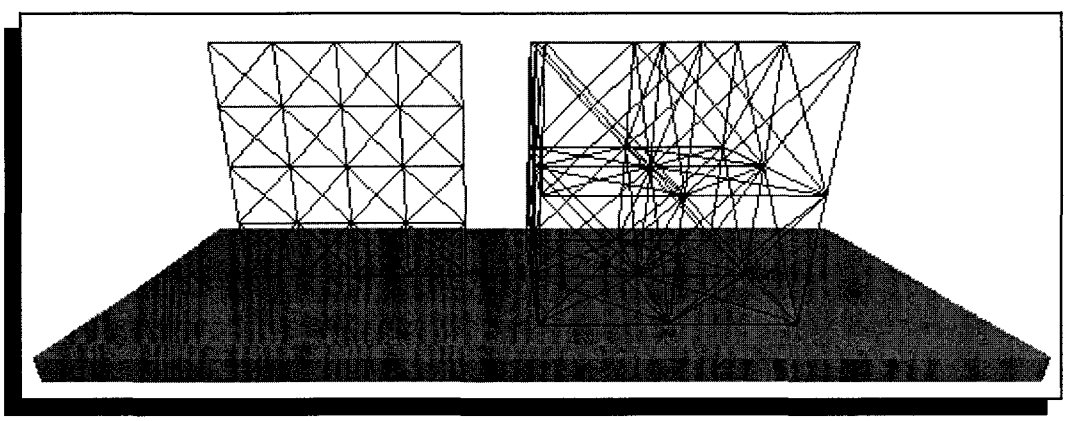


Figure 3.2 : les deux types de maillages testés

Les paramètres importants de ce type de maillage est la masse de chacun des nœuds et la raideur des ressorts de liaisons. Lorsque l'on étudie un oscillateur simple, c'est le rapport k/m qui conditionne le problème. Il est clair que ce rapport est toujours aussi important mais il faut aussi tenir compte des effets liés au maillage.

Il est à noter que l'utilisation courante d'un tel maillage apporte aussi de gros défauts. En premier lieu, la masse totale de l'objet approximé est fixe et ainsi lorsque l'on augmente le nombre de points, la masse associée à chacun d'entre eux diminue d'autant. Cela tend à faire augmenter le rapport k/m, ce qui est une source d'instabilité. Ensuite vient le problème de la similitude de raideur globale. Pour comprendre, prenons l'exemple d'un ressort de raideur k avec une extrémité fixe ; si l'on applique un effort à l'autre extrémité, la position d'équilibre induit un déplacement d. Ensuite, on subdivise ce ressort en deux en introduisant en son milieu un nouveau point. Si on veut obtenir ce même déplacement d en appliquant le même effort, il faut alors utiliser deux ressorts de raideur $2k$. Ceci contribue aussi à augmenter le rapport k/m.

Notre démarche est de dire que lorsque l'on veut simuler un objet déformable, sa masse est une caractéristique simple à obtenir contrairement à ses caractéristiques de défor-

mations. Nous avons donc orienté nos tests de la façon suivante : la masse de l'objet étant fixée, quelle raideur maximale peut supporter la méthode d'intégration ?

Nous avons donc fixé la masse totale de l'objet à 1 kg qui se répartit sur les points du maillage. Puis nous avons déterminé empiriquement les valeurs limites au delà desquelles les maillages divergeaient pour une valeur du pas d'intégration de 10 ms. Les différents tableaux qui suivent récapitulent ces informations pour la méthode d'Euler modifiée et pour la méthode de RK4. Les autres méthodes explicites (Euler, Adams-Bashforth et PECE) ne sont pas reprises car elles apportent des résultats très mauvais avec des valeurs de raideurs très faibles.

Nombre de Points	Masse d'un point	Raideur Maximale	Rapport k/m
4	250g	2400	9600
9	111,1g	750	6750
16	62,5g	340	5440
25	40g	150	3750

Table 3.1 :Maillage 2D, méthode Euler modifié

Nombre de Points	Masse d'un point	Raideur Maximale	Rapport k/m
4	250g	5000	20000
9	111,1g	1600	14400
16	62,5g	770	12320
25	40g	460	11500
36	27,8g	300	10800

Table 3.2 : Maillage 2D, méthode RK4

Nombre de Points	Masse d'un point	Raideur Maximale	Rapport k/m
8	125g	750	6000
27	37g	100	2700
64	15,6g	20	1280

Table 3.3 : Maillage 3D, méthode Euler modifié

Nombre de Points	Masse d'un point	Raideur Maximale	Rapport k/m
8	125g	1650	13200
27	37g	200	5400
64	15,6g	45	2880

Table 3.4 : Maillage 3D, méthode RK4

Aux valeurs de raideur limite, le maillage est stable lors de la chute mais subit des phénomènes de vibrations qui s'arrêtent difficilement à cause du contact avec le plan. Il est à signaler que dès que le rapport k/m devient inférieur à 5000, le tissu ne peut plus conserver sa structure lors du choc avec le plan. On peut souligner que la méthode de RK4 s'en tire assez bien mais on constate bien que la discrétisation oblige à diminuer toujours plus la raideur. On voit donc que cela limite le nombre de points que l'on peut utiliser pour décrire un corps. On constate aussi que le maillage 3D subit cette loi de manière un peu plus contraignante. Ce

maillage comporte plus de liaisons à nombre de points similaire. Cela illustre le fait que la répartition des ressorts induits des comportements additionnels qu'il faut pouvoir maîtriser.

Tissu

Nous avons voulu essayer les méthodes avec un modèle masses/ressorts un peu différent : il résulte du travail de DEA de J. Davanne ([Dav00]) qui se base sur le modèle de X. Provot ([Pro95]). Il est difficile de vouloir faire une étude similaire à la précédente car la modélisation fait intervenir 3 sortes de ressorts qui possèdent des raideurs différentes. Nous nous sommes plutôt attachés ici à obtenir des informations sur le comportement du modèle dans sa plage de non divergence.

La 1ère scène test (Figure 3.3) est le lâché d'un carré de tissu dans la cavité abdominale et la manipulation du tissu grâce aux pinces. Cela permet de voir le comportement dans une situation moins « académique » que précédemment, surtout en présence d'interactions complexes. En effet, les collisions avec le fond de la cavité sont irrégulières. On constate que le tissu n'arrive jamais vraiment à trouver une position d'équilibre : il se place dans une position autour de laquelle il vibre. La manipulation est délicate car la vitesse des pinces doit rester basse pour éviter une divergence.

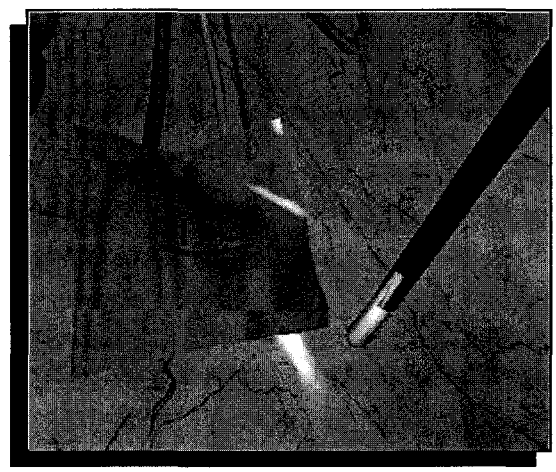


Figure 3.3 : Un morceau de tissu dans la cavité abdominale

La 2ème scène test (Figure 3.4) est un drap accroché par les 4 coins dans lequel tombent des pavés (identiques à celui du test du §3.2.1). De plus, les pinces sont aussi présentes pour permettre de manipuler l'ensemble des objets. On retrouve le même phénomène vibratoire que précédemment, légèrement plus prononcé à cause des mouvements des pavés contenus dans le tissu.

3.3 BILAN

Les tests avec différentes méthodes explicites que nous avons effectués se sont révélés acceptables dans certains cas comme une simulation de solide simple. Dans le cas d'objets déformables structurés, la méthode d'Euler et les méthodes multipas explicites donnent de très mauvais résultats ; seules la méthode d'Euler modifié et la méthode RK4 s'en tirent assez bien. Cependant, elles restent uniquement utilisables dans une plage de valeurs restreinte.

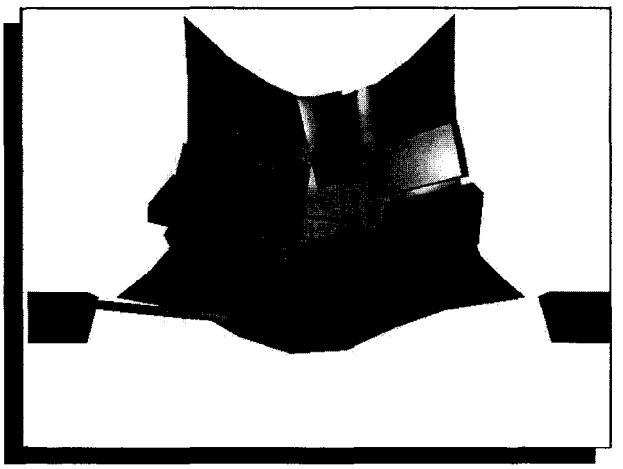


Figure 3.4 : Un drap tendu

Nous avons uniquement utilisé un maillage masses/ressorts pour les tests avec un objet déformable. Le maintien de la structure nécessite des raideurs de ressorts assez importantes qui provoquent les instabilités dans la résolution. Si on emploie un autre modèle mécanique, il devra lui aussi générer des efforts importants pour maintenir la structure du corps.

On voit donc que la stabilité est primordiale pour la simulation d'objets déformables structurés. Pour augmenter cette stabilité, on peut utiliser des méthodes de RK d'ordre supérieur mais elles nécessiteront encore plus d'évaluations de f pour seulement repousser les limites. Il apparaît clair que la solution consiste à se tourner vers les méthodes implicites.

Chapitre 4 : Une méthode rapide d'intégration implicite

Nous avons constaté lors du précédent chapitre que les méthodes explicites divergent très facilement et sont dès lors très délicates à utiliser pour la simulation d'objets déformables. Nous avons vu au travers des chapitres 2 et 3 que seules des méthodes implicites peuvent apporter la stabilité nécessaire. Celles-ci sont plus complexes car elles nécessitent la résolution d'un système d'équations non-linéaires. Cette opération mérite d'être étudiée car elle tend à augmenter fortement le temps de calcul d'un pas d'intégration.

Nous tenons à préciser que notre objectif est d'avoir une méthode aussi générale que possible, i.e. capable de résoudre n'importe quelles équations. Cela implique notamment de ne pas utiliser de simplifications liées à un formalisme mécanique (comme l'expression de la jacobienne par exemple).

4.1 Vers une résolution implicite d'une EDO

4.1.1 Résolution de systèmes non-linéaires

Une fois la méthode choisie, elle induit un système non-linéaire à résoudre c'est-à-dire trouver la valeur X^*, solution de l'équation $F(X)=0$, X étant un vecteur d'état comme défini dans le §3.1. Par exemple, la méthode d'Euler implicite appliquée au problème $\dot{X}=f(X)$ donne l'équation suivante à résoudre :

$$F(X)=X-X_t-hf(X) \tag{4.1}$$

Ainsi la valeur telle que $F(X)=0$ est la valeur $X_{t+\Delta t}$ qui constitue le pas d'intégration.

La méthode la plus simple pour obtenir une solution est d'utiliser la méthode du point fixe. Il faut formuler le problème pour le mettre sous la forme :

$$\tilde{F}(X)=X \tag{4.2}$$

d'où le nom de point fixe. Pour la méthode d'Euler implicite, cela donne simplement

$$\tilde{F}(X)=X_t+f(X) \tag{4.3}$$

On construit alors une suite de la façon suivante :

$$X_{n+1}=\tilde{F}(X_n) \quad X_0=X_t \tag{4.4}$$

Cette suite converge si $\tilde{F}$ est contractante dans un domaine autour de la solution ([Kel95]) et si le point de départ se situe dans cet intervalle.

Cette méthode est plaisante car toute méthode d'intégration est formulée de cette façon. Le problème est que cela rend « explicite » la méthode ; les méthodes PECE sont en fait une application de cette méthode en limitant strictement le nombre d'itérations et les tests du chapitre 3 ont donné des résultats décevants. [LW70] montre qu'il faut proscrire cette méthode et utiliser une méthode de type Newton.

La méthode de Newton pour la résolution de systèmes non-linéaires se base sur un principe simple : approcher la fonction par sa « pente » locale, calculer la solution de ce problème approché et recommencer jusqu'à convergence. La Figure 4.1 montre ce principe sur un exemple en dimension 1. La formulation de la méthode est la suivante :

$$X_{n+1} = X_n - J^{-1}(X_n) F(X_n) \tag{4.5}$$

avec $J = \partial F / \partial X$ représentant la matrice Jacobienne. Ainsi, chaque itération de Newton est constituée du calcul de cette matrice et son inversion. On dispose d'algorithmes pour effectuer cette inversion mais elle représente un coût important. Afin de réduire celui-ci, il est préférable de résoudre le système linéaire suivant :

$$J(X_n) s = F(X_n) \tag{4.6}$$

s représente alors l'incrément qu'il faut ajouter à X_n pour obtenir X_{n+1}.

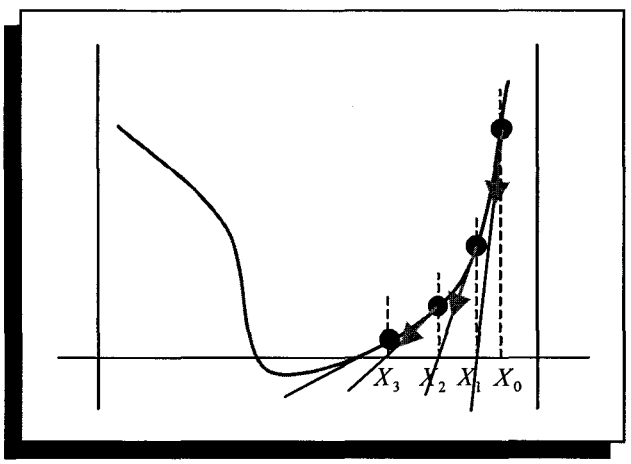


Figure 4.1 : Itérations de Newton

La fonction à annuler et le point initial doivent satisfaire certaines conditions pour que la suite converge. La jacobienne doit être une fonction lipschitzienne et la jacobienne au point solution ne doit pas être singulière. De plus, l'estimation initiale doit être suffisamment proche de la solution. Cette imprécision du positionnement nécessaire de l'estimation initiale pose problème lorsque l'on ne sait absolument pas où se situe la solution. Nous verrons ce problème dans la partie 4.3.2.

Sous ces conditions, non seulement les itérations de Newton convergent mais en plus elles convergent quadratiquement, c'est-à-dire :

$$\|e_{n+1}\| \leq K \|e_n\|^2 \quad avec \quad e_n = X_n - X^* \tag{4.7}$$

Cette propriété est importante car elle indique que lorsque l'on se trouve assez proche de la solution (e_n petit), la suite va converger très vite. On peut assouplir les conditions de convergence en supprimant l'obligation de non-singularité de la jacobienne de la solution ; dans ce cas, on perd alors la convergence quadratique pour revenir à une simple convergence linéaire.

Il faut aussi pouvoir déterminer à quelle itération le calcul peut être arrêté. A cette fin, on procède à l'évaluation de F au point considéré qui représente le résidu du système. C'est une mesure de l'éloignement par rapport à la solution. Il faut tout de même préciser que cela n'est pas forcément une bonne mesure : dans le cas d'une fonction qui s'annule en un point plat (voir Figure 4.2), on peut être loin de la solution tout en ayant une valeur de F proche de 0. Il est montré ([Kel95]) que cette mesure est bonne à condition que la jacobienne de la solution soit bien conditionnée[10]. On peut donc arrêter les itérations en fixant au préalable une précision par rapport au résidu du système.

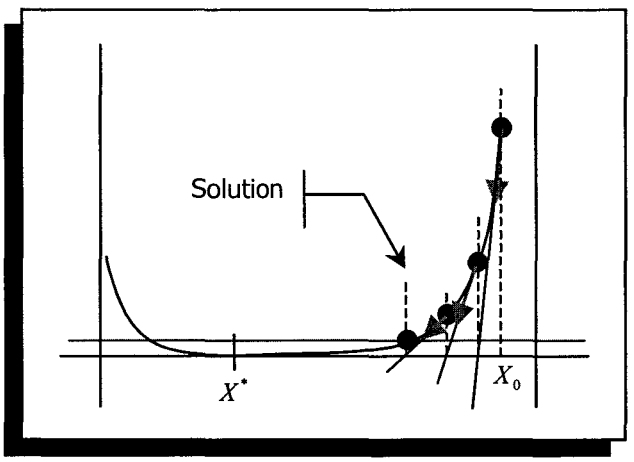


Figure 4.2 : Problème de l'utilisation du résidu comme mesure d'arrêt des itérations

Enfin deux variantes de la méthode de Newton sont couramment utilisées dans le but de diminuer le nombre d'évaluations de la jacobienne. En effet, le calcul de celle-ci reste quelque chose de coûteux et numériquement instable (voir partie 4.1.2.1). La première est la plus utilisée : c'est la méthode de la corde. Elle consiste à évaluer la jacobienne à l'estimation de départ et de conserver cette matrice pour les itérations suivantes (voir Figure 4.3). Cela permet d'utiliser des techniques de précalcul pour la résolution du système linéaire. On perd évidemment la vitesse de convergence quadratique. Ainsi le nombre d'itérations augmente mais le coût en calcul d'une itération devient vraiment faible.

La 2ème variante est un moyen terme entre Newton et la méthode de la corde : c'est la méthode de Shamanskii. Elle consiste à utiliser des itérations identiques à celles de la corde mais en mettant à jour la jacobienne toutes les m itérations. Cela permet de diminuer le nombre d'itérations par rapport à la méthode de la corde au prix de calculs supplémentaires, en restant cependant moins coûteux que la méthode de Newton. L'autre intérêt est l'augmentation de la vitesse de convergence : l'erreur entre deux itérations où la jacobienne est réévaluée évolue de la façon suivante :

$$\|e_{n+1}\| \leq K \|e_n\|^m \tag{4.8}$$

[10] On appelle $\kappa = \|A\| \|A^{-1}\|$ le nombre condition de la matrice A. La matrice est d'autant mieux conditionnée que κ est proche de 1.

Le tout est de trouver un compromis entre le nombre d'évaluations de la jacobienne et le coût en calcul de celle-ci. D'après [Bre73], il résulte que la méthode de la corde est souvent la meilleure solution, lorsque l'estimation initiale est proche de la solution et que le système est grand.

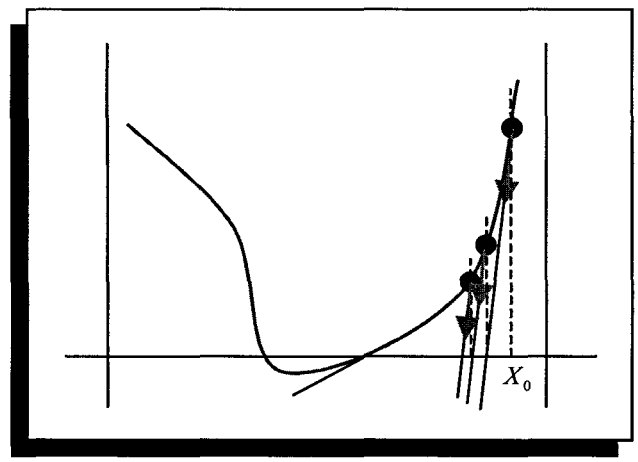


Figure 4.3 : Itérations de la méthode de la corde

4.1.2 Implantation

4.1.2.1 Expression de la Jacobienne

Nous avons vu qu'une étape de la résolution du système non-linéaire comprend une évaluation de la matrice jacobienne de F en un point donné. Deux choses sont à prendre en compte en ce qui concerne cette matrice : son stockage et son calcul.

Si le vecteur d'état qui décrit le système mécanique simulé est constitué de n variables, la matrice jacobienne est alors une matrice $n \times n$. Les scènes simulées avec un objet déformable contiennent régulièrement plusieurs centaines de variables et des scènes complexes plusieurs milliers. Cette matrice tient donc une place importante en mémoire et surtout sa manipulation lors des calculs ultérieurs (résolution du système linéaire) est un facteur important dans le coût en calcul.

Si la matrice jacobienne est dense, il n'y a rien à faire : il faut stocker la matrice in extenso. Par contre, si elle possède une structure particulière, il est fortement conseillé d'utiliser une représentation adéquate. Il arrive que cette matrice soit creuse : une structure de donnée dédiée permet alors, non seulement de réduire la taille en mémoire, mais aussi de réduire le coût en calcul des opérations classiques, comme une multiplication matrice vecteur, par rapport à une matrice dense.

L'autre partie est le calcul effectif de cette jacobienne. Lorsqu'une expression analytique existe, il est clair que le processus est simplifié et en général relativement peu coûteux. On peut dire qu'en général, on peut obtenir une telle expression quand on utilise un modèle en particulier. Par exemple, [SDB99] obtient une telle expression, qui est même constante grâce à d'autres simplification. Mais il est difficile d'avoir une telle expression analytique avec tous les modèles mécaniques.

Afin d'obtenir une expression de la jacobienne, on peut utiliser un calcul par différentiation : le but est d'obtenir la matrice jacobienne par un calcul purement numérique. Il repose sur la définition de la matrice jacobienne

$$J = \left(\frac{\partial F}{\partial X_1} \; \frac{\partial F}{\partial X_2} \; \cdots \; \frac{\partial F}{\partial X_n} \right) \tag{4.9}$$

On utilise alors une approximation au premier ordre de la dérivation :

$$\frac{\partial F}{\partial X_i} = \frac{F(X + he_i) - F(X)}{h} \tag{4.10}$$

e_i représente les vecteurs de la base canonique de $\mathbb{R}^n$.

Il est à noter que l'utilisation des (e_i) constitue déjà un choix car c'est une direction de différentiation parmi une infinité de possibilités. De plus, il faut déterminer quelle valeur utiliser pour h. Il faut qu'elle soit suffisamment petite pour obtenir une valeur correcte mais il faut tenir compte de l'erreur due à la représentation interne des nombres d'un ordinateur. Dans [Kel95], il est montré que l'erreur commise est de l'ordre de $O(h + \varepsilon / h)$ avec ε l'erreur d'arrondi maximale et donc le choix de $h = \sqrt{\varepsilon}$ minimise le terme dans le O.

Enfin, il reste à porter attention au coût en calcul de la jacobienne. La formule (4.9) indique que cela nécessite n utilisations de (4.10). On peut considérer que l'évaluation de F en X est mise en commun et donc on aboutit à $n+1$ évaluations de F. Cela constitue un coût énorme. Il est possible de réduire ce coût comme le montre [VM00] mais cela reste dédié à un modèle mécanique particulier.

4.1.2.2 Résolution de systèmes linéaires

Une fois la jacobienne déterminée, il faut résoudre le système linéaire (4.6). Initialement, la méthode de Newton a été imaginée avec l'utilisation d'une méthode de résolution exacte mais une méthode itérative avec arrêt selon une tolérance peut être utilisée. Pour de plus amples détails sur la résolution de tels systèmes, on peut consulter ([Rou00], [Kel95]).

Les méthodes exactes fournissent un algorithme dont le coût est fixe ; on peut les séparer en deux catégories : celles qui font un calcul direct (Gauss, Jordan) et celles à factorisation (LU, QR, Choleski). Les premières sont un peu moins coûteuses pour une résolution complète. L'intérêt des 2èmes est qu'elles sont séparées en deux phases : une transformation de la matrice du système en un produit de matrices de formes particulières qui vont simplifier la phase de résolution proprement dite. Si on doit effectuer plusieurs résolutions avec la même matrice, les méthodes à factorisation sont alors bien plus efficaces.

Globalement, la résolution complète possède une complexité en n^3. Dans les méthodes à factorisation, c'est cette phase de factorisation qui porte la complexité, celle de la phase de résolution proprement dite étant quadratique. On voit clairement l'avantage des méthodes à

factorisation lors de résolutions multiples, comme quand on utilise la méthode de la corde ou Shamanskii. Il reste que bien souvent cela reste très coûteux. Ces méthodes sont tout de même utilisées dans [SDB99], car la matrice Jacobienne est considérée comme constante tout au long de la simulation et donc la factorisation est effectuée en précalcul

L'alternative est l'utilisation de méthodes itératives. Elles ne sont pas exactes[11] mais elles permettent d'approcher la solution pour une certaine précision. De nombreuses méthodes existent. Elles sont généralement bien plus performantes lorsque le système est grand. En effet, elles nécessitent bien moins d'itérations pour atteindre une solution dans l'intervalle de tolérance. Dans [Bar95] et [EEH00] est utilisée la méthode du gradient conjugué qui est en général la méthode la plus performante mais qui est limitée au problème dont la matrice est symétrique définie positive (SDP). Il existe des méthodes pour les problèmes qui ne remplissent pas ces conditions comme le gradient bi-conjugué, version modifiée du gradient conjugué dont le coût est double. L'utilisation de ces méthodes itératives introduit donc un $2^{ème}$ niveau d'itération dans le processus global de la résolution d'un système non linéaire.

4.2 Broyden

Notre objectif est de pouvoir résoudre le système d'équations non linéaires induit par la méthode de résolution d'EDO de façon la plus générale possible. Ainsi, nous voulons éviter toutes contraintes sur le type de modèles mécaniques (plus de matrices SDP et de linéarisation simplificatrice). De plus, la formalisation de la matrice jacobienne est une partie coûteuse et sensible numériquement. Nous avons donc cherché un moyen d'éviter cela.

4.2.1 Broyden

Les méthodes dites Quasi-Newton décrivent une catégorie de méthodes qui se servent d'itérations similaires à celles de Newton mais qui n'utilisent pas de la matrice jacobienne de F. Cela conduit à l'utilisation de la formule suivante :

$$X_{n+1} = X_n - B_n^{-1} F(x_c) \tag{4.11}$$

Il faut donc construire une suite de matrices (B_i) qui permettent la convergence vers la solution : c'est ce qui va caractériser la méthode.

La méthode de Broyden régit la mise à jour des matrices de la façon suivante :

$$B_{n+1} = B_n + \frac{F(X_{n+1})\,{}^t s_n}{{}^t s_n s_n} \quad avec \quad s_n = X_{n+1} - X_n \tag{4.12}$$

Cette formule est une extension de la méthode de la sécante, bien connue pour les problèmes en dimension 1 (voir Figure 4.4). On trouvera dans [Kel95] des détails concernant la preuve de

[11] Certaines méthodes comme le gradient conjugué peuvent être vues comme exactes mais perdent alors leur atout de rapidité.

la convergence de cette méthode. Les conditions de convergence sont identiques à celles de la méthode de Newton. La vitesse de convergence est dite superlinéaire, i.e.

$$\lim_{n\to\infty} \frac{\left\| X_{n+1} - X^* \right\|}{\left\| X_n - X^* \right\|} = 0 \tag{4.13}$$

On voit donc que la vitesse de convergence est correcte (intermédiaire entre Newton et la corde), avec un coût de calcul réduit, notamment en ce qui concerne le nombre d'évaluations de F, une seule étant nécessaire par itération « non-linéaire ».

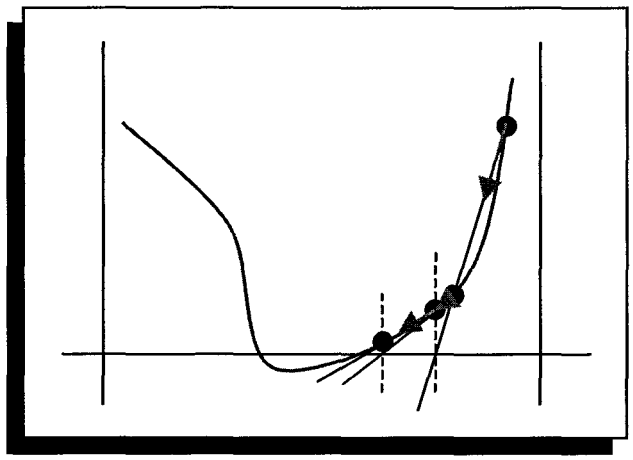


Figure 4.4 : méthode de la sécante (il faut un point de départ et une pente de départ ou deux points de départ)

Nous devons maintenant construire deux suites : celle des vecteurs (x_i) et celle des matrices (B_i). Nous avons donc besoin de deux valeurs initiales. Ainsi, il est possible d'éviter le calcul de la jacobienne mais il reste tout de même à stocker cette matrice et résoudre le système linéaire associé. Cela représente la version de base de la méthode, présente dans [PFTV93]. Cette version est améliorée par l'utilisation d'une décomposition QR (matrice orthogonale et matrice triangulaire) ; en effet, il existe des formules directes de mise à jour des deux matrices. Il est à noter également que même si la suite a pour but l'approximation de la jacobienne, elle ne converge pas obligatoirement vers cette jacobienne.

Une amélioration de la méthode est de construire directement la suite des inverses de (B_i) : de cette manière, on évite le calcul de l'inversion. Pour construire cette suite, il faut pouvoir exprimer B_{n+1}^{-1} en fonction de B_n^{-1} et des autres données disponibles. A cette fin, on détermine une expression analytique de l'inverse grâce à la formule de Sherman-Morisson :

$$\left(B + u\,{}^t v\right)^{-1} = I - \left(\frac{\left(B^{-1}u\right){}^t v}{1 + {}^t v B^{-1} u} \right) B^{-1} \tag{4.14}$$

Il suffit de considérer que l'itération de Broyden s'écrit de la façon suivante :

$$
\begin{aligned}
B_{n+1} &= B_n + u_n {}^t v_n \\
u_n &= F(x_{n+1}) / \|s_n\| \\
v_n &= s_n / \|s_n\|
\end{aligned}
\tag{4.15}
$$

Grâce à cette formulation, nous avons maintenant la possibilité d'obtenir directement l'inverse des B_i. A ce stade, nous avons donc non seulement évité le calcul de la jacobienne mais aussi l'utilisation d'une méthode de résolution de système linéaire.

Un dernier résultat permet de réduire encore le coût de calcul et par la même occasion facilite l'initialisation de l'algorithme. Pour construire la suite des B_i, il faut avoir à disposition une matrice de départ. Dans [Kel95], il est établi que l'on peut toujours prendre la matrice identité comme valeur pour B_0. En utilisant ceci on peut réécrire (4.15):

$$
\begin{aligned}
B_n^{-1} &= \prod_{j=0}^{n-1} \left(I - w_j {}^t v_j \right) \\
v_i &= s_i / \|s_i\| \\
w_i &= \left(B_i^{-1} u_i \right) / \left(1 + {}^t v_i B_i^{-1} u_i \right)
\end{aligned}
\tag{4.16}
$$

En poursuivant les calculs, on peut obtenir une formulation ne comprenant que des vecteurs s_i :

$$
\begin{aligned}
B_n^{-1} &= \prod_{j=0}^{n-1} \left(I + \frac{s_{j+1} {}^t s_j}{\|s_j\|^2} \right) \\
s_n &= B_n^{-1} F(X_n)
\end{aligned}
\tag{4.17}
$$

Pour aboutir à (4.17), il a fallu imposer que B_0 soit égale à la matrice identité. Pour le cas où on aurait à disposition une estimation de la jacobienne qui soit relativement peu coûteuse à calculer, il aurait été intéressant de l'utiliser. Cela reste possible mais il faut alors l'utiliser en l'incluant dans l'évaluation de F.

4.2.2 Implantation

A partir de (4.17), il n'est pas encore tout à fait possible de réaliser une implémentation de la méthode de Broyden. En effet, s_n intervient dans le calcul de B_n^{-1} et vice versa. On obtient donc une équation en s_n qu'il faut isoler pour obtenir :

$$
s_n = -\frac{B_{n-1}^{-1} F(X_n)}{1 + {}^t s_{n-1} B_{n-1}^{-1} F(X_n) / \|s_{n-1}\|^2}
\tag{4.18}
$$

La calcul de s_n fait donc intervenir la matrice B_{n-1}^{-1} dans son produit avec $F(X_n)$; il faut alors construire cette matrice grâce à (4.17), laquelle utilise les valeurs précédentes de (s_i). Il faut alors effectuer le produit matrice vecteur avec $F(X_n)$. Il serait souhaitable d'éviter l'explicitation des matrice (B_i) pour limiter l'encombrement mémoire ; de plus, comme ces matrices sont reconstruites à partir des (s_i), il serait avantageux de n'avoir à stocker que ces vecteurs.

Pour cela, nous allons calculer directement le vecteur $B_{n-1}^{-1}F(X_n)$. Pour cela, on définit la suite suivante :

$$\begin{aligned} z_0 &= F(X_n) \\ z_i &= z_{i-1} + \frac{s_i{}^t s_{i-1} z_{i-1}}{\|s_{i-1}\|^2} \end{aligned} \tag{4.19}$$

On remarque que z_{n-1} est égal à $B_{n-1}^{-1}F(X_n)$. L'intérêt de cette formulation est le réarrangement par associativité du terme $s_i{}^t s_{i-1} z_{i-1}$ en calculant $s_i({}^t s_{i-1} z_{i-1})$. De cette façon, on ne fait plus intervenir de matrice. De plus, ce calcul d'un terme de cette suite peut s'implanter en n'utilisant qu'un seul vecteur. Au final, on aboutit à l'algorithme suivant :

$X = X_0$
$s_0 = -F(X_0)$
n=0 {numéro d'itération}
Tant Que non fini
(1) $X = X + s_n$
(2) $z = -F(X)$
Pour $i = 0$ **à** $n-1$
(3) $z = z + s_i{}^t s_{i-1} z / \|s_{i-1}\|^2$
Fin Pour
(4) $s_n = z / \left(1 - {}^t s_{n-1} z / \|s_{n-1}\|^2\right)$
(5) n = n + 1
Fin Tant Que

Pour évaluer la complexité de l'algorithme, nous allons comptabiliser les opérations flottantes nécessaires à une itération. En notant N la taille du vecteur d'état, la ligne (1) apporte N additions. La complexité de la ligne 2 est la complexité de l'évaluation de F. Au passage, on peut noter qu'il n'y a qu'une seule évaluation de F par itération. Dans les deux lignes suivantes interviennent les normes au carré des vecteurs (s_i). Pour réduire les calculs, on peut évaluer la norme au carré du vecteur s_n juste après sa détermination et conserver ce résultat. Ainsi, la ligne (4) nécessite N+1 additions, 2N multiplications, 2 divisions. La ligne (3) utilise 2N

additions, 2N multiplications, 1 division et est parcourue n fois. Ainsi, la complexité en nombre d'opérations flottantes d'une itération est en $O(Nn+Comp_F)$. Le seul élément sur lequel on peut influer est n.

Pour limiter la complexité de la méthode, il faut limiter le nombre n. On peut aussi ajouter qu'imposer une limite est nécessaire puisqu'il n'est pas possible de stocker un nombre infini de vecteurs. On fixe donc une valeur $nmax$ au delà de laquelle on arrête le calcul. Si le calcul n'a pas abouti en $nmax$ itérations, on peut reprendre le calcul depuis le début en utilisant comme estimation initiale X_0 la dernière valeur de X. Une autre stratégie est d'éliminer le vecteur s le plus ancien et donc d'effectuer ensuite les calculs sur les n vecteurs les plus récents.

Enfin, pour déterminer l'arrêt des itérations, on peut utiliser comme pour la méthode de Newton l'évaluation du résidu en imposant une tolérance. Nous obtenons donc une méthode qui demande $O(Nn)$ opérations flottantes et 1 évaluation de F par itération et qui nécessite $n+3$ vecteurs de taille N. On constate que la complexité des opérations définissant la méthode de Broyden est linéaire par rapport à la taille du vecteur d'état.

4.3 Résultats

4.3.1 Comparaison avec Newton

Nous avons voulu confronter les deux méthodes sur une équation pour dégager leurs différences de comportements. Nous avons pris le problème suivant :

$$z^3 = 1 \ \ pour \ \ z \in \mathbb{C} \tag{4.20}$$

Nous avons effectué la résolution de ce problème en faisant varier l'estimation initiale de z. Ensuite, nous avons dressé deux cartes pour chacune des méthodes : une (Figure 4.5) qui indique vers quelle racine de (4.20) ont convergé les itérations et une (Figure 4.6) qui indique le nombre d'itérations nécessaires à la convergence.

On voit clairement que les bassins d'attraction des racines sont totalement différents et que donc le cheminement des itérations vers la convergence est dissemblable (cela rejoint le fait que la suite des matrices B_i ne converge pas obligatoirement vers la jacobienne). De même, la répartition du nombre d'itérations nécessaires est différente, un point de départ délicat pour l'une des méthodes ne le sera pas forcément pour l'autre. Broyden présentent un nombre d'itérations plus élevés mais comme une itération est peu coûteuse, le temps de calcul de l'ensemble des problèmes est plus court que Newton : 0,3 s contre 4 s avec un pentium II 366 MHz. On peut remarquer cependant que seule la méthode de Broyden présente des points de départs pour lesquels la méthode ne converge pas.

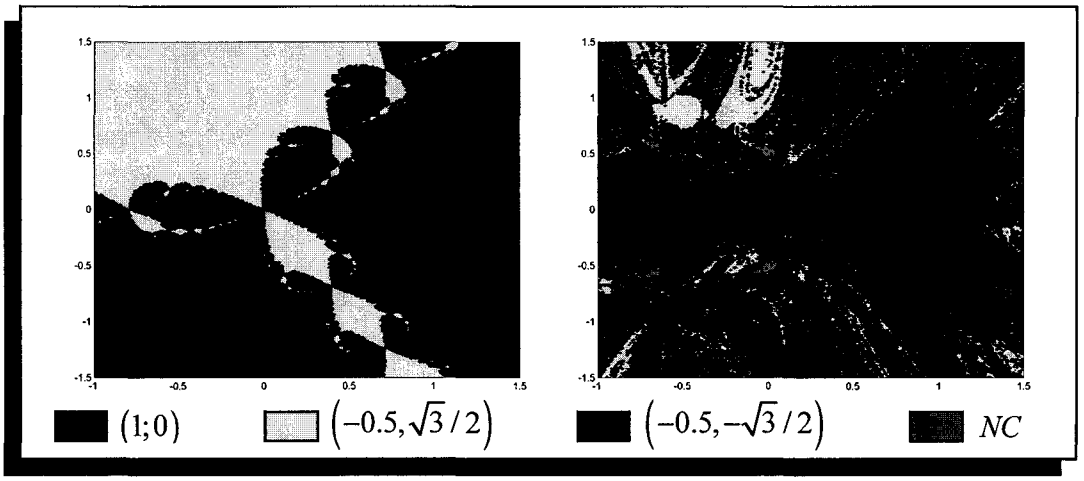


Figure 4.5 : Domaines d'attractions des racines de $z^3=1$. Newton à gauche, Broyden à droite

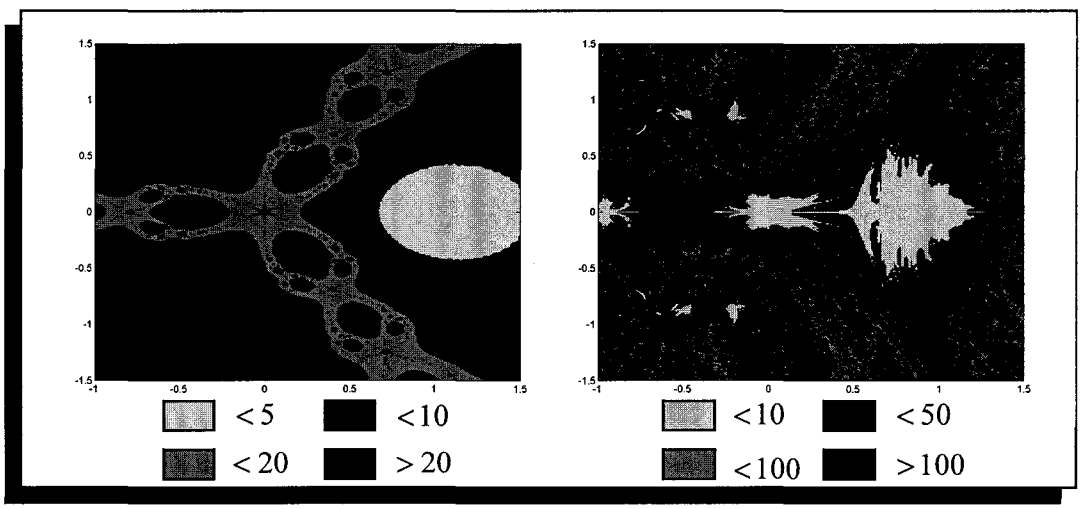


Figure 4.6 : Nombres d'itérations nécessaires pour résoudre le problème $z^3=1$. Newton à gauche, Broyden à droite.

4.3.2 Euler implicite

Dans un 2ème temps, nous avons repris le test des maillages de la partie 3.2.2 pour déterminer les limites de raideur utilisable ainsi que l'influence de la taille du système sur le temps de calcul. Nous utilisons toujours un pas de temps de 10 ms et une masse globale de 1 kg pour le maillage. Pour cela, nous allons utiliser la méthode d'Euler implicite ; dans la suite, lorsque nous ferons allusion à la méthode de Broyden, il est sous-entendu que c'est en association avec Euler implicite. Il est à noter que théoriquement il n'y a pas de limites supérieures pour les raideurs possibles mais l'introduction d'une méthode numérique de résolution de système non-linéaire amène des défauts de convergence.

Nous obtenons le même type de résultats qu'avec les méthodes explicites, i.e. plus le maillage devient fin, plus le rapport k/m doit être maintenu bas. La méthode de Broyden

permet d'augmenter la limite de ce rapport : avec la méthode de RK4, le rapport k/m maximal est de l'ordre de 10^3 à 10^4 ; maintenant ce rapport s'établit entre 10^4 et 10^5.

Pour étudier son coût, la Figure 4.7 représente le nombre d'itération pour chaque pas de temps pour 4 secondes de simulation pour un maillage 2D de 25 points avec une valeur de raideur proche de la limite. On voit qu'au début de la simulation (chute libre) et à la fin (position de repos sur le plan), la méthode de Broyden oscille entre 1 et 5 itérations, ce qui en moyenne représente un coup inférieur à RK4. On voit des élévations brutales du nombre d'itérations qui se produisent lors des contacts avec le plan. On peut noter que le premier pic est le plus haut parce que la vitesse du choc est la plus élevée. Il faut donc prévoir une augmentation du temps de calcul dès qu'un choc se produit.

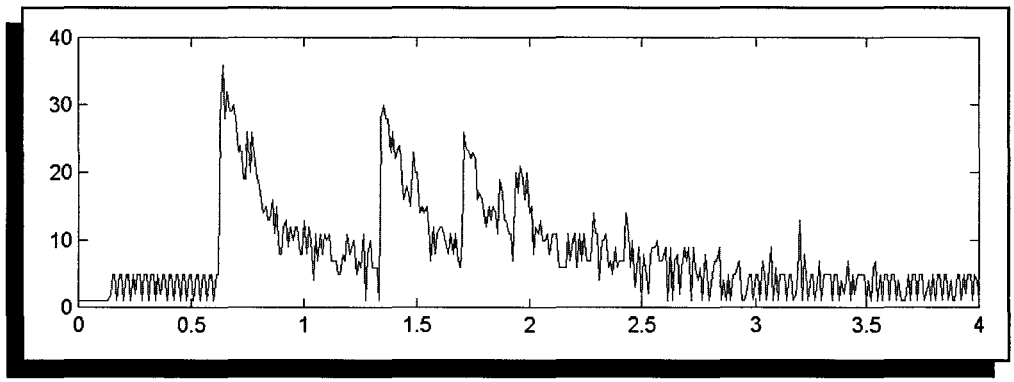


Figure 4.7 : Nombre d'itérations de Broyden en fonction du temps

Le nombre moyen d'itérations sur ces 4 secondes de simulation est 8.24, ce qui induit un peu plus de 8 évaluations de F en moyenne. Le système utilisé dans cet exemple comporte 150 variables et donc induit 150 évaluations de F pour générer une jacobienne. Cela représente un avantage certain par rapport à la méthode de Newton.

La Table 4.1 récapitule les temps (en moyenne et au maximum) de calculs par pas de temps (mesuré sur un pentium III 1 GHz) en fonction du nombre de points et pour une raideur dans la plage de valeurs possibles pour Broyden. Il est difficile de comparer avec la méthode de Newton car ce n'est pas une version optimale : nous avons fait une implémentation avec utilisation de la méthode LU et du bi-gradient conjugué en utilisant la bibliothèque MTL ([SL99]) pour les systèmes linéaires. Mais il reste que la formation de la jacobienne et la non prise en compte de structures spécifiques de la jacobienne comme le fait qu'elle peut être creuse rend le temps de calcul très long.

Nombre de Variables	k/m	Broyden	Newton LU	Newton Bi-CG
96	2,0e4	3,8 ms/10,2 ms	55,7 ms/109 ms	22,2 ms/72,6 ms
150	1,8e4	9,5 ms/20,6 ms	206 ms/406 ms	66,2 ms/198 ms
600	1,5e4	35,4 ms/154 ms	6 s/ 13 s	1,04 s/3,39 s

Table 4.1 : Temps de calcul pour la résolution d'un pas d'Euler implicite. Pour les méthodes itératives, on trouve le temps moyen et le temps max.

On remarque tout de suite que Broyden est très rapide par rapport aux méthodes de Newton. Il reste que les raideurs doivent rester dans une plage relativement restreinte et que le choc avec le plan provoque très facilement la divergence. On peut dire que Broyden converge moins bien que Newton mais lorsqu'il converge, il le fait plus rapidement. Le pas de temps étant fixé à 10 ms, Broyden est le seul capable de garantir un temps moyen inférieur.

Adaptation du Pas de Temps

Nous avons vu dans la section précédente que les limites de raideur maximale ont été repoussées par rapport à une méthode explicite mais un peu moins loin que les implantations de Newton de [BW98] et[EEH00]. Il serait profitable de pouvoir simuler une plus grande plage de valeurs. C'est dans ce but que nous utilisons une adaptation du pas de temps.

En général, la variabilité du pas de temps est utilisée pour diminuer le temps de calcul. En effet, si pour deux pas de temps différents, le temps de calcul est identique, il semble plus intéressant de prendre le plus grand : le temps simulé étant plus long, on calcule plus vite la solution. C'est de cette façon que cette adaptation temporelle est utilisée dans [BW98] par exemple. Il faut rappeler que le but des auteurs est l'animation non interactive qui prend beaucoup de temps d'exécution et donc une amélioration dans ce sens est intéressante.

Par contre, notre but est l'interactivité et surtout la synchronicité : il faut que le temps simulé et le temps de l'opérateur humain soit le même. Même si le calcul d'un pas prend beaucoup moins de temps que la durée du pas simulé, nous devons suspendre l'exécution pour assurer l'adéquation entre les deux temps. Ce temps « gagné » ne peut pas être utilisé pour autre chose car aucun calcul ne peut être anticipé à cause de l'intervention extérieure. De plus, le pas de temps ne peut pas être trop allongé sinon on perd alors l'interactivité. Cela conduit naturellement à l'utilisation d'un pas de temps fixe dont on sait calculer suffisamment rapidement l'évolution.

Cependant, les méthodes numériques de résolution de systèmes non linéaires convergent selon certaines conditions (comme avoir une estimation initiale assez proche de la solution). On ne peut manipuler les caractéristiques d'une scène pour les remplir ou alors au prix d'une perte de réalisme. Une solution est de détecter la divergence pour alors diminuer le pas de temps.

La méthode de résolution de systèmes non linéaires donne une mesure de cette divergence par le nombre d'itérations nécessaires à son calcul. On peut dire que si celui-ci devient trop grand, c'est que les conditions ne sont pas propices à l'obtention d'une solution. On se fixe alors une limite maximale du nombre d'itérations ; si celle-ci est dépassée, on recommence le calcul avec un pas de temps inférieur. Il existe d'autres techniques d'adaptation de pas de temps, mais elles sont liées au problème à résoudre contrairement à l'utilisation du résidu.

Afin de conserver, la synchronisation avec la fréquence de simulation, notre choix est de ne délivrer une solution qu'après la durée du pas de temps initial écoulée. Si une divergence est détectée, nous décidons de diviser le pas de temps Δt par deux. Puis nous résolvons deux

pas d'intégration à $\Delta t / 2$. Si l'un de ces deux pas venait à diverger, il suffit d'effectuer la même division. De cette façon, nous gardons une simulation à la fréquence demandée.

L'ajout de cette technique permet à la méthode de Broyden de mieux se comporter, on peut alors simuler une plus grande plage de raideurs avec des temps acceptables. La Table 4.2 donne des résultats pour différentes tailles de maillages avec pour nombre d'itérations maximum 20. Un rapport k / m de 10^6 donne un objet quasi-rigide tandis que pour une valeur de 10^4, le corps se déforme sensiblement. De même, nous obtenons une simulation correcte de tissus dans les deux scènes décrites au 0, il n'y a plus de phénomènes vibratoires.

Nombre de Variables	k / m	Temps Moyen	Temps Maximum
96	10^5	4,3 ms	9,5 ms
96	10^6	17,2 ms	21,7 ms
150	10^5	8,9 ms	20,9 ms
150	10^6	34,5 ms	44,3 ms
600	10^4	29,7 ms	107 ms
600	10^5	98,8 ms	231 ms
1350	10^5	487 ms	1,15 s

Table 4.2 : Temps d'exécution d'Euler implicite avec adaptation du pas de temps

Cette adaptation du pas de temps nous garantit la stabilité mais il faut maintenant composer avec des ralentissements plus importants. En effet, nous aurons bien une synchronisation avec la fréquence de simulation demandée mais le temps de calcul peut excéder le temps simulé. Nous avons pris le parti de tolérer des ralentissements éventuels, étant donné l'apport de la méthode quant aux raideurs qu'il devient possible de simuler. Pour ne pas perdre la synchronisation avec le temps de l'opérateur, nous attendons la prochaine échéance pour entreprendre le pas suivant. Tant que ces ralentissements ne sont pas trop fréquents, ils gênent peu l'utilisateur.

Il est à noter qu'il est difficile d'établir des performances d'une méthode à cause du nombre de paramètres qui influe dans la simulation. Par exemple, nos simulations ont été faites avec amortissement ambiant nul et un amortissement des liaisons très faible (rapport amortissement sur masse inférieur à 10^{-3}). Il est clair qu'avec des valeurs d'amortissements plus importantes, on peut obtenir de meilleures performances. Il faut compter avec d'autres paramètres comme les raideurs associées aux pénalités de collisions par exemple.

En plus des paramètres physiques, il faut ajouter ceux qui gouvernent les méthodes de résolution. Dans le cas de Broyden, il y a le nombre d'itérations maximum que l'on s'autorise. Pour la Table 4.2, nous l'avons fixé à 20. Pour avoir une idée de son influence sur le temps de calcul, nous avons utilisé les paramètres physiques de la $5^{ème}$ ligne de la Table 4.2 (150 variables et rapport k / m égal à 10^5 et nous avons fait varié le nombre maximal d'itérations (Figure 4.8).

Il apparaît qu'il faut conserver assez bas le nombre maximal d'itérations pour obtenir les meilleures performances. Cela tend à augmenter le nombre de divisions du pas de temps mais

d'un autre côté permet une détection plus précoce des divergences. Nous tenons à préciser que cette valeur optimale est très dépendante des paramètres physiques de la scène et qu'il est difficile de donner une valeur générale.

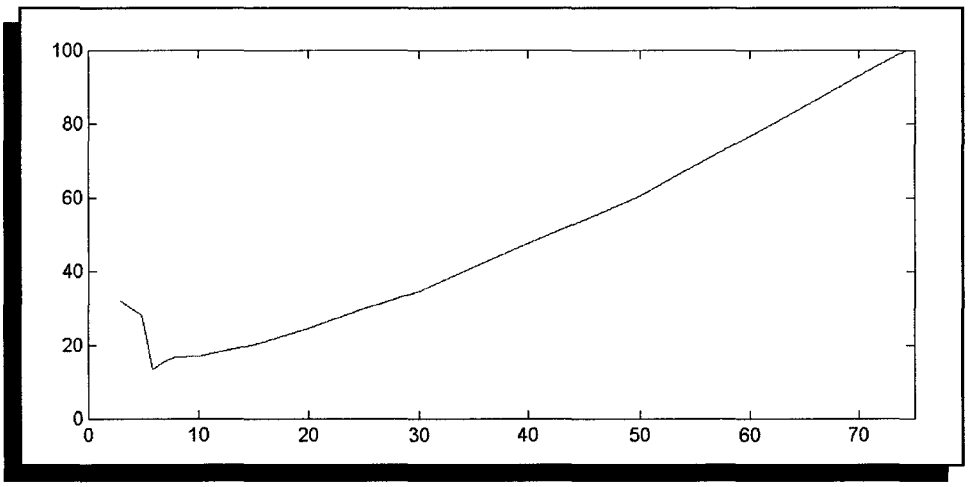


Figure 4.8 : temps moyen du calcul d'un pas de temps en fonction du nombre maximum d'itérations de Broyden

4.3.3 Comportement des méthodes d'intégration

Nous pouvons maintenant employer des méthodes implicites pour résoudre des EDO. Nos premiers tests ont été conduits avec la méthode d'Euler implicite mais bien d'autres méthodes avec des propriétés intéressantes (comme une meilleure précision) existent. Pour étudier les de comportements que peuvent apporter d'autres méthodes implicites, nous avons à nouveau effectué des tests avec le maillage masses/ressort.

Les tests sur Euler implicite ont montré le coût que représentait la résolution implicite. Nous avons donc décidé d'écarter toutes les méthodes de Runge-Kutta implicites. En effet, ce sont alors les vecteurs k_i qui sont déterminés par un système d'équations non linéaires. Une méthode RK à s étages comprend s vecteurs k_i à calculer. Donc, si le vecteur d'état comprend N variables alors le système à résoudre est de taille sN.

Il nous reste donc les méthodes multipas qui ne modifient pas la taille du système. Les méthodes que nous allons utiliser sont celles d'Adams-Moulton et les BDF. La méthode d'Adams-Bashforth d'ordre 2 est aussi nommée méthode du trapèze, nous utiliserons cette dernière appellation dans la suite. Nous allons coupler chacune de ces méthodes avec la méthode de Broyden et l'utilisation d'un pas de temps adaptatif.

Si on regarde les domaines de stabilité des différentes méthodes énumérées ci-dessus (voir §2.3.2), on remarque tout de suite qu'Euler implicite possède le domaine le plus vaste. Il semble donc que l'utilisation des autres méthodes peut apporter des problèmes. Nous avons donc fait des 1ers tests grossiers afin de constater des problèmes de convergence.

De fait, les méthode d'Adams-Bashforth d'ordre supérieur ou égal à trois sont inutilisables avec de grandes raideurs. Dans une moindre mesure, on remarque la même chose pour les méthodes BDF d'ordre supérieur ou égal à 3. Il nous faut donc abandonner l'idée d'utiliser des méthodes multipas d'ordre de précision élevée. Il nous reste donc la méthode du trapèze et la méthode BDF d'ordre 2.

Dans un premier temps, nous avons voulu connaître l'influence de l'emploi de ces méthodes sur le temps de calcul. La Table 4.3montre les temps moyen d'un pas d'intégration pour un temps simulé toujours de 10ms et un nombre maximum d'itérations de Broyden égal à 10. On voit clairement que la méthode d'Euler implicite est la plus rapide.

Nombre de Variables	k/m	Euler	Trapèze	BDF 2
96	10^5	3 ms	3,2 ms	6,3 ms
96	10^6	9 ms	9,2 ms	15,2 ms
150	10^5	7,2 ms	8,6 ms	13,1 ms
150	10^6	19,2 ms	22,3 ms	30,9 ms
600	10^4	22,4 ms	30,1 ms	56,5 ms
600	10^5	74,6 ms	88,8 ms	144 ms
1350	10^5	380 ms	530 ms	666 ms

Table 4.3 : Temps moyen d'un pas de calcul pour différentes méthodes

On peut voir deux raisons à ceci. D'abord, les méthodes ajoutent un peu de complexité à l'évaluation de F, qui consiste essentiellement en des additions de vecteurs. Mais surtout, le domaine de A-stabilité plus faible, contraint à opérer plus de divisions du pas d'intégration. La Table 4.4 montre la répartition des pas d'intégration selon le nombre de divisions nécessaires à la résolution. La colonne numérotée $i = 0,1,2,3$ indique le pourcentage des pas qui ont nécessité i divisions. La dernière colonne indique le surcoût par rapport à une résolution sans division. On remarque clairement que l'évolution de ce surcoût est conforme à l'augmentation du temps de calcul.

Méthodes	0	1	2	3	Résolution supplémentaire
Euler	30,3 %	59,6 %	4,5 %	5,6 %	170 %
Trapèze	6,8 %	88,3 %	3,7 %	1,2 %	198 %
BDF 2	3,5 %	85,1 %	11,1 %	3,8 %	237 %

Table 4.4 : Répartition des divisions du pas d'intégration selon la méthode employée

Enfin, nous avons voulu caractériser la conformité des solutions calculées par rapport aux paramètres physiques d'une scène. Nous avons utilisé une scène sans amortissement et nous avons observé l'énergie du maillage 2D, celle-ci devant rester constante. La Figure 4.9 montre cette évolution pour les 3 méthodes. La 1ère remarque est que la méthode d'Euler implicite fait diminuer l'énergie pour faire en sorte que le maillage atteigne une position de repos. La perte d'énergie se produit uniquement lors des contacts : l'effet d'amortissement induit par la méthode d'Euler se met en action lorsque les efforts mis en jeu deviennent importants. Il reste

qu'au lieu d'avoir un mouvement qui oscille perpétuellement, on obtient un objet qui atteint une position d'équilibre stable.

Par contre, la méthode du trapèze conserve quasiment parfaitement l'énergie du corps simulé et de fait, celui-ci rebondit et regagne sa hauteur de lâcher. Enfin, la méthode BDF d'ordre 2 provoque une perte d'énergie mais de bien moindre ampleur qu'Euler. Cette méthode peut donner l'illusion d'une préservation de l'équilibre énergétique à court terme mais on ne manquera pas de constater cette perte à moyen et long terme.

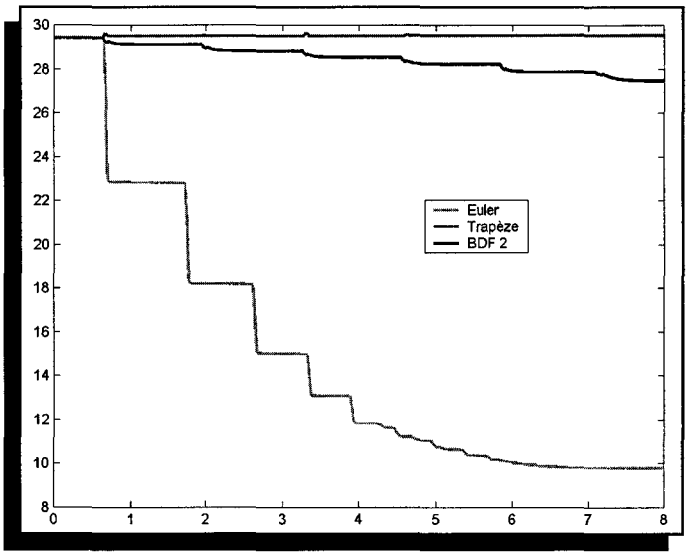


Figure 4.9 : Evolution de l'énergie d'un maillage masses/ressorts

Il semble donc que la méthode du trapèze peut constituer une méthode intéressante grâce à sa propriété de conservation. Il faut tout de même porter attention au fait que cette méthode amortit mal les oscillations transitoires : on constate dans certains cas que l'objet conserve une sorte d'instabilité autour d'une position d'équilibre. Ces oscillations sont d'autant plus perceptible que la raideur est faible.

4.4 BILAN

Nous avons présenté la méthode de Broyden pour la résolution de systèmes non linéaires afin d'utiliser des méthodes implicites de résolution d'EDO. Cette méthode présente l'avantage, par rapport aux méthodes classiques, de ne pas avoir à former la matrice jacobienne de la fonction dont on cherche un zéro. Cela donne une méthode rapide mais qui converge moins bien.

Grâce à l'utilisation d'une stratégie avec un pas de temps adaptatif, nous arrivons à faire en sorte que la méthode converge. Cela se fait au prix d'une augmentation du temps de calcul qui gène la synchronisation avec le temps réel. Il faut alors tolérer des ralentissements dans la simulation.

Enfin, nous avons mis en évidence le fait que la méthode d'Euler implicite est la méthode de résolution implicite d'EDO la plus rapide et la plus stable. Cependant, elle agit en

amortissant énormément les mouvements. Une méthode comme celle du trapèze permet d'obtenir une simulation sans cette déperdition d'énergie, au prix d'un plus grand temps de calcul et de moins de stabilité

Chapitre 5 : Le multi-systeme

Dans les systèmes mécaniques précédents, tous les degrés de libertés sont réunis au sein d'un vecteur d'état. Celui-ci peut devenir très imposant et la résolution demande plus de temps. Nous avons donc étudié un moyen d'accélérer la résolution d'un tel système composé de plusieurs corps. Il s'agit de séparer la résolution de chaque corps avec l'idée de résoudre localement les problèmes liés à l'intégration.

5.1 Description

5.1.1 Motivations

Dans la section 3.2.2, nous avons présenté une scène test constituée d'un morceau de tissu accroché par ses 4 coins dans lequel chutent des pavés. Nous avons donc regroupé dans la même scène, un modèle déformable et dix solides. Il est nécessaire de simuler le tissu avec une méthode implicite pour assurer la stabilité.

Le regroupement de ces degrés de libertés fait que la taille du vecteur d'état est assez grande pour que le calcul de la résolution implicite devienne trop coûteux. On peut noter deux faits :

- l'évolution des différents corps se produit de façon assez indépendante. Les interactions de collisions peuvent être pensées comme des forces extérieures.
- La division éventuelle de pas de temps demandée par un des corps va entraîner le re-calcul de l'évolution de certains corps pour qui cela se passe bien.

C'est ce qui conduit à l'idée de séparer les corps en sous-systèmes autonomes. En effet, chaque corps étant en mesure de construire sa trajectoire, la résolution des équations se fait de manière séparée réduisant la taille des systèmes à résoudre.

Sans découpage, la convergence est globale entraînant le calcul complet du vecteur d'état même si seule une partie des corps est concernée par des problèmes de divergence. Avec la séparation en sous-systèmes, chaque système est responsable de sa convergence et donc chaque système converge à sa vitesse.

Cette convergence locale fait que chaque système gère son adaptations de pas de temps. Ainsi, un corps qui nécessiterait une adaptation de pas de temps entraînerait un ralentissement de sa résolution uniquement. Il faut alors porter attention à la synchronisation sur la fréquence de simulation choisie : chaque système adapte son pas de temps pour faciliter son calcul mais tous donne leur résultat final pour le pas initialement demandé. De cette façon, tous les systèmes se trouvent dans un état temporel cohérent.

5.1.2 Interactions entre sous-systèmes

Pour opérer le découpage en plusieurs sous-systèmes, il faut pouvoir faire en sorte que les interactions entre corps qui étaient jusqu'à maintenant comprises dans le système soient transformées en forces extérieures pour chacun des corps. Les interactions entre corps peuvent être séparées en deux catégories : celles qui sont mises en œuvre par des forces appliquées à

chacun des systèmes et celles qui agissent par contraintes, i.e. en imposant des conditions qu'il faut respecter strictement.

Les premières ne posent pas de problèmes particuliers. Les forces d'interactions sont simplement des forces extérieures aux sous-systèmes, au même titre que la gravité par exemple. Le traitement des secondes est plus délicats. Il existe plusieurs façons de maintenir ces contraintes réalisées dans une résolution numérique :

- par pénalisation, i.e. par introduction d'efforts visant à ramener le système dans un état où la contrainte doit être réalisée.
- par les multiplicateurs de Lagrange (voir § 2.4.1.2 et § 2.4.2.4)
- par projection dans l'espace des valeurs possibles (voir § 2.4.2.3)

Concernant la pénalisation, c'est de cette façon qu'est conçu notre système de gestion des collisions. Celui-ci génère en effet des couples en collisions qui permettent le calcul d'une force qui va tendre à séparer les corps. La gestion de telles contraintes devient identique à la gestion de forces extérieures au système. Il reste à pouvoir les évaluer : il faut que chaque corps ait connaissance de l'état des autres corps. Cela oblige à maintenir une communication entre les objets autonomes. Dans notre cas, la gestion efficace des collisions nous oblige à maintenir une structure de données centralisée qui peut servir à obtenir ces informations.

L'utilisation des multiplicateurs de Lagrange conduit à un système comme (2.69). Dans [Rem00], l'auteur donne une méthode qui consiste à résoudre de manière découplée ce système. De cette façon, on résout le sous-système composé des contraintes pour en déduire des accélérations de corrections contre la violation des contraintes que l'on peut alors ajouter aux accélérations du mouvement hors contraintes. Dans un sens, cela revient à en déduire des pénalités comme précédemment mais les multiplicateurs permettent de modéliser des contraintes beaucoup plus complexes. Pour les utiliser, il faut effectuer une phase préliminaire de détermination des accélérations de correction, phase qui sera donc centralisée.

Il est à noter que les deux techniques précédentes tendent à réaliser les contraintes mais elles peuvent être momentanément ou durablement violées (surtout pour la première). L'utilisation d'une projection dans l'espace des contraintes permet d'assurer leur satisfaction. Il faut alors opérer une sorte de correction des solutions calculées. A nouveau, c'est un traitement qui doit être centralisé. La difficulté de cette technique est de trouver une manière générique de calculer cette projection.

En résumé, on peut considérer que les interactions peuvent être classées en termes de contraintes souples ou strictes. Les premières peuvent être parfaitement intégrées dans une séparation des systèmes ; le traitement des secondes est plus complexe mais reste possible par projection.

Il reste à préciser la manière dont ces interactions vont être évaluées. En effet, elles nécessitent de connaître la position de corps des autres systèmes. Or, on ne dispose pas des informations internes à la résolution des autres corps. Il faut donc se fixer une position de référence qui va rester constante pendant le calcul. Le fait de maintenir des corps fixes pendant

la résolution existe déjà dans SPORE : c'est de cette façon que les corps physiques prennent en compte les corps actifs.

Ainsi, lorsque le calcul de l'évolution d'un corps débute, l'état des autres objets est considéré comme fixe. Suivant le moment où le calcul débute, des systèmes ont peut-être déjà été résolus et d'autres attendent leur tour. Si l'on procède à une mise à jour de l'état global dès qu'un système a atteint le pas suivant, l'ordre dans lequel les systèmes sont calculés influe sur l'état global final.

Il nous paraît important que le placement des corps les uns par rapport aux autres doit être indépendant d'un ordonnancement arbitraire des calculs. Pour éviter cela, nous choisissons de baser les résolutions de chaque système sur l'état de début de pas d'intégration. Ce principe apparaît nécessaire dans le cas où l'on envisage une parallélisation, où l'autonomie doit être maximale. Le fait de considérer comme fixe les autres corps n'est pas sans influence sur l'évolution du corps considéré ; nous y reviendrons dans le paragraphe 5.2

5.1.3 Autres Apports

La séparation en sous-systèmes peut être mise à profit pour d'autres voies que celles qui nous ont motivées initialement. D'abord, nous pouvons profiter de cette autonomie de résolution pour adapter la méthode utilisée suivant le système : il s'agit de la multi-intégration. Dans l'idée initiale, une méthode de résolution d'EDO était choisie et chacun des corps l'utilisait de son côté. Nous avons imaginé utiliser autant de méthode de résolution d'EDO qu'il y a de systèmes.

Nous pouvons alors employer la méthode la plus appropriée aux besoins des corps d'un système dans le but d'obtenir un gain en performance et/ou un gain en comportement. Nous avons vu dans la partie 3.2.2 que notre modélisation d'un flux sanguin pouvait se contenter d'une intégration explicite ; par contre, un tissu a besoin d'une méthode implicite pour assurer une bonne stabilité. Dans une scène qui allierait ces deux corps, il est intéressant de pouvoir résoudre les corps séparément et chacun avec leur méthode. On garde ainsi la stabilité pour le tissu et on économise une résolution implicite pour le sang.

Nous pouvons également aller vers plus d'autonomie en effectuant une résolution répartie. En effet, le découpage en sous-système utilisant chacun sa propre méthode de résolution rend le calcul totalement cloisonné. Dès lors, il n'y a qu'un pas à faire pour imaginer l'exécution en parallèle de l'évolution de chacun des corps. Cette mise en parallèle peut être entreprise comme du multi-threading sur une même machine ou comme une répartition sur un ensemble de machines. Dans le dernier cas, il faut tout de même veiller au fait que la gestion des interactions entre systèmes impose une certaine dose de centralisation. Ce concept de simulation répartie est un des aspects du projet ALCOVE ([Alc02]) nouvellement démarré dans l'équipe GRAPHIX.

5.1.4 Mise en œuvre

Pour permettre l'utilisation de la séparation en multi-systèmes, il faut apporter des modifications aux structures décrites dans la section 3.1.1. En premier lieu, il faut pouvoir isoler les variables qui décrivent chaque sous-système. Pour ce faire, nous avons opéré un découpage au niveau du vecteur d'état. Le vecteur d'état devient une table indexée par un numéro représentant un système ; un élément de cette table est semblable à ce qu'était un vecteur d'état auparavant, i.e. deux membres X et V, contenant l'état respectivement en position et en vitesse du sous-système.

Il faut aussi apporter les modifications nécessaires dans les définitions des classes virtuelles décrites dans le §3.1.2. Pour la définition d'un corps physique, il faut juste ajouter un membre qui indique dans quel système se trouvent les variables d'état correspondant à ce corps. Quant à la définition de la classe *integrateur*, il faut lui adjoindre une méthode similaire à *PasInteg* avec un paramètre supplémentaire permettant d'identifier le système sur lequel il faut effectuer la résolution. Enfin, la classe *SystemeMecanique* doit aussi dédoubler ses routines *Calcul** pour faire les calculs sur un unique système. De plus, la méthode *AjouteCorpsPhysique* doit maintenant prendre en paramètre dans quel système se trouve le corps à prendre en compte.

5.2 Résultats

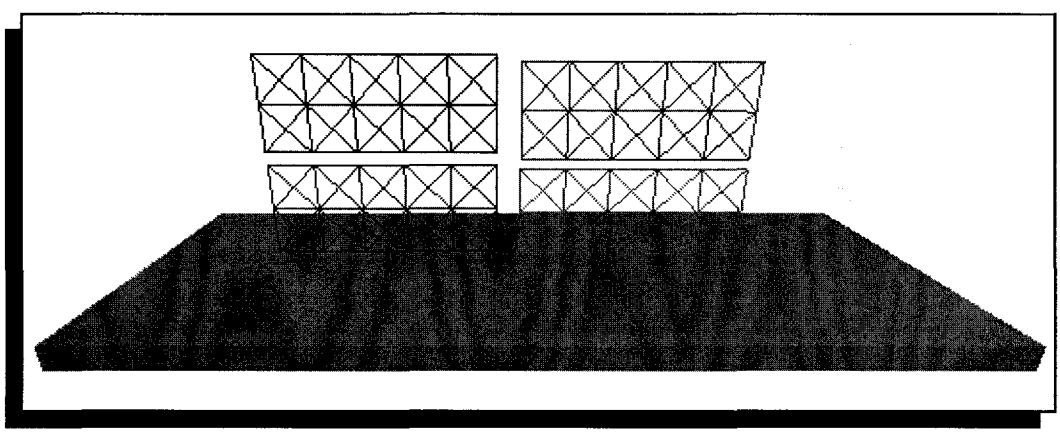


Figure 5.1 : Scène test

Nous avons réalisé une première implantation de ce découpage en systèmes indépendants qui fonctionne sur nos applications de tests avec les maillages. Son insertion dans SPORE est actuellement en cours.

Dans un premier temps, nous avons voulu contrôler la différence de comportement avec une résolution mono-système. Pour cela, il faut une scène test qui comporte des objets en interaction. Nous avons donc réalisé le test de la Figure 5.1. Les deux maillages de gauches sont simulés comme un système monobloc, tandis que les deux maillages de droite sont simulés selon le principe énoncé précédemment. Pour la réponse à la collision, nous utilisons le principe de pénalités.

Nous avons constaté que les deux simulations se comportaient différemment mais aboutissaient tous deux à l'équilibre final (Figure 5.2). Il est à noter que le comportement généré par la résolution multi-système reste plausible.

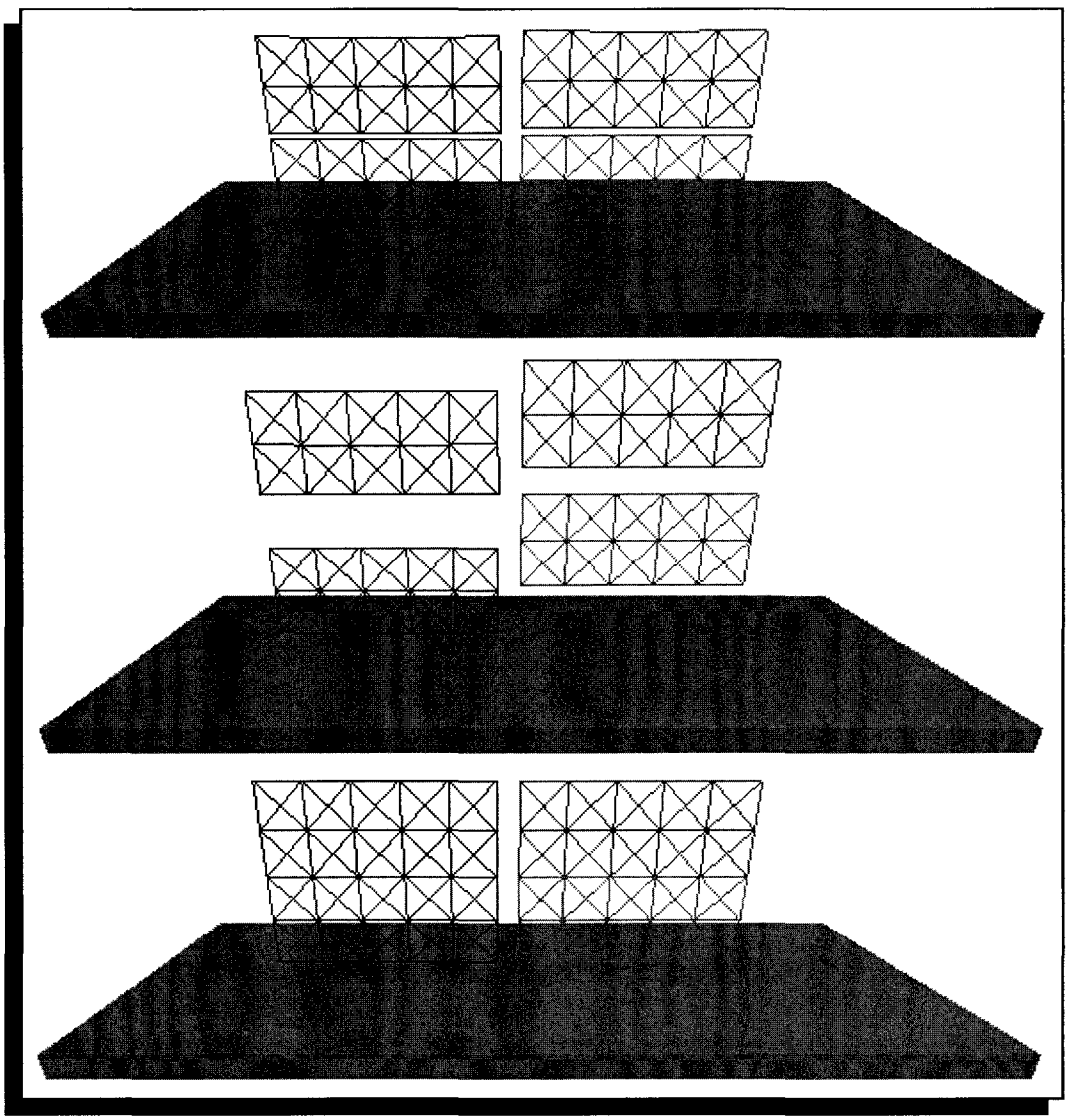


Figure 5.2 : Etapes des simulations mono- et multi-système.

Dans l'exemple précédent, l'interaction entre les deux maillages est de type répulsif. Il faut maintenant voir ce qui se passe avec des interactions de liaison. Nous avons commencé par deux maillages qui possèdent chacun un points fixe et qui sont reliés par un ressort (Figure 5.3). Là aussi, la simulation multi-système est différente mais elle reste plausible. Pour illustrer ceci, la Figure 5.4 montre l'évolution de la longueur du ressort reliant les deux corps. Pour réaliser cette figure, nous avons utilisé une raideur faible pour ce ressort.

On constate que la résolution multi-système amène plus rapidement vers la position finale : elle ajoute une sorte d'amortissement. On remarque aussi que la fréquence des oscillations est également différente : la raideur du ressort obtenu n'est pas celui spécifié. Ces modifications sur les paramètres physiques de la scène peuvent être gênantes quand on veut obtenir une simulation précise. Dans cet exemple, il n'y avait qu'une liaison, il faut maintenant étudier

le comportement de la résolution multi-système avec des corps reliés par un grand nombre de ressort.

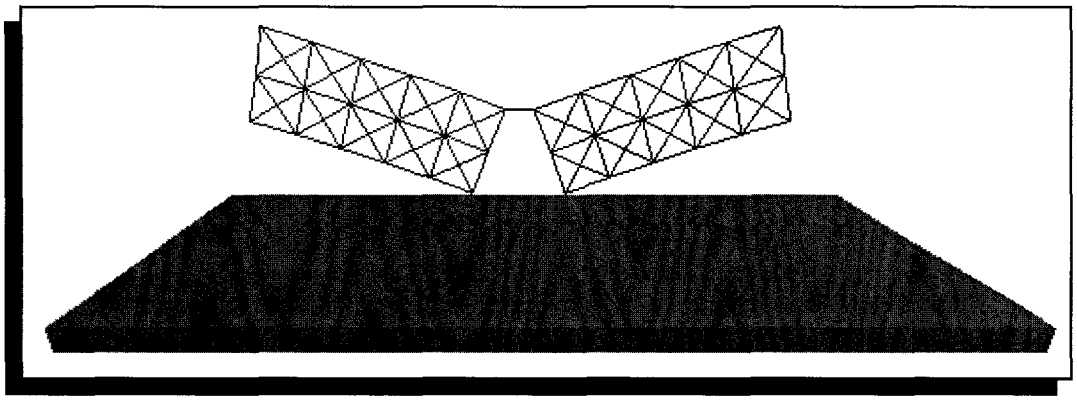


Figure 5.3 : Test de liaison entre corps.

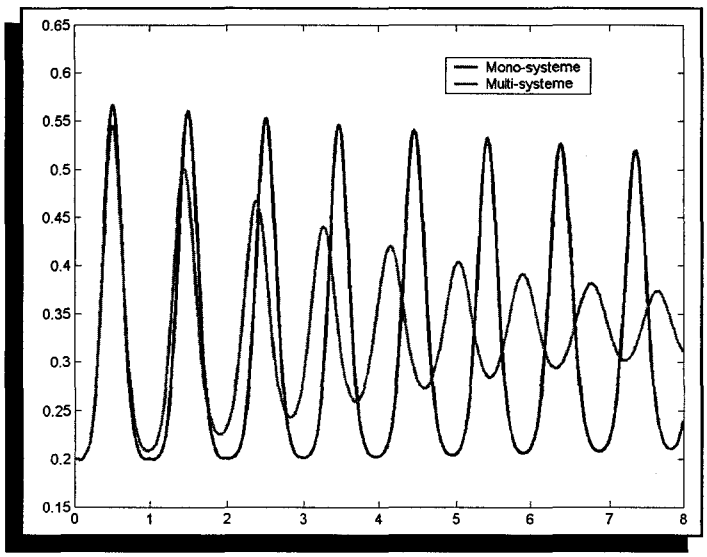


Figure 5.4 : Evolution de la longueur du ressort de liaison.

A cette fin, nous avons décomposé un maillage en deux parties avec des ressorts de liaisons qui permettent de reconstituer le maillage « normal » (Figure 5.5). Le résultat obtenu est cette fois-ci très éloigné de la solution mono-système. L'assemblage se déplace très lentement : le maillage « normal » a le temps de stabiliser après quelques rebonds tandis que l'assemblage s'est à peine déplacé.

En effet, la résolution locale d'un maillage considère comme fixe la position de l'autre maillage. Les maillages sont ainsi contraints de rester proche. Les forces d'interactions luttent alors contre la gravité. On voit donc que la résolution en multi-système n'est pas adapté à certains cas. Il semble qu'une liaison provoque une limitation du déplacement d'ensemble. Celui-ci devient perceptible lorsque deux corps sont fortements liés.

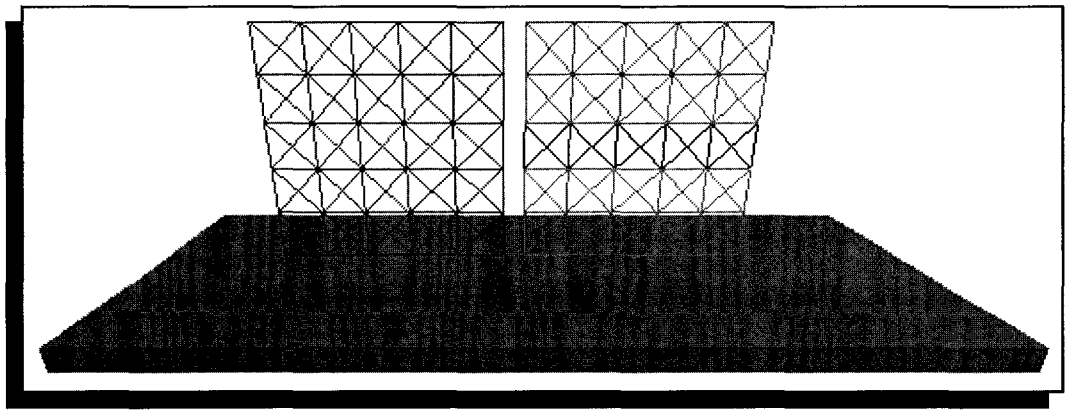


Figure 5.5 : Le maillage de droite en fait composé de deux maillages distincts reliés par les ressorts de la tranche centrale.

Enfin, nous avons voulu vérifier le comportement de la technique multi-système avec le positionnement de contraintes entre les systèmes. Nous avons choisi de satisfaire celles-ci par projection. Nous avons effectué des tests avec 3 maillages qui possèdent des points mis en commun (Figure 5.6). Nous obtenons une simulation similaire à celle résolue en mono-système : les corps ne possédant pas d'interactions dans la phase de résolution, les positions

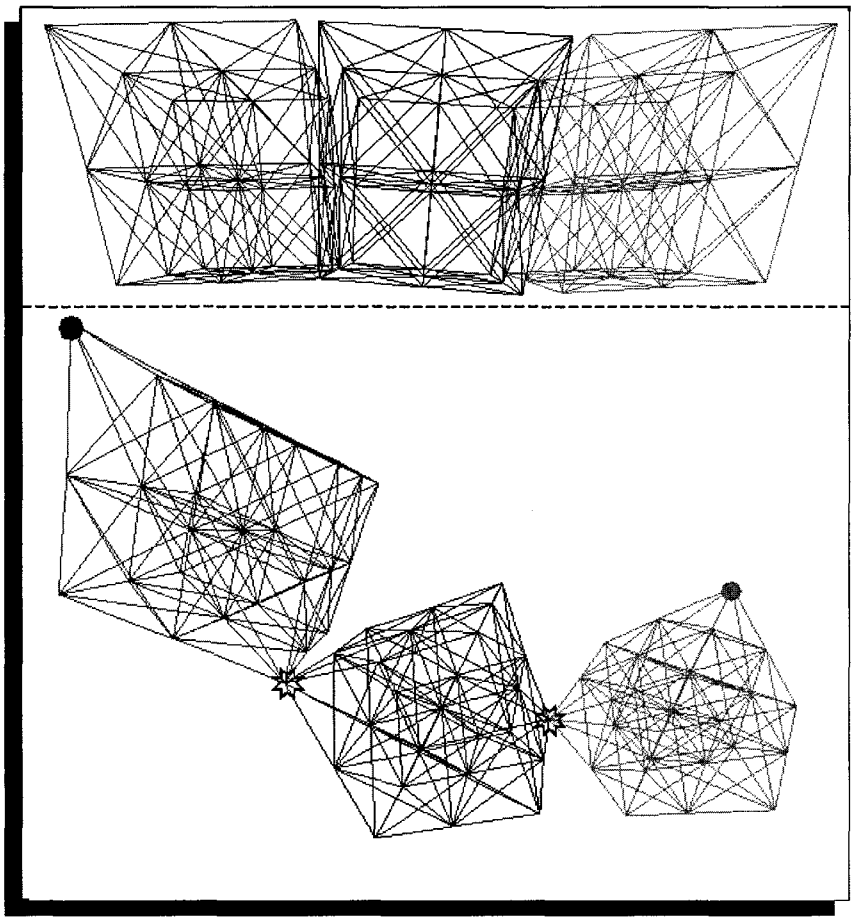


Figure 5.6 : Scène avec contraintes. Les disques représentent des points fixes, les étoiles les liaisons entre corps.

calculées sont donc les mêmes. Ensuite la phase de projection est identique dans les deux types de résolution. C'est donc une solution utilisable pour réaliser des contraintes.

Un des buts de la résolution multi-système est une accélération de la simulation. Nous avons utilisé une scène comportant deux maillages côte à côte. Dans un premier temps, nous avons fait varier la taille du maillage et le rapport k/m, tout en gardant les deux maillages identiques. Nous nous sommes aperçus qu'il n'y a pas de différences notables quant aux performances. Cela s'explique par le fait que la complexité d'une itération de Broyden est de l'ordre de $O(nN+Comp_F)$ (voir § 4.2.2) avec n, le numéro d'itération, et N, la taille du système. En découpant, en deux systèmes, la complexité associé à chaque système est de l'ordre de $O(nN_i+Comp_{F_i})$. La complexité d'une itération de la résolution complète devient alors $O(nN_1+Comp_{F_1}+nN_2+Comp_{F_2})$ soit $O(nN+Comp_{F_1}+Comp_{F_2})$. On voit donc que la complexité des opérations propres à une itération de Broyden est identique à un système unique. De plus, le terme F_i est constitué essentiellement du calcul des forces s'appliquant au sous-système i. L'évaluation de F n'est que le résultat de l'évaluation des forces s'appliquant à tous les systèmes, i.e. le cumul des F_i. Ainsi, les complexités sont là aussi identiques.

Il est tout de même possible d'obtenir un gain de temps en influant sur un autre paramètre de la résolution : la division du pas de temps. En effet, si l'on considère deux maillages identiques mais avec des raideurs différentes, ils ne nécessiteront pas le même nombre de divisions de pas de temps. Ainsi, la résolution multi-système donne la liberté à chacun des systèmes d'adapter le pas de temps au lieu de s'aligner sur la division la plus fine. La Table 5.1 montre les résultats obtenus avec différents maillages et raideurs.

Nombre de variables	k_1/m	k_2/m	Mono-système	Multi-système	Gain
2*45	10^5	10^6	8 ms	5 ms	37,5 %
2*96	10^5	10^6	18,9 ms	12 ms	36,5 %
2*150	10^5	10^6	37,5 ms	23,7 ms	36,8 %
2*600	10^4	10^5	160 ms	96,7 ms	39,5 %
45+150	9.10^4	$2,5.10^5$	14,2 ms	12,9 ms	9,1 %
150+600	10^5	4.10^5	100 ms	82,2 ms	11,8 %

Table 5.1 : Comparaison entre résolution mono- et multi-systèmes

On voit qu'un gain appréciable est possible. Il est tout de même très dépendant des paramètres physiques, qui influent sur le nombre de divisions du pas de temps. Dans l'optique du gain en performance, nous avons mis en place le fait d'utiliser autant de méthodes d'intégration que de systèmes. Le but recherché est d'utiliser des méthodes implicites pour les systèmes raides et une méthode explicite pour les autres. Il est clair que celles-ci ne sont utilisables qu'avec des cas particuliers comme notre simulation de flux sanguin ou qu'en utilisant des modèles restant dans les limites de stabilité (chapitre 3).

Nous avons commencé par des maillages qui restent dans les limites de stabilité (voir 3.2.2). Nous avons alors constaté qu'il n'y avait pas de différence notable entre les deux techni-

ques. Il faut noter que l'utilisation de la méthode de Broyden ne requiert qu'une seule évaluation des forces par itération et que son utilisation avec un problème non-raide engendre un coût similaire à une méthode explicite.

5.3 BILAN

Nous avons exposé une technique permettant de rendre autonome la résolution d'un corps au cœur d'un système complet. Ce découpage en sous-système a pour objectif une amélioration des performances. Cette autonomie permet aussi d'envisager une résolution répartie. Cependant, il faut tenir compte des interactions entre systèmes qui demande une centralisation de certaines données, dont la taille dépend de la (des) méthode(s) choisie(s) pour gérer ces interactions.

Nous avons donc mis en place le principe du multi-système et nous avons effectués quelques expérimentations. Nous avons ainsi constaté que cette résolution donnait une simulation différente de la résolution mono-système. La simulation obtenue reste acceptable parce que plausible sauf dans le cas d'un grand nombre de liaisons. Pour la gestion de contraintes, on peut utiliser la technique de la projection. Enfin, il est possible d'obtenir un gain en temps de calcul, celui-ci étant dépendant des caractéristiques physiques de la scène.

Notre étude sur le multi-système n'est pas terminée. Il faut maintenant étudier des cas où il y a un grand nombre d'interactions complexes, ce que nous ferons dès la mise en place de cette technique dans SPORE. De plus, le multi-système sera mis en œuvre sous forme répartie dans ALCOVE, ce qui pourrait être la source d'une nouvelle accélération. Dans ce cadre, il faudrait étudier les différentes manières de prise en compte des interactions afin de déterminer laquelle est la plus appropriée.

Conclusion

Nos travaux ont pour objectif la mise en œuvre de simulations basée sur la physique. Il s'agit de pouvoir calculer l'évolution d'objets en se basant sur des équations tirées des études de mécanique. Les équations mises en jeu sont des équations différentielles, qui peuvent être ramenées à des Equations Différentielles Ordinaires relativement au temps.

Ce type de simulation présente un grand intérêt dans la conception de simulateurs pédagogiques médicaux dans laquelle s'est investie l'équipe GRAPHIX. De tels simulateurs exigent que la résolution soit au mieux temps réel et au pire, d'un niveau d'interactivité suffisante pour ne pas perturber l'utilisateur et surtout de permettre l'utilisation de périphériques haptiques.

Dans ce but, nous avons entamé l'élaboration d'une plateforme générique (SPORE), capable d'effectuer la simulation de scènes comportant des corps mécaniques variés. Elle se doit donc de comprendre une méthode de résolution rapide des EDO. Celle-ci doit aussi présenter des qualités de robustesse numérique.

Les méthodes de résolution d'EDO sont classées en deux grandes familles : les explicites qui permettent un calcul direct de la nouvelle position et les implicites qui demandent la résolution d'un système non-linéaire. A l'intérieur de ces familles, on peut distinguer des propriétés importantes comme la précision par rapport à la vraie solution et notamment la stabilité (aptitude à ne pas diverger).

Les 1ères semblaient intéressantes pour pouvoir assurer le temps réel du fait de leur faible coût en calcul. Elles se sont révélées être utilisables dans des cas très précis du fait de leur très faible stabilité. Cette constatation conduit naturellement à vouloir employer les méthodes implicites. Dès lors, il faut pouvoir résoudre un système d'équations non-linéaires en gardant un coût modique.

Les méthodes classiquement utilisées demande la formation d'une matrice jacobienne, coûteuse dans sa phase de construction, et la résolution de système linéaire basée sur cette matrice. Afin de contourner ces problèmes, nous employons la méthode de Broyden dans une version qui évite tout stockage de matrice et ne forme pas de jacobienne. Cela nous permet d'obtenir une méthode de résolution générale, assez rapide pour assurer le temps réel dans des scènes simples et l'interactivité dans des scènes plus complexes.

Le temps de résolution dépend fortement du comportement de la méthode de résolution d'EDO utilisée. Il apparaît que la méthode d'Euler implicite reste la plus stable et la plus rapide. Elle présente toutefois la particularité d'être un peu trop convergente et d'amortir un

peu trop les mouvements. Pour une meilleure adéquation avec les paramètres physiques, on peut utiliser d'autres méthodes mais au prix de perte de stabilité.

Afin d'assurer que les méthodes fonctionneront à tout moment, nous avons mis en place une stratégie d'adaptation du pas d'intégration grâce à la mesure du résidu du système non-linéaire. Le niveau de divisions nécessaire est dépendant de la stabilité de la méthode. Cette division est interne à la résolution : de l'extérieur, les résultats obtenus sont toujours synchronisés (dans le temps simulé) avec une fréquence donnée afin d'assurer la cohérence avec l'acquisition de positions des périphériques et du rendu haptique.

Enfin, nous présentons l'idée de rendre autonome la résolution des corps regroupés en sous-systèmes dans le but d'un gain en performance. Nous avons effectué une première implantation qui donne des résultats encourageants. Il faut maintenant poursuivre cette idée pour mettre en place une résolution distribuée. Celle-ci permettra d'obtenir un gain de temps sur certaines architectures et surtout peut servir à la réalisation d'objectifs du projet ALCOVE, entamé depuis peu au sein de l'équipe GRAPHIX.

Nous avons exploité la méthode de Broyden dans le cadre d'une animation dynamique. Il serait aussi intéressant d'étudier son comportement avec une méthode de résolution statique, notamment en terme de performance. De plus, il est aussi possible de réaliser une version dédiée à la résolution de système linéaire qui pourrait concurrencer d'autres méthode itératives. Cette méthode bien que relativement ancienne attend encore de se révéler dans le domaine de l'animation.

Bibliographie

[A094] Alishenas T. and Olafsson 0. Modeling and velocity stabilization of constrained mechanical systems. BIT, volume 34, pages 455-483, 1994.

[AD92] B. Arnaldi and G. Dumont. Vehicle simulation versus vehicle animation. Proceedings of the 3rd Eurographics workshop on Animation and Simulation, 1992

[Alc02] Rapport INRIA

[ARW95] Asher U.M., Ruuth S.J. and Wetton B.T. Implicit-explicit methods for time-dependant partial differential equations. SIAM J. Numer. Anal., 32(3), pages 797-823, 1995

[Aub97] A. Aubry. Méthodes de Runge-Kutta pour les équations différentielles algébriques d'indice deux et les systèmes hamiltoniens. Thèse de Doctorat, Université de Rennes I,1997.

[Aul01] D. D'Aulignac. Modélisation de l'interaction avec objets déformables en temps-réel pour des simulateurs médicaux. Thèse de doctorat de l'INPG, décembre 2001

[Bar95] D. Baraff. Interactive simulation of rigid bodies. IEEE Computer Graphics and Application, 15, 3, pages 63-75, Mai 1995.

[Bat82] K.J. Bath. Finite element procedures, Prentice-Hall 1992

[BC00] D. Bourguignon and M.P. Cani. Controlling anisotropy in mass-spring systems. Proceeding of CAS 2000 Eurographic workshop, pages 113-123, 21-22 août 2000, Interlaken, Suisse.

[BC96] M. Bro-Nielsen and S. Cotin. Real-time volumetric deformable models for surgery simulation using finite elements and condensation. Proceedings of the EUROGRAPHICS'96 conference, pages 57-66, 28-30 août 1996, Poitiers.

[BDL] Bureau des longitudes. http://www.bdl.fr/ephemeride.html

[Bre73] R. Brent. Some efficient algorithms for solving systems of nonlinear equations, SIAM, J. Numer. Anal., 10 (1973), pages 327-344.

[But64] J.C. Butcher. On Runge-Kutta processes of high order. J. Austral. Math. Soc., Vol. IV, Part 2, pages 179-194

[But65] J.C. Butcher. On the attainable order of Runge-Kutta methods. Math. Of Comp., Vol19, pages 408-417.

[But85] J.C. Butcher. The non-existence of ten stage eigth order explicit Runge-Kutta methods. BIT, Vol. 25, pages 521-540.

[BW98] D. Baraff and A. Witkin. Large Steps in cloth simulation. Proceedings of ACM SIGGRAPH'98, pages 43-54, 1998.

[CM89] M. Crouzeix et A.L. Mignot. Analyse numérique des équations différentielles. 2ème édition, 1989

[Dav00] J. Davanne. Modèle de tissu biologique pour une simulation temps réel avec retour d'efforts. Mémoire de DEA d'informatique de l'université de Lille I. Juillet 2000

[DDBC99] G. Debunne, M. Desbrun, A. Barr and M.P. Cani. Interactive multiresolution animation of deformable models. Proceedings of Eurographics Workshop on Computer Animation and Simulation 1999.

[DG96] M. Desbrun and M.P. Gascuel. Smoothed particles : a new paradigm for animating highly deformable bodies. Proceedings of the CAS'96 Eurographics workshop, pages 61-76.

[DMC02] J. Davanne, P. Meseure and C. Chaillou. Stable Haptic Interaction In A Dynamic Virtual Environment. Proceedings of IROS conference, pages -, Lausanne, Octobre 2002.

[DZ93] P. Dworkin et D. Zelter. A new model for efficient dynamic simulation. Eurographics workshop on animation and simulation, pages 135-147, 4-5 septembre 1993

[EEH00] B. Eberhardt, O. Etzmuß and M. Hauth. Implicit-explicit schemes for fast animation with particles systems. Proceedings of Eurographics Workshop on Computer Animation and Simulation 2000, pages 137-151.

[GTT89] J.P. Gourret, N. Magnenat Thalmann and D. Thalmann. Simulation of object and human skin deformation in a grasping task. SIGGRAPH'89 Conference Proceedings, pages 21-30, 31 juillet-4 août 1989, Boston.

[Hai78] E. Hairer. A Runge-Kutta method of order 10. J. Inst. Maths Apllics,Vol. 21, pages 47-59, 1978.

[Hai99] E. Hairer. Numerical Geometric Integration. Cours de l'université de Genève , Mars 1999. Disponible sur http://www.unige.ch/math/folks/hairer/polycop.html.

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

[HMC00] L. Hilde, P. Meseure et C. Chaillou. Méthodes de résolution d'équations dynamiques du mouvement, Actes des 13èmes journées AFIG , Grenoble, France, 29 Novembre-1er Décembre 2000, pages 191-200.

[HMC01] L. Hilde, P. Meseure et C. Chaillou. A fast implicit integration method for solving dynamic equations of movement. Proceedings of VRST'01, Banff, Canada, 15th-17th november 2001.

[HW00] E. Hairer and G. Wanner. Solving ordinary differential equations, Volume I. Springer Verlag, 2nd edition, 2000.

[HW96] E. Hairer and G. Wanner. Solving ordinary differential equations, Volume II. Springer Verlag, 2nd edition, 1996.

[Jou95] A. Joukhadar. *Robot*Φ :Dynamic modeling system for robotics applications. Rapport de recherche INRIA 2543, mai 95.

[JQDC00] A.C. Jambon, D. Querleu, P. Dubois, C. Chaillou, P. Meseure, S. Karpf, C. Géron. SPIC: Pedagogical Simulator for Gynaecologic Laparoscopy. Proceedings of the 8th Medicine Meets Virtual Reality Conference, Newport Beach, 27th-30th January 2000, pages 139-145.

[Kel95] C.T. Kelley. Iterative methods for linear and non-linear equations. Philadelphia: society for industrial and applied mathematics, 1995

[Laz95] Lazarus. Courbes, cylindre et métamorphose pour l'image de synthèse. Thèse de doctorat de l'université de Paris VII, décembre 1995.

[LC99] D. Lamy and C. Chaillou. Design, implementation and evaluation of an haptic interface for surgical gestures training. Proceedings of the 7th GTRV congress, Laval, France, June 1999, pages 107-116

[Len01] J. Lenoir. Simulation mécanique temps réel de fils chirurgicaux. Mémoire de DEA de l'université de Lille I, juillet 2001.

[LJFC91] Luciani, A., S. Jimenez, J.-L. Florens, C. Cadoz and O. Raoult, Computational Physics : a Modeler-Simulator for Animated Physical Objects. Eurographics'91 conference proceedings, Vienne (Autriche), 1991.

[LW70] W. Liniger and R.A. Willoughby. Efficient integration methods for stiff systems of ordinary differential equations. SIAM J. Numer. Anal. Vol. 7, pages 47-66

[MC00] P. Meseure and C. Chaillou. A deformable body model for surgical simulation. Journal of visualisation and computer animation, 11, 4, September 2000, pages 197-208.

[Mes02] P. Meseure. Animation basée sur la physique pour les environnements interactifs temps-réel. Habilitation à diriger des recherches, 17 décembre 2002.

[Mes97] P. Meseure. Modélisation de corps déformables pour la simulation d'actes chirurgicaux. Thèse de doctorat en informatique de l'université de Lille I, janvier 1997.

[MKCR95] P. Meseure, S. Karpf, C. Chaillou, P. Dubois et J.F. Rouland. Low cost medical simulation : a retinal laser photocoagulation simulator. Proceedings of Graphics Interface 95, Québec city, Québec, 17-19 may 1995, pages 155-162.

[ML00] C.A. Mendoza et C. Laugier. A solution for the difference rate sampling between haptic devices and deformable virtual objects. International symposium in robotics and automation. Monterrey, Mexico, November 2000.

[Noc01] O. Nocent. Animation dynamique de corps déformables continus – application à la simulation de textile tricot. Thèse de doctorat de l'université de Reims, 20 décembre 2001

[Nou99] J.M. Nourrit. Modélisation, animation et visualisation de textiles à base de mailles. Thèse de doctorat de l'université de Reims, 5 janvier 1999.

[PB88] J.C. Platt and A.H. Barr. Constraint methods for flexible models. SIGGRAPH'88 conference proceedings, pages 279-288, 1-5 août 1998, Atlanta.

[PDR97] F. Peugnet, P. Dubois and J.F. Rouland. Clinical assessment of a training simulator for retinal photocoagulation. Proceedings of CVRMed-MRCAS'97, pages 409-412, 19-22 Mars, 1997

[PFTV93] W.Press, B. Flannery, S. Teukolsky, W. Vetterling. Numerical Recipes in C : the art of scientific computing. Cambridge University Press. http://www.nr.com

[PL99] G. Picinbono and J.C. Lombardo. Extrapolation : a solution for force feedback ? Proceedings of the 7th GTRV congress, Laval, France, June 1999, pages 117-125

[PLDA00] G. Picinbono, J.C. Lombardo, H. Delingette and N. Ayache. Anisotropic elasticity and force extrapolation to improve realism of surgery simulation. Proceedings of the ICRA 2000 conference.

[Pre01] S. Prévost. Modélisation implicite et visualisation multi-échelle par squelette à union de boules et graphe de recouvrement. Thèse de doctorat de l'université de Reims, 11 novembre 2001

[Pro95] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. Proceedings of the Graphics Interface'95 Conference, Québec City, Québec, 17-19 mai 1995, pages 147-154

[Rem00] Y. Rémion. Animation dynamique : moteur lagrangien généraliste et applications. Habilitation à diriger des recherches de l'université de Reims, 2000

[Rou00] F. Rousselle. Amélioration des méthodes de résolution utilisées en radiosité. Thèse de doctorat de l'université du littoral, 14 décembre 2000.

[SACN99] D. Stora, P.O. Agliati, M.P. Cani, F. Neyret, J.D. Gascuel. Animated Lava Flows. Proceedings of Graphics Interface, 1999, pages 203-210.

[SDB99] P. Shröder, M. Desbrun and A. Barr. Interactive animation of structured deformable objects. Proceedings of Graphics Interface'99.

[SL99] J.G. Siek, A. Lumsdaine. The Matrix Template Library : generic components for high-performance scientific computing. Computing in science and engineering, pages 70-78, 1999. http://www.lsc.nd.edu/research/mtl.

[TMC01] F. Triquet, P. Meseure and C. Chaillou. Fast polygonisation of implicit surfaces. WSCG'01, pages 283-290, février 2001, Plzen, République Tchèque.

[TO99] G. Turk and J.F. O'Brien. Shape Transformation Using Variational Implicit Functions. Proceedings of SIGGRAPH'99, pages 335-342

[Tou98] D. Tournès. L'origine des méthodes multipas pour l'intégration numérique des équations différentielles ordinaires. Revue d'histoire des mathématiques, 1998, pages 5-72.

[TPBF87] D. Terzopoulos, J. Platt, A. Barr and K. Fleisher. Elastically deformable models. SIGGRAPH'87 conference proceedings, pages 205-214, 27-31 juillet 1987 Anahaim.

[Tri01] F. Triquet. Facettisation temps réel de surfaces implicites. Thèse de doctorat de l'université des sciences et technologie de Lille, décembre 2001.

[Tur96] G. Turner. Modélisation de corps mous par éléments finis en vue d'animation temps réel. DEA d'informatique, Université des sciences et technologies de Lille I, juillet 1996.

[VCDM97] E. Varlet, C. Chaillou, P. Dubois et V. Maunoury. A simulator for initial training in ultrasound endoscopy. World congress on medical physics and biomedical engineering, Nice, France, September 1997

[VCM95] P. Vollino, M. Courchesne and N. Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. SIGGRAPH'95 proceedings, pages 137-144.

[VM00] P. Volino, N. Magnenat-Thalmann, Implementing fast Cloth Simulation with Collision Response, Computer Graphics International 2000, pages 257-266, 2000.

[Wib00] L. Wibaux. Une méthode de synthèse d'images échographiques : application à des simulateurs médicaux d'entraînements. Thèse de doctorat de l'université de Lille I, juillet 2000.

[ZS95] C.B. Zilles, J.K. Salisbury. A constraint based God Object method for haptic display. Proceedings of the IEEE Conference on intelligent robots and systems, 1995, pages 146-151.