Accélération de la simulation des corps mous pour les applications en temps réel

par

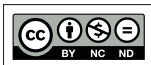
Nabil BOUKADIDA

MÉMOIRE PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
COMME EXIGENCE PARTIELLE À L'OBTENTION DE LA MAÎTRISE
AVEC MÉMOIRE EN GÉNIE DES TECHNOLOGIES DE
L'INFORMATION
M. Sc. A.

MONTRÉAL, LE 11 DÉCEMBRE 2020

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

Nabil Boukadida, 2020



Cette licence Creative Commons signifie qu'il est permis de diffuser, d'imprimer ou de sauvegarder sur un autre support une partie ou la totalité de cette oeuvre à condition de mentionner l'auteur, que ces utilisations soient faites à des fins non commerciales et que le contenu de l'oeuvre n'ait pas été modifié.

PRÉSENTATION DU JURY

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE

M. Eric Paquette, directeur de mémoire
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Sheldon Andrews, co-directeur
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. Michael John Mcguffin, président du jury
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

M. David Labbé, Examinateur Externe
Département de génie logiciel et des technologies de l'information à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 30 NOVEMBRE 2020

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

REMERCIEMENTS

Le travail mené dans cette maîtrise n'aurait pas été possible sans le support financier et logistique de Mitacs, à qui j'exprime ma gratitude pour m'avoir offert cette opportunité unique.

Je ne manquerai pas également de remercier l'entreprise de OSSimtech pour avoir donné un nouveau souffle aux études théoriques, à travers le projet de collaboration. Cette précieuse contribution m'a fait découvrir tout un monde de potentialités de la mise en application des concepts développés au cours de la maîtrise. Le bon déroulement de cette expérience était assuré par toute l'équipe de OSSimtech qui m'a marqué par son professionnalisme, sa dynamique et son incroyable sens de partage.

Toutefois, je dois une grande part de reconnaissance au directeur de recherche, Eric Paquette qui s'est investit pour explorer toutes les pistes afin de trouver les solutions optimales aux problématiques rencontrées. Cette approche a amélioré considérablement la qualité du travail en général et du mémoire en particulier grâce à un suivi et une présence continus, même au cours des circonstances les moins favorables. Je remercie également le codirecteur de recherche Sheldon Andrews dont l'expertise et les conseils m'ont fait gagner du temps en m'orientant vers les ressources pertinentes et les méthodes de travail efficaces.

Enfin, je ne pourrai me passer de remercier les membres du laboratoire de multimédia pour m'avoir inclus dans une ambiance aimable de travail et d'entraide et pour avoir été disposés à venir en aide par le partage de moyens et d'informations.

Accélération de la simulation des corps mous pour les applications en temps réel

Nabil BOUKADIDA

RÉSUMÉ

La simulation réaliste de la déformation des corps souples est une tâche complexe. Les méthodes basées sur la physique peuvent produire des résultats plausibles, mais elles peuvent être très coûteuses pour les applications en temps réel. Dans ce mémoire, nous proposons d'améliorer le taux de convergence de la dynamique basée sur les positions (XPBD) en utilisant l'accélération d'Anderson avec la méthode de sur-relaxation successive. Chaque itération du solveur XPBD est traitée comme une itération de point fixe, et l'accélération est appliquée sur les multiplicateurs de Lagrange produits par le solveur. De plus, des mesures de précautions sont installées pour assurer la stabilité de la simulation. L'approche utilise des contraintes basées sur la physique, qui permettent un contrôle plus fin sur les propriétés d'élasticité du matériau, ajoutant ainsi plus de réalisme à la simulation. Les résultats obtenus montrent que l'approche proposée apporte une amélioration significative du taux de convergence par rapport aux autres techniques d'accélération numérique.

Mots-clés: infographie, animation basée sur la physique, déformation de corps souple, XPBD, accélération numérique, accélération d'Anderson, méthode de surrelaxation successive

Acceleration of soft body simulation for real-time applications

Nabil BOUKADIDA

ABSTRACT

The simulation of realistic soft-body deformations is a challenging task. Physically based methods can produce plausible results, but they can be very expensive for real time applications. In this thesis we propose to improve the convergence rate of the extended position-based dynamics (XPBD) by using Anderson acceleration with successive over relaxation. Each iteration of the XPBD solver is treated as a fixed-point iteration, and the acceleration is applied on the Lagrange multipliers produced by the solver. In addition, we add multiple safeguards to ensure the stability of the simulation. The approach uses physics based constraints, which allow for finer control over the elasticity properties of the material, and thus adding more realism to the simulation. The obtained results show that the proposed approach provides a significant improvement of the convergence rate over the other numerical acceleration techniques.

Keywords: computer graphics, physically-based animation, soft-body deformation, XPBD, numerical acceleration, Anderson acceleration, successive over relaxation

TABLE DES MATIÈRES

Page

LISTE DES FIGURES

Page

LISTE DES ABRÉVIATIONS, SIGLES, TRADUCTIONS ET ACRONYMES

CPU	Processeur
Compliant constraint	Contrainte conforme
Constraint manifold	Variété de contrainte
Damping force	Force d'amortissement
FEM	Méthode des éléments finis - *Finite element method*
Flexibilité	Inverse de la raideur (*compliance*)
GPU	Processeur graphique
Geometric stiffness	Rigidité géometrique
PBD	Dynamique basée sur les positions - *Position based dynamics*
SOR	Méthode de surrelaxation successive - *Successive Over-Relaxation*
Strain	Vecteur de déformation
XPBD	Dynamique basée sur les positions améliorée - *Extended PBD*

LISTE DES SYMBOLES ET UNITÉS DE MESURE

$\cdot : \cdot$	$A : B$ définit la somme des produits $A_{i,j} \cdot B_{i,j}$
$\vec{a}_i$	Accélération de la particule i
α	L'inverse de la matrice de raideur
$C_i(x)$	Contrainte de l'élément i
Δt	Pas de temps
E	Module de Young
ε	Vecteur de déformation (*strain*)
ε_N	Vecteur de déformation pour un étirement
ε_S	Vecteur de déformation pour un cisaillement
$\vec{f}_i$	Force appliqué à la particule i
F	Gradient de déformation
h	Pas de temps
λ	Multiplicateur de Lagrange
λ	Premier coefficient de Lamé
K	Raideur
m_i	Masse de la particule i
maxIterations	Nombre maximum d'itération pour un solveur itérative
N	Nombre de particules
n	Cardinalité d'une contrainte
nc	Nombre de contraintes
∇u	Gradient d'une fonction u
p_i	Particule i
ψ	Densité d'énergie

R	Matrice de rotation
R_f	Matrice de réflexion
$tr(A)$	Trace de la matrice A
μ	Deuxièmme coefficient de Lamé
v	Coefficient de Poisson
$\vec{v}_i$	Vitesse de la particule i
V	Volume d'un tétraèdre
$\vec{x}$	Vecteur contenant toutes les positions des particules p_i
$\vec{x}_i$	Position de la particule i
$\vec{X}_i$	Position de la particule i au repos
$\|\|\vec{y}\|\|$	Norme d'un vecteur $\vec{y}$
W	Énergie de déformation
w_i	Inverse de la masse m_i

INTRODUCTION

Les simulateurs de déformation des corps mous occupent une place grandissante dans des applications diversifiées du domaine académique, professionnel et de divertissement. Par exemple, les formations médicales font de plus en plus appel à de nouveaux supports d'apprentissage proposant des solutions interactives alternatives aux solutions traditionnelles de formation chirurgicale. Dans ce contexte, les simulateurs de chirurgie ont la qualité de fournir un support flexible en termes pédagogiques, économiques et logistiques. Non seulement ils permettent de substituer les cadavres comme support d'entraînement, les simulateurs de chirurgie ont l'avantage d'imiter les conditions réelles d'intervention. On cite à titre d'exemple les manipulations sur les organes, ou bien l'écoulement des fluides tel que le saignement des tissus suite à leur dissection. Un simulateur réussi doit répondre à certains critères pour qu'il puisse offrir une expérience comparable à celle d'une opération chirurgicale réelle. Ceci nécessite un niveau de précision suffisant des interactions de l'utilisateur sur les objets de l'environnement qui sont essentiellement des corps souples. Ces derniers doivent réagir aux sollicitations en respectant les critères donnés par leurs configurations géométriques et leurs propriétés de matériaux. On entend par là que le solveur au cœur des traitements des données du simulateur doit être régi par des modèles physiques. Le solveur doit aussi être capable d'assurer un temps de réponse réduit pour donner un niveau d'interactivité plausible. II doit également garantir une fréquence de rafraîchissement assez élevée pour communiquer avec les dispositifs haptiques et assurer le réalisme du rendu des objets. Il faut donc faire des compromis sur l'implémentation des modèles physiques et adopter les méthodes d'approximation adéquates pour assurer un niveau d'interactivité convaincant sans trop affecter le comportement physique du système en question. Dans le cas spécifique d'un simulateur d'interventions sur les organes humains, il s'agit de modéliser la déformation due aux forces extérieures appliquées sur l'organe par l'utilisateur du simulateur. Les méthodes d'approximation géométriques des volumes déformables sont employées afin de réduire les temps de calcul.

Pour décortiquer ce problème, l'état de l'art des recherches est passé en revue dans l'objectif de sélectionner une méthode adéquate au scénario d'une simulation en temps réel. Certaines de ces méthodes dites conditionnellement stables ont des limitations liées à l'ampleur des déformations supportées. D'autres donnent lieu à un comportement physiquement incorrect, ce qui résulte en une modélisation non fidèle aux différents types de matériaux et leurs comportements de déformations. Certaines méthodes peuvent être gourmandes en termes de ressources de calcul et en conséquence, inadéquates pour le cas d'une application en temps réel. Par exemple, dans le cas du projet traité dans ce mémoire, l'utilisation seule de la dynamique basée sur les positions (XPBD) aboutit à une convergence très lente en raison de la nature itérative du solveur utilisé.

Plusieurs techniques ont été proposées pour améliorer la convergence dont la surrelaxation successive (SOR) dans XPBD (Macklin *et al.*, 2016) et l'accélération de Chebyshev proposé par Wang (2015). L'utilisation de la SOR peut parfois causer une surestimation de la solution ce qui mène souvent à un effet contreproductif où le taux de convergence devient encore plus faible, et dans certains cas donne une divergence dans la solution. La méthode de Chebyshev, peut offrir une convergence beaucoup plus rapide que celle obtenue avec SOR, en raison du facteur de relaxation qui change dynamiquement lors de l'approximation de la solution suivante. Toutefois le gain dans le taux de convergence demeure négligeable pour les modèles tétraédriques qui nous intéressent dans ce mémoire.

Le travail mené dans cette thèse contribue aux approches étudiées en s'appuyant sur une méthode itérative (XPBD) et améliorant le résultat de sa convergence en utilisant une nouvelle technique d'accélération numérique basée sur la méthode de Anderson. Différents tests ont été conduits afin d'analyser en premier temps la stabilité et l'efficacité de l'approche proposée. Ensuite, les résultats obtenus sont comparés avec d'autres méthodes de l'état de l'art. Bien que notre approche améliore le taux de convergence, la simulation devient instable dès que

nous appliquons des forces de déformation importantes sur l'objet. Ceci entraîne l'inversion des tétraèdres qui résident dans l'objet.

Dans ce mémoire, les chapitres sont organisés comme suit : Le chapitre 1 présente une revue de la littérature des méthodes utilisées en infographie pour simuler les objets déformables. Le chapitre 2 présente la méthode proposée dans le cadre de ce mémoire avec plus de détail. Ceci inclut les contraintes basées sur la position (XPBD) ainsi que l'approche de Anderson qui sert à accélérer la résolution des équations numériques. Le chapitre 3 fournit une comparaison avec d'autres méthodes d'accélération déjà établies avec XPBD. Le chapitre 4 explique les limitations rencontrées. Finalement, le chapitre 5 propose des travaux futurs qui visent à améliorer l'approche proposée.

CHAPITRE 1

ÉTAT DE L'ART

Ce chapitre donne un aperçu des méthodes liées à l'animation des corps souples et se concentre en particulier sur les systèmes à base de particules. En premier lieu, la structure du système de particules ainsi que les techniques qui sont utilisées pour faire avancer l'état du système dans le temps sont présentées d'une manière générale. La deuxième partie du chapitre passe en revue les méthodes d'accélération numérique qui ont été utilisées dans l'approche adoptée.

1.1 Système de particules

Les systèmes de particules sont souvent utilisés pour simuler une variété de phénomènes physiques complexes allant de la turbulence des fluides à la déformation des corps élastiques. L'implémentation de ces effets envisage une modélisation physique du système de particules détaillée et adaptée à la variété d'applications qui en tirent parti.

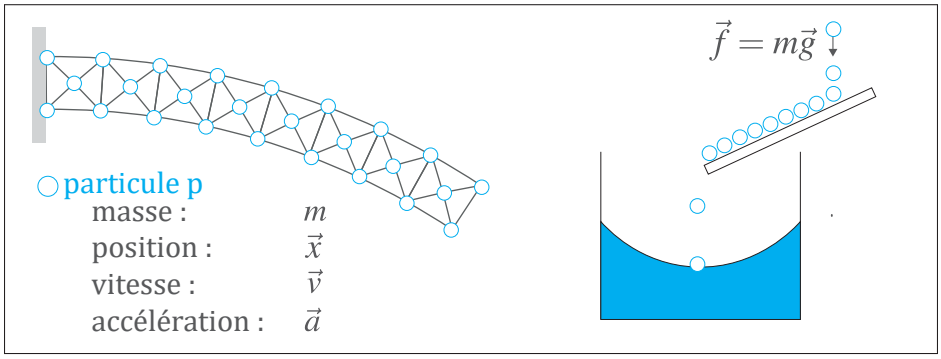


Figure 1.1 Simulation d'une poutre en porte-à-faux (gauche) et de fluides (droite) en utilisant les systèmes de particules

Le déplacement des particules dans ces systèmes se fait dans un espace 3D et s'appuie sur un ensemble d'attributs assigné à chaque particule. Le rafraîchissement des positions de ces particules se fait par une intégration sur la vitesse. L'accélération des particules est la résultante des forces qui leur sont appliquées au cours de la simulation.

Il existe plusieurs techniques de modélisation des systèmes de particules selon le scénario de la simulation. Par exemple, pour la simulation des corps déformables, un système de particules sous la forme de maillage de particules est mis en œuvre, avec les particules représentant un objet physique. Le comportement du système de particules dépend des propriétés physiques qui lui sont assignées telles que sa masse m, son accélération $\vec{a}$, sa vitesse $\vec{v}$ et sa position $\vec{x}$. Ceci est illustré dans la figure 1.1 où deux systèmes de particules sont utilisés pour simuler la déformation des corps mous et des fluides.

1.2 Intégration temporelle

La simulation est un processus complexe parce que son principe de base repose sur la prédiction de l'état futur d'une opération, qui, dans le cas traité ici, est la déformation des corps souples. La simulation de la déformation se décompose en cadres. L'intervalle de temps qui sépare deux cadres successifs correspond à la taille de pas de temps h. Les états correspondants aux cadres relatifs aux prochains pas de temps doivent être calculés au fur et à mesure de l'acquisition des positions $\vec{x}$ et des vitesses $\vec{v}$ des points constitutifs de l'objet déformé. La figure 1.2 illustre un exemple pour une animation avec des pas de temps discrets.

Un schéma d'intégration du temps qui repose sur des équations temporelles dynamiques sert à décrire l'état du système en question en fonction des termes de position $\vec{x}$, de vitesse $\vec{v}$ et de pas de temps h. Ce chapitre présente trois schémas d'intégrations d'Euler : un schéma d'intégration temporelle explicite, un schéma d'intégration de temps symplectique et un schéma d'intégration implicite.

1.2.1 Intégration explicite

Ce schéma est aussi connu sous le nom de *forward Euler* ou encore *explicit Euler*. L'idée derrière cette méthode se base sur l'utilisation d'une formule explicite pour approximer les intégrales du schéma d'intégration de temps continu. Il s'agit d'une méthode qui suppose que les valeurs de la position et de la vitesse ne changent pas beaucoup au cours d'un même pas de

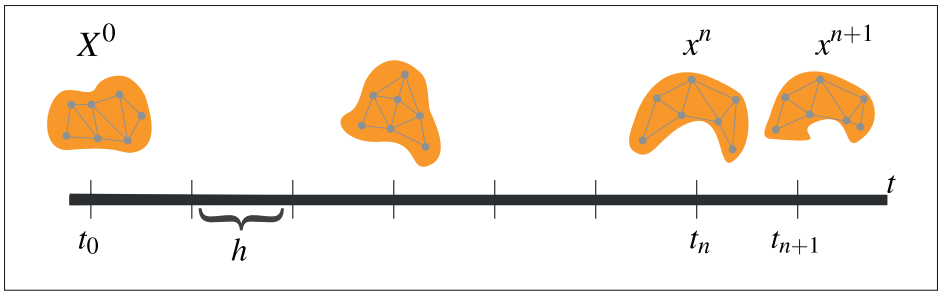


Figure 1.2 Avancement de la simulation dans le temps par des pas discrets

temps. L'information de la position et de la vitesse à l'état actuel est utilisée comme étant une approximation constante :

$$\vec{v}^{t+1} = \vec{v}^t + h\frac{\vec{f}}{m} \tag{1.1}$$

$$\vec{x}^{t+1} = \vec{x}^t + h\vec{v}^t \tag{1.2}$$

1.2.2 Intégration symplectique

L'intégration symplectique, aussi appelée *semi-explicit Euler*, *semi-implicit Euler* ou encore *symplectic Euler*, permet d'améliorer les résultats obtenus par le schéma d'intégration de temps explicite. Pour obtenir le schéma d'intégration de temps symplectique, il suffit de prendre $\vec{v}^{t+1}$ au lieu de $\vec{v}^t$ comme paramètre dans l'équation différentielle $\vec{x}^{t+1}$:

$$\vec{v}^{t+1} = \vec{v}^t + h\frac{\vec{f}}{m} \tag{1.3}$$

$$\vec{x}^{t+1} = \vec{x}^t + h\vec{v}^{t+1} \tag{1.4}$$

Le principe de cette méthode ne s'éloigne pas beaucoup de celui du schéma d'intégration de temps explicite du fait de l'exécution séquentielle de ses opérations. La stabilité de cette méthode est limitée et ne peut être maintenue que si le pas de temps est suffisamment petit.

1.2.3 Intégration implicite

La méthode d'intégration de temps implicite (Baraff & Witkin, 1998), aussi connue sous l'appellation de *backward Euler* ou *implicit Euler*, met en relation ses valeurs de sortie avec les termes de l'équation du prochain pas de temps résolvant ainsi les inconnues du système d'une manière implicite. Le calcul du prochain état est obtenu par la résolution du système ci-dessous :

$$\vec{\Delta}x = h(\vec{v}^t + \vec{\Delta}v) \tag{1.5}$$

$$\vec{\Delta}v = \frac{h}{m}\vec{f}(\vec{x}^t + \vec{\Delta}x, \vec{v}^t + \vec{\Delta}v) \tag{1.6}$$

En utilisant l'expansion de Taylor, il est possible de faire l'approximation de la force f comme

$$\vec{f}(\vec{x}^t + \vec{\Delta}x, \vec{v}^t + \vec{\Delta}v) = \vec{f}^t + \frac{\partial \vec{f}^t}{\partial \vec{x}}\vec{\Delta}x + \frac{\partial \vec{f}^t}{\partial \vec{v}}\vec{\Delta}v, \tag{1.7}$$

où la dérivée $\frac{\partial \vec{f}^t}{\partial \vec{x}}$ donne le Jacobien en termes de positions et $\frac{\partial \vec{f}^t}{\partial \vec{v}}$ est le Jacobien en termes de vitesses. La résolution de ce système revient donc à résoudre l'équation

$$\vec{\Delta}v = \frac{h}{m}(\vec{f}^t + \frac{\partial \vec{f}^t}{\partial \vec{x}}h(\vec{v}^t + \Delta\vec{v}) + \frac{\partial \vec{f}^t}{\partial \vec{v}}\vec{\Delta}v). \tag{1.8}$$

Le calcul des positions des points constitutifs de l'objet déformé est ensuite fait de la même manière que dans le schéma symplectique (équation 1.4)

Les principaux critères de comparaison entre les différents schémas d'intégration de temps sont liés à la stabilité de leur comportement, la taille des pas de temps nécessaires pour assurer cette stabilité et leur coût en termes de ressources de calcul. Dans cet égard, les méthodes d'intégration explicite et symplectique peuvent être décrites comme étant conditionnellement stables puisqu'elles requièrent l'utilisation de pas de temps réduits pour assurer la stabilité de

la simulation. La méthode symplectique a l'avantage d'être plus stable que la méthode explicite et elle conserve aussi l'énergie du système. Le calcul de ces deux méthodes a l'avantage d'être adapté aux ressources de calcul parallèle où les équations sont distribuées en équations indépendantes pour chaque particule et assignées chacune à un cœur du processeur parallèle.

Quant au schéma d'intégration de temps implicite, il est décrit comme étant inconditionnellement stable parce qu'il n'introduit pas d'énergie supplémentaire au système même si l'on utilise de larges pas de temps. Cependant, il peut causer une dissipation de l'énergie se traduisant en un amortissement non désirable à l'animation. En plus son exécution s'avère trop exigeante en termes de ressources de calcul, le rendant ainsi peu pratique dans le scénario d'une simulation en temps réel. Ceci est dû au fait que ce schéma repose sur un système d'équations non-linéaires qui ne peut être résolu qu'en le transformant en séquence d'équations linéaires. En plus, la résolution des équations linéaires obtenues fait appel à des méthodes de calcul complexes telles que la factorisation de Cholesky, où la résolution itérative du système d'équations algébriques. Tout ce processus résulte en un temps de traitement plus important. Dans l'approche proposée, l'intégration semi-implicite sera donc utilisée pour avancer l'état du système au prochain cadre, car celle-ci est à la fois rapide et stable et permet également de conserver l'énergie.

1.3 Simulation de corps élastiques

1.3.1 Systèmes masses-ressorts

Les systèmes masses-ressorts sont parmi les moyens les plus simples pour simuler des objets déformables. Étant facile à implémenter et non coûteux en termes de calcul, ce type de système est souvent utilisé lorsque le comportement visuel prime sur la précision numérique. Il est fréquemment utilisé pour simuler une variété d'objets tels qu'une corde, un tissu, des objets déformables, ou encore dans des scénarios de simulation chirurgicale (Zhang *et al.*, 2017).

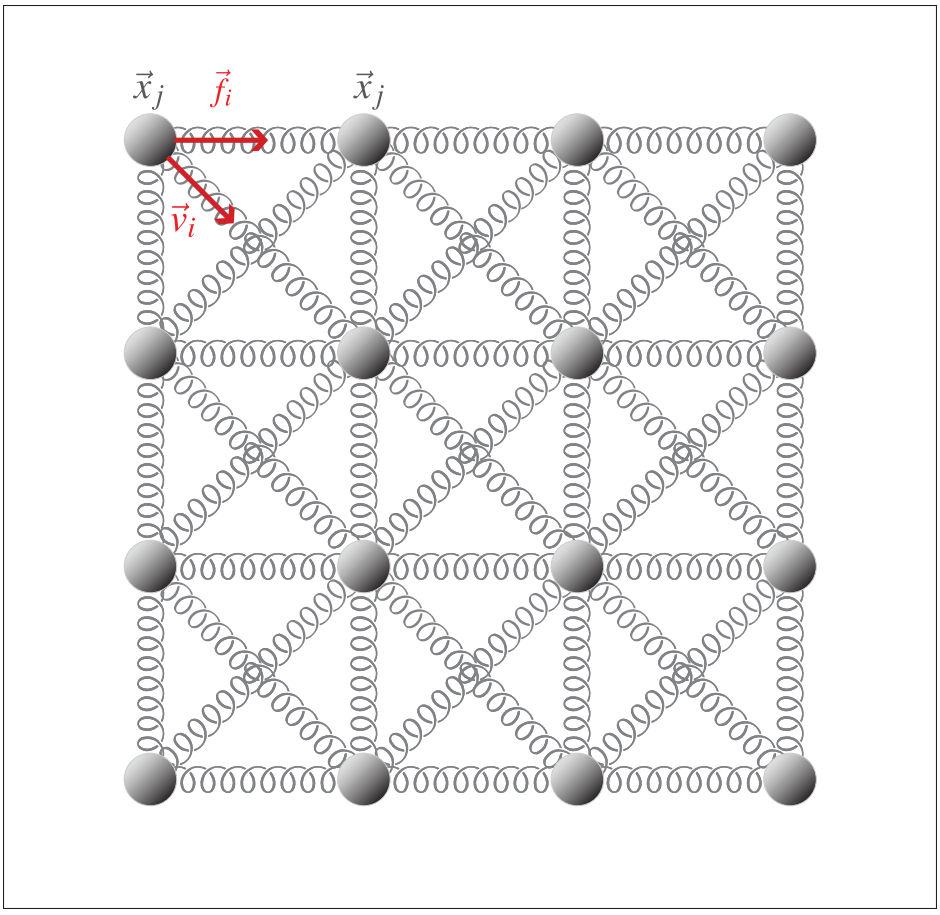


Figure 1.3 Système masses-ressorts

Dans le cas de modélisation d'une simulation masse-ressort, un objet 3D discret constitué d'un système de particules interconnectées entre elles par un ensemble de ressorts constitue la référence du modèle à développer. Chaque particule dans ce système possède une masse non nulle m et obéit à la deuxième loi de Newton. Les ressorts ont des masses nulles et possèdent une distance l_0 à l'état de repos. À titre d'exemple, la figure 1.3 visualise un tissu simulé avec un système de masses-ressorts.

L'objectif de cette méthode consiste à utiliser les ressorts comme un moyen pour retourner l'objet à son état de repos quand les particules se déplacent, causant ainsi des étirements et des compressions des ressorts. À titre d'exemple, la figure 1.4 illustre les forces produites aux extrémités d'un ressort lorsqu'il est soumis à un étirement. L'évaluation de l'interaction entre

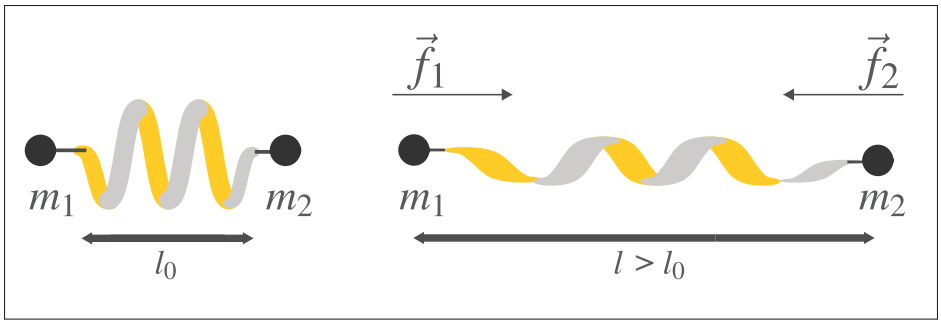


Figure 1.4 Étirement d'un ressort de longueur de repos l_0

un ensemble de ressorts revient à calculer les forces internes de l'objet déformable. Donc, pour calculer l'état du système dans les pas de temps futurs, il suffit d'appliquer une intégration de temps sur le terme de l'accélération $\vec{a}$ dans la deuxième loi de Newton

$$m\vec{a} = \vec{f} \tag{1.9}$$

où m est la masse d'une particule p, et $\vec{f}$ est la force de chaque ressort donnée par la loi de Hooke :

$$\vec{f}_1 = m\vec{a}_1 = k\Big(\frac{\|\vec{x}_2 - \vec{x}_1\|}{l_0} - 1\Big) \cdot \frac{\vec{x}_2 - \vec{x}_1}{\|\vec{x}_2 - \vec{x}_1\|} \tag{1.10}$$

$$\vec{f}_2 = -\vec{f}_1 \tag{1.11}$$

En faisant l'intégration de l'accélération dans les équations 1.10 et 1.11, on obtient en premier lieu les vitesses des particules et par la suite les nouvelles positions. L'implémentation de cette approche nécessite un choix soigné de la méthode d'intégration, comme discuté dans la section précédente. Dans les applications temps-réel, l'intégration Eulérienne implicite peut être très coûteuse en termes de calcul et peut causer un *damping* important à la simulation. D'autre part, l'intégration semi-implicite peut être instable dans le cas où le pas de temps h n'est pas bien choisi.

Malgré la simplicité et la popularité des systèmes masses-ressorts, la simulation de différents types de matériaux dans des milieux continus reste toujours difficile. En raison, cette approche

considère que l'objet est constitué d'un système de ressorts isolés où la masse est distribuée sur un ensemble de particules. En plus, cette approche ne repose que sur les propriétés physiques liées aux ressorts pour calculer les forces internes d'un objet déformable. Ces propriétés sont peu adaptées pour la simulation d'autres types de matériaux possédant des propriétés élastiques plus complexes. Afin d'imiter le comportement d'un matériau spécifique, il faut faire un réglage très précis des coefficients de raideur de chacun de ces ressorts. Ce type de réglage pourrait être très délicat, car il peut introduire des comportements anisotropiques non désirables à la simulation. D'autre part, le comportement de la simulation dépend largement de la topologie du réseau de ressorts. La force résultante appliquée à une particule p va être affectée par la direction et la longueur de repos des ressorts qui sont au voisinage, qui eux-mêmes dépendent des positions des particules qui sont au voisinage de la particule p. Une autre limitation de cette méthode est due au comportement de la déformation qui dépend souvent de la résolution du système masses ressort. Plus la résolution des ressorts est fine plus le comportement obtenu lors de la simulation devient souple ce qui complique davantage le contrôle sur l'élasticité des matériaux pour l'utilisateur. Dans notre approche, l'objectif était de simplifier à l'utilisateur le contrôle de l'élasticité de l'objet juste en réglant certaines propriétés physiques d'un matériau. Pour cela, d'autres méthodes de la littérature plus basées sur la physique ont été étudiées.

1.3.2 Méthode des éléments finis

La méthode des éléments finis (FEM) est parmi les méthodes les plus utilisées lorsque la précision physique est une préoccupation de première importance. Cette approche permet de simuler une variété de matériaux en se basant sur les lois de la mécanique des milieux continus. Cette méthode consiste à estimer l'énergie potentielle stockée dans l'objet lorsqu'il est soumis à une déformation et à calculer ensuite la force interne qui est égale au gradient de l'énergie. Cette force permet à l'objet de retourner à son état de repos. Pour ce faire il faut en premier lieu trouver une fonction qui permet de transformer la configuration de repos

à la configuration déformée, pour ensuite utiliser cette fonction afin d'estimer la quantité de déformation et calculer l'énergie potentielle stocké dans l'objet.

Il n'est pas possible d'évaluer les forces dans un milieu continu car ceci nécessite des ressources de calcul infinies. Ainsi, la méthode des éléments finis utilise une discrétisation spatiale du corps élastique en un ensemble d'éléments dits éléments finis. Souvent, des triangles sont utilisés dans l'espace 2D, et des tétraèdres ou bien des hexaèdres sont utilisés dans l'espace 3D pour couvrir le domaine à simuler.

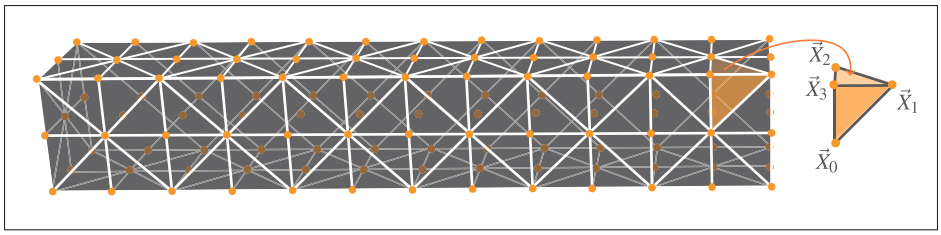


Figure 1.5 Maillage d'éléments finis 3D

Dans le maillage tétraédrique (figure 1.5) chaque élément (tétraèdre) possède quatre nœuds (positions) notés par les vecteurs $\vec{X}_0$, $\vec{X}_1$, $\vec{X}_2$, $\vec{X}_3$ à la pose de repos et par les vecteurs $\vec{x_0}$, $\vec{x_1}$, $\vec{x_2}$, $\vec{x_3}$ à la pose déformée. Afin de mesurer la déformation d'un objet, il faut commencer par définir une fonction bijective Φ entre la configuration de repos notée par $\vec{X}$ et la configuration déformée notée par $\vec{x}$:

$$\Phi\left(\vec{X}\right) = \vec{x}. \tag{1.12}$$

Dans l'approximation linéaire de FEM, la fonction de déformation Φ peut être exprimée par la fonction linéaire

$$\Phi\left(\vec{X}\right) = F\vec{X} + \vec{t}, \tag{1.13}$$

$$F = \frac{\partial \Phi\left(\vec{X}\right)}{\partial X}, \tag{1.14}$$

où F est le gradient de déformation, (une matrice de taille 3x3) et $\vec{t}$ représente le vecteur de translation entre la pose de repos et la pose déformée. Comme expliqué dans Sifakis & Barbic

(2012), pour trouver l'expression de F il suffit d'écrire les équations algébriques pour les quatre nœuds du tétraèdre :

$$\begin{aligned} F\,\vec{X}_0 + \vec{t} &= \vec{x}_0 \\ F\,\vec{X}_1 + \vec{t} &= \vec{x}_1 \\ F\,\vec{X}_2 + \vec{t} &= \vec{x}_2 \\ F\,\vec{X}_3 + \vec{t} &= \vec{x}_3 \end{aligned} \tag{1.15}$$

Ce système d'équations peut être réarrangé et décris dans la forme matricielle :

$$F\left[\vec{X}_0 - \vec{X}_3 \quad \vec{X}_1 - \vec{X}_3 \quad \vec{X}_2 - \vec{X}_3\right] = \left[\vec{x}_0 - \vec{x}_3 \quad \vec{x}_1 - \vec{x}_3 \quad \vec{x}_2 - \vec{x}_3\right] \tag{1.16}$$

$$F\,D_m = D_s \tag{1.17}$$

Etant donné que les positions sont connues à la configuration de repos ainsi que dans la configuration déformée, il demeure possible de calculer le gradient de déformation par le calcul

$$F = D_s\,{D_m}^{-1}. \tag{1.18}$$

Bien que le gradient de déformation fournisse de bonnes informations sur la déformation appliquée à l'objet, il ne représente pas une mesure correcte de la déformation. Ceci est dû à l'évaluation de la rotation comme étant une déformation, ce qui reste indésirable dans tous les cas. Pour remédier à ce problème, une matrice ε connue sous le nom de *strain* est appliquée au modèle. Cette fonction a la particularité de rester invariable quand une rotation est appliquée au modèle. La formulation la plus utilisée de strain est *Green strain* dont l'expression est

$$\varepsilon = \frac{1}{2}\,(F^{\intercal}F - I). \tag{1.19}$$

Avec cette formulation, il est possible de voir que lorsque l'objet élastique subit une rotation $R \in SO(3)$, dans le cas où $F = R$, le strain est égal à $\varepsilon = 0$ parce que $R^\intercal R = I$, et il en résulte que $R^\intercal R - I = 0$. Donc le Green strain peut représenter une bonne mesure de la déformation et il s'avère adapté pour la plupart des scénarios. Mais dans certains cas il peut produire des artéfacts à chaque fois où la déformation devient tellement importante que les éléments deviennent inversés. Dans ce cas, le gradient de déformation contient une matrice de réflexion R_f où $R_f^\intercal R_f$ sera aussi égal à l'identité, et par conséquent ε sera égale à 0 ce qui se traduit par des éléments qui demeurent inversés. Ce problème peut être remédié par l'utilisation du modèle d'élasticité linéaire de co-rotation. Plus de détails peuvent être trouvés dans la publication de Sifakis & Barbic (2012). Bien que ce modèle soit plus robuste pour les déformations extrêmes, le Green strain a été utilisé dans ce projet de recherche. La raison principale de ce choix se manifeste dans les coûts de calcul supplémentaires de la décomposition polaire occasionnés par l'utilisation de la déformation du modèle de corotation. De plus, la plupart des organes humains sont suffisamment rigides pour éviter l'inversion d'éléments internes, ce qui rend le choix de Green strain plus approprié.

Une fois que la mesure de déformation est obtenue, il devient possible d'estimer l'énergie potentielle stockée qui est due à la déformation en utilisant la loi de comportement du matériau (*material law*). Cette loi fournit un modèle permettant d'approximer la densité d'énergie ψ répartie dans le volume d'un tétraèdre lors d'une déformation. Plusieurs modèles ont été proposés, parmi eux on cite le modèle de déformation de Saint Venant–Kirchhoff (StVk) où le Green Strain est utilisé pour mesurer la quantité de déformation avec la densité d'énergie de déformation donnée par

$$\psi(\varepsilon) = \mu\varepsilon : \varepsilon + \frac{\lambda}{2} tr^2(\varepsilon), \tag{1.20}$$

$$\text{où } \varepsilon : \varepsilon = \sum_{i,j} \varepsilon_{i,j} \cdot \varepsilon_{i,j} \tag{1.21}$$

avec μ et λ les coefficients de Lamé donnés par les équations 1.22 et 1.23 :

$$\lambda = \frac{Ev}{(1+v)(1-2v)} \tag{1.22}$$

$$\mu = \frac{E}{2(1+v)} \tag{1.23}$$

Le coefficient de Poisson v représente le taux de contraction/dilatation dans les directions latérales lorsque l'objet soumet une déformation. Le module de Young E désigne la résistance du matériau contre les déformations telles que l'étirement.

Dans l'approche linéaire de FEM, la densité d'énergie est uniforme dans tout le volume V du tétraèdre, ceci implique que le calcul d'énergie totale W emmagasinée dans un tétraèdre est obtenu par

$$W \;=\; \psi(\varepsilon)\, V \;= \psi(\varepsilon)\,.\,\frac{1}{6}\left|det(D_m)\right|. \tag{1.24}$$

L'étape finale de FEM consiste à calculer les forces appliquées sur chaque nœud du tétraèdre, et de faire la sommation de toutes les forces appliquées sur tous les tétraèdres pour obtenir la force résultante. Pour calculer la force pour un seul nœud, il suffit de calculer la dérivée négative de l'énergie par rapport au nœud comme suit :

$$\vec{f_i} \;=\; -\frac{\partial W}{\partial X_i} \tag{1.25}$$

Une fois que les forces pour chaque nœud sont calculées, il devient possible d'estimer la pose de l'objet élastique dans le cadre suivant en effectuant une intégration pour calculer la vitesse et puis les positions.

1.3.3 Dynamique basée sur les positions

La dynamique basée sur la positions (PBD) est une méthode d'intégration temporelle introduite par Müller *et al.* (2007). L'idée principale de cette méthode consiste à simuler des forces internes en utilisant des contraintes plutôt que des forces. Chaque contrainte correspond à une fonction $C_i : \mathbb{R}^{3n} \rightarrow \mathbb{R}$ possèdant une cardinalité n. Cette cardinalité correspond au nombre de particules qui sont impliquées dans une contrainte. Il existe généralement deux types de contraintes, les contraintes bilatérales et les contraintes unilatérales qui correspondent respectivement aux contraintes d'égalité et d'inégalité. Dans ce mémoire nous utilisons seulement les contraintes d'égalité où les contraintes seront imposées lorsque l'égalité $C(x) = 0$ n'est pas respectée.

Algorithme 1.1 PBD

Fonction : PBD
Arguments : x^t
Sortie : x^{t+1}

Prédiction du vitesse $\vec{v}^{t+1} = \vec{v}^t + h\vec{a}^t$
Amortissement du vitesse
Prédiction de position $\vec{x}^{t+1} = \vec{x}^t + h\vec{v}^{t+1}$
for $i < maxIterations$ **do**
    for $C_j(\vec{x})$ $j \in (1, ..., nc)$ **do**
        Projection de la contrainte $C_j(\vec{x})$:
            Résoudre $\vec{\Delta}x$ pour la contrainte $C_j(\vec{x})$
            $\vec{x}^{t+1} = \vec{x}^{t+1} + \vec{\Delta}x$
    end
    $i \leftarrow i + 1$
end
Mise à jours des vitesses des particules $\vec{v}^{t+1} = \frac{1}{h}\left(\vec{x}^{t+1} - \vec{x}^t\right)$

L'algorithme PBD commence par une mise à jour des vitesses pour toutes les particules en faisant une intégration d'Euler explicite. Ensuite, il est possible d'appliquer un amortissement sur les vitesses afin de renforcer la stabilité de la simulation. Cette étape est optionnelle, mais elle permet d'assurer une meilleure stabilité à la méthode de PBD.

L'étape suivante utilise une intégration implicite sur les positions. Les lignes 4 à 11 de l'algorithme 1.1 consistent à résoudre les contraintes au niveau des positions des particules. Cette étape est souvent appelée projection de contraintes. Pendant la boucle de projection, les contraintes sont linéarisées les unes après les autres avant d'être résolues séquentiellement. L'expression linéaire pour chaque contrainte peut être obtenue par une expansion de Taylor donné par l'équation 1.26 :

$$C(x+\Delta x) \approx C(x) + \nabla_x C(x) \cdot \Delta x = 0 \tag{1.26}$$

Le but est de calculer Δx de telle sorte que la contrainte soit minimisée. Le déplacement Δx est déterminé par la résolution d'un système non linéaire d'équations où, dans une itération donnée, les positions impliquées dans une contrainte donnée sont projetées pour satisfaire uniquement la contrainte donnée. L'expression de ce déplacement est

$$\Delta x_i = -k w_i \frac{C(x_1,\ldots,x_n)}{\sum_j w_j \left|\nabla_{x_j} C(x_1,\ldots,x_n)\right|^2} \nabla_{x_i} C(x_1,\ldots,x_n) \ , \tag{1.27}$$

où $k \in [0,1]$ est le paramètre de rigidité qui reflète le degré de satisfaction voulu pour la contrainte.

Avec les positions sont projetées sur le *manifold* de contrainte, il reste important de corriger les vitesses des particules afin d'éviter un dépassement (*overshooting*) provenant de l'intégration explicite.

Lors de la projection des contraintes, il est important de conserver la quantité de mouvement (équation 1.28) ainsi que le moment cinétique (équation 1.29) pour les forces internes, car sinon des forces artificielles dites forces fantôme seront introduites au système. Ces forces fantômes

impactent la simulation en la rendant instable.

$$\sum_i m_i \vec{\Delta} x_i = 0 \tag{1.28}$$

$$\sum_i r_i \times m_i \vec{\Delta} x_i = 0 \tag{1.29}$$

Afin d'éviter ce genre de problème et conserver la quantité de mouvement et le moment cinétique pour ces forces internes, il est important que la fonction de contrainte soit indépendante des mouvements de la translation globale et de la rotation globale du système. En maintenant ces deux propriétés, le gradient de la fonction de contrainte deviendra également indépendant des déplacements de translations et de rotation globales et par conséquent il sera de même pour le déplacement de la particule engendré par l'équation 1.27.

La dynamique basée sur la position peut gérer de nombreux types de contraintes grâce à des fonctions de contraintes. Dans l'article original de Müller *et al.* (2007), les auteurs ont dérivé divers types de contraintes pour simuler différents phénomènes tels que la flexion, l'étirement, la compression, la conservation du volume, les collisions, etc. Plus de détails sur ces types de contraintes peuvent être trouvés dans l'article de Müller *et al.* (2007).

Dans ce projet de recherche, nous avons expérimenté avec différents types de contraintes afin de simuler des objets déformables. Les premiers résultats obtenus avec des contraintes de distance et de volume ont été encourageants mais ils ont montré un manque de contrôle précis sur les propriétés physiques d'un matériau donné. En outre, la rigidité dans le PBD classique dépend du nombre d'itérations ainsi que de la taille du pas de temps utilisé dans le solveur. Cela est principalement dû à l'effet de la multiplication du $\vec{\Delta} x$ par k à chaque itération qui est non linéaire. Donc le contrôle de la rigidité des contraintes sur plusieurs itérations d'une manière cohérente n'est pas évident.

Depuis l'introduction du PBD, de nombreuses méthodes ont été proposées pour étendre son utilisation et atténuer certaines de ses limites. PBD a été utilisée dans la simulation des fluides (Macklin & Müller, 2013) puis un solveur unifié a été proposé pour supporter la simulation

des différentes phases de la matière à la fois (Macklin *et al.*, 2014). Bender *et al.* (2014) ont également dérivé une méthode pour permettre de simuler les propriétés physiques réelles des matériaux en utilisant des contraintes d'énergies. Cette méthode a permis la simulation de phénomènes physiques tels que les déformations élastiques et anisotropiques. Elle repose sur la réduction d'énergie de la méthode FEM discutée dans la section 1.3.2 où cette fois PBD est utilisé pour minimiser l'énergie de déformation au lieu de FEM. Cette méthode fonctionne très bien sur une variété de scénarios de simulation, couvrant une variété de cas allant de la simulation de tissu à la simulation de corps volumétriques. La fonction de contrainte est définie comme l'énergie potentielle due à la déformation qui est calculée de la même manière que dans FEM où l'expression finale est donnée par l'équation 1.24. Le Jacobien de la contrainte n'est rien d'autre que les forces tirés de la littérature FEM et données par l'équation 1.25. En cas de force externe forte, il existe un risque d'inversion tétraédrique. Ce problème a été résolu par Irving *et al.* (2004) où ils appliquent une SVD sur le gradient de déformation F pour supprimer la réflexion :

$$F = U\hat{F}V^{\mathsf{T}} \tag{1.30}$$

U et V sont deux matrices de rotation et $\hat{F}$ une matrice diagonale contenant les valeurs singulières. La réflexion est ainsi supprimée en multipliant par -1 le plus petit élément de $\hat{F}$ et sa colonne correspondante dans U.

La méthode XPBD proposée par Macklin *et al.* (2016) a abordé le problème de la dépendance entre la rigidité et le nombre d'itérations. Elle ajoute à PBD la notion de force de contrainte qui permet d'estimer l'ampleur de la force interne. Dans cette reformulation, une valeur $\Delta\vec{\lambda}$ est calculée à chaque itération. Cette valeur représente un changement incrémentiel vers un multiplicateur total λ qui permet d'imposer la force de la contrainte. Cette nouvelle méthode emploie une reformulation de contrainte conforme (*compliant constraint*) proposée par Servin *et al.* (2006). L'emploi de cette méthode consiste à exprimer l'énergie due à la déformation en fonction d'une contrainte $C(x)$ et une raideur α^{-1}. L'utilisation de cette reformulation

permet de réécrire le problème d'optimisation de PBD en un problème dual de Lagrange où le but est de maximiser une nouvelle fonction objectif sur les multiplicateurs de Lagrange λ. Contrairement à PBD, avec cette nouvelle formulation il est possible de limiter les forces de contraintes en utilisant le terme de conformité comme un terme de régularisation. Ainsi le comportement de la simulation n'est plus dépendant du nombre d'itérations utilisé dans le solveur. Cette reformulation a également l'avantage de rendre possible la simulation des propriétés physiques telles que le module de Young et le coefficient de Poisson en utilisant des contraintes.

Solveurs

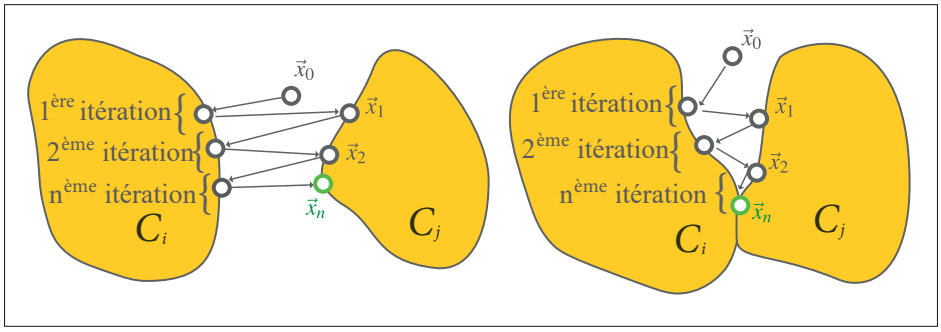


Figure 1.6 Projection de contraintes avec Gauss-Seidel : contraintes non chevauchées (gauche) contraintes chevauchées (droite)

La résolution d'équations constitue une partie importante d'une simulation physique. Les équations mathématiques sont utilisées pour modéliser une propriété ou une relation spécifique d'un phénomène physique. Afin de maintenir ces propriétés, un ensemble d'inconnus est utilisé pour aboutir à une solution qui satisfait les équations mathématiques. Par exemple, dans ce projet de recherche, un solveur à base Gauss-Seidel est utilisé pour résoudre le problème d'optimisation de XPBD. Le solveur de Gauss-Seidel est un solveur itératif utilisé pour résoudre un système linéaire d'équations sous la forme de $A\Delta\vec{\lambda} = b$. Le solveur commence par une estimation initiale $\vec{\lambda}_0 = \vec{0}$ de la solution, et à chaque itération, le solveur rectifie à avec $\Delta\lambda_i$ l'estimation actuelle jusqu'à ce qu'il atteigne la solution finale $\vec{\lambda}^*$. Dans le cadre de XPBD,

le calcul de $\Delta\lambda_i$ est donné par la résolution de l'équation

$$A\Delta\vec{\lambda}_i = \left[\nabla C(\vec{x})\, M^{-1} \nabla C(\vec{x})^{\mathsf{T}} + \frac{\alpha}{h^2}\right] \Delta\vec{\lambda}_i = -C(\vec{x}) - \frac{\alpha}{h^2}\vec{\lambda}_i \tag{1.31}$$

À chaque itération, le solveur Gauss-Seidel boucle sur toutes les contraintes et rectifie les positions pour toutes les particules afin de satisfaire les contraintes. Au cours de cette boucle, le solveur utilise les positions récemment estimées de $\vec{x}$ qui optimisent les contraintes précédentes afin de résoudre les contraintes restantes. Ce processus est visualisé dans la figure 1.6 où il est montré comment le solveur traite les contraintes une par une. La figure 1.6 décrit le comportement du solveur dans deux cas de deux systèmes de contraintes différents. Dans la figure à droite, il s'agit du cas simple où les contraintes se chevauchent et ainsi partagent une solution qui correspond aux deux contraintes. La figure à gauche illustre le comportement d'un solveur avec un système de contraintes non superposées. Dans ce cas, la dernière contrainte résolue sera la plus optimisée comparée aux autres contraintes. Ce qui suggère que l'ordre de résolution des équations peut affecter la solution finale où la solution atteinte n'est pas garantie d'être unique.

Il convient aussi de souligner que la méthode de XPBD est très stable car elle est dérivée d'un schéma d'intégration implicite où les forces de contraintes vont toujours essayer de ramener les particules dans une configuration valide. Toutefois, des instabilités peuvent se produire si le pas de temps h est assez grand et que la raideur des contraintes est aussi grande. Un autre inconvénient hérité c'est l'amortissement numérique où le corps simulé perd son énergie à chaque pas de temps et donc se retrouve à un état immobile après un certain temps.

La méthode de Jacobi a également été utilisée dans ce projet de recherche. Ce solveur a l'avantage de permettre une parallélisation massive sur le GPU pour la résolution du système d'équations donné par 1.31. Cependant, son principal inconvénient est la lenteur de son taux de convergence en comparaison à celui de Gauss-Seidel. La principale différence par rapport au solveur Gauss Seidel est l'utilisation des anciennes positions $\vec{x}_{i-1}$ de l'itération précédente pour rectifier les solutions à l'itération actuelle $\vec{x}_i$. Si des particules sont utilisés dans plusieurs

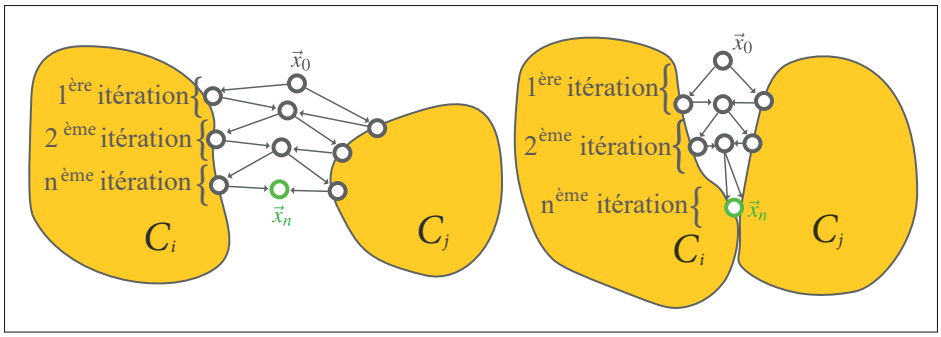


Figure 1.7 Projection de contraintes avec Jacobi : contraintes non chevauchées (gauche) contraintes chevauchées (droite)

contraintes, elles seront projetées sur les espaces de contraintes en utilisant leurs positions $\vec{x}_{i-1}$ de l'itération précédente. Ensuite, une moyenne de toutes les positions projetées est calculée pour estimer les nouvelles positions $\vec{x}_i$. La figure 1.7 illustre le processus de projection des contraintes avec le solveur de Jacobi.

1.3.4 Dynamique projective

La dynamique projective proposée par Bouaziz *et al.* (2014) est une autre approche basée sur des contraintes pour simuler la déformation des corps mous. La méthode repose sur une reformulation des équations de mouvements en un problème de minimisation de l'énergie de déformation donnée par l'équation

$$W(\vec{q},\vec{x}) = \frac{w}{2}\|A\vec{x} - B\vec{q}\|_F^2 + \delta_C(\vec{q}), \tag{1.32}$$

où W est l'énergie mesurant la distance entre la configuration actuelle et la configuration non déformée et A et B sont des matrices constantes.

La solution à ce problème de minimisation repose sur la résolution des deux variables $\vec{x}$ et $\vec{q}$ où un solveur local/global est utilisé pour trouver la bonne solution. Les variables $\vec{x}$ et $\vec{q}$ correspondent respectivement aux positions des particules p et leurs projections sur l'espace des contraintes. À chaque itération, le solveur alterne entre deux étapes, une étape locale et une

étape globale où à chaque étape une seule variable est maintenue fixe et l'autre variable est traitée comme une inconnue.

L'étape de projection locale consiste à fixer les positions $\vec{x}$ et à calculer leurs projections $\vec{q}$ sur les différentes contraintes. Le processus ressemble beaucoup à l'étape de projection effectuée dans PBD où une itération est effectuée sur les contraintes pour projeter les positions une par une sur le *manifold* des contraintes.

Au cours de l'étape globale, les projections $\vec{q}$ sont fixées et les positions $\vec{x}$ sont résolues pour minimiser l'équation 1.32. La solution pour $\vec{x}$ correspond au point critique de l'équation 1.32 où le gradient est égal à zéro.

Bien que cette méthode soit très robuste même dans des circonstances extrêmes, elle souffre de quelques limitations ce qui la rend inadaptée au projet de recherche présenté dans ce mémoire. Premièrement, une des limitations de la dynamique projective est l'amortissement implicite qui provient de la formulation du schéma d'intégration implicite. Il a été également démontré que ce problème est d'autant plus amplifié lorsque le nombre d'itérations dans l'étape globale n'est pas suffisant. Enfin, en raison de la nécessité d'un solveur direct dans l'étape globale, cette méthode ne peut pas tirer profit de la parallélisation GPU.

1.4 Méthodes d'accélération numérique

Les solveurs itératifs, y compris ceux présentés dans cette thèse, ne convergent pas rapidement vers la solution correcte. Pire encore, certains de ces solveurs ne convergent jamais. De nombreuses techniques d'accélération numérique ont été proposées pour améliorer la convergence sans affecter la performance globale de l'algorithme. Ces techniques utilisent les anciennes informations produites par le solveur pour l'aider à atteindre la solution correcte plus rapidement.

1.4.1 Méthode de surrelaxation successive

L'une des méthodes utilisées pour accélérer la convergence d'un solveur itératif est appelée la méthode de sur-relaxation successive. Cette méthode peut être considérée comme une extrapolation appliquée sur le solveur Gauss-Seidel, où l'extrapolation se fait selon la direction obtenue par les deux dernière solutions $(\vec{x}^{k+1} - \vec{x}^{k})$ multipliée par un coeffcient ω. Le coefficient ω supérieur à 1 détermine le degré de l'extrapolation.

1.4.2 Méthode semi-itérative de Chebyshev

La méthode semi-itérative de Chebyshev proposée par Wang (2015) est une autre méthode utilisée pour accélérer la convergence des solveurs itératifs destinés à la simulation des corps mous. Son principe consiste à prendre l'interpolation de m itérations précédentes pour calculer la prochaine itération. Cette méthode repose sur les polynômes de Chebyshev pour trouver les coefficients barycentriques. Plus de détails sur la dérivation de la méthode peuvent être trouvés dans l'article de Wang (2015). Dans le cadre de ce projet de recherche, nous avons expérimenté avec cette méthode afin d'améliorer la convergence de notre solveur mais nous n'avons pas obtenu un gain de performance significatif. Plus de détails concernant les résultats peuvent être trouvés dans le chapitre 3.

1.4.3 Méthode d'Anderson

Cette technique d'accélération numérique, également connue sous le nom d'Anderson *mixing*, a été initialement proposée par Anderson (1965). Cette méthode a été utilisée avec succès dans divers domaines tels que l'électronique, la chimie et récemment en infographie (Peng *et al.*, 2018; Zhang *et al.*, 2019). L'accélération d'Anderson est également connue par ses deux autres variantes, de type I et de type II. Dans ce mémoire, seulement l'accélération de type II sera traitée. L'idée générale de cette méthode consiste à prendre une interpolation linéaire de m solutions précédentes avec des poids dynamiques. Le résultat de l'interpolation vise à accélérer la convergence de la méthode de points fixes.

Dans une méthode de points fixes, le but est de trouver une solution x^* pour une fonction $f(x) = 0$ en cherchant le point fixe x^* pour une fonction $g : \mathbb{R}^n \to \mathbb{R}^n$ tel que

$$x^* = g(x^*) = f(x^*) + (x^*), \tag{1.33}$$

où la définition de $g(x^*)$ vient du fait que l'on ajoute x^* des deux côtés de $f(x^*) = 0$.

Généralement, un moyen simple de trouver la solution à ce problème consiste à évaluer $g(x_i)$ plusieurs fois et espérer que la séquence x_0, x_1, ..., x_n converge vers la solution x^* et l'erreur donnée par l'équation 1.34 converge vers 0.

$$e_i = g(x_i) - x_i \tag{1.34}$$

L'accélération d'Anderson permet d'améliorer la convergence de la séquence en combinant les m dernières solutions obtenues à partir de la séquence. Elle peut également être utilisée pour détecter la divergence de la solution. L'algorithme 1.2 décrit l'algorithme général pour l'accélération Anderson, où x_{AA} est la solution accéléré et e est connu comme la fonction résiduelle donné par l'équation

Algorithme 1.2 Accélération d'Anderson

for $k = 1..n$ **do**
  $x_k = g(x_{k-1})$
  $m_k = min(m, k)$
  $E_k = [e_{k-m_k}|, ..., |e_k]$
  Résoudre le vecteur des coefficients $\alpha = (\alpha_0, ..., \alpha_{m_k})$ pour minimiser :
  $||E_k \alpha||$ tel que $\sum_{i=0}^{m_k} \alpha_i = 1$
  $x_{AA} = \sum_{i=0}^{m_k} \alpha_i g\left(x_{k-m_k+i}\right)$
end

Dans un travail récent, Peng et al. (2018) ont proposé une méthode où l'accélération Anderson a été utilisée afin d'améliorer la convergence du solveur local/global de dynamique projective (section 1.3.4). Les itérations du solveur local/global sont traitées comme une séquence d'itérations de points fixes. L'accélération Anderson est appliquée sur la séquence pour prédire l'itération suivante. Afin d'éviter toute divergence par rapport à la bonne solution, les auteurs ont proposé de calculer l'énergie avant et après l'étape d'accélération et de faire des approximations qui se traduisent par une énergie minimale. Il en résulte une diminution constante de l'énergie et évite ainsi les instabilités ou la stagnation dans la simulation.

Dans ce projet de recherche, XPBD a été adoptée pour simuler la déformation des corps mous car celle-ci offre le meilleur compromis entre précision, efficacité et stabilité. Cependant, puisque le solveur itératif utilisé avec cette méthode souffrait d'une faible convergence, il était très difficile de simuler des corps non souples avec un nombre d'itérations limité. Par conséquent, l'objectif le plus prioritaire consistait à améliorer la convergence du solveur de XPBD afin de pouvoir utiliser moins d'itérations. Nous avons donc investigué les nouvelles méthodes d'accélération numériques qui n'ont pas été précédemment utilisées avec XPBD tel que la méthode de Anderson. Ensuite, une amélioration supplémentaire a été apportée à la convergence en faisant la combinaison de Anderson avec d'autres méthodes d'accélération numériques.

CHAPITRE 2

ANDERSON ET DYNAMIQUE BASÉE SUR LES POSITIONS

Ce chapitre présente l'approche développée dans le cadre de ce mémoire. La section 2.1.1 décrit le simulateur de base utilisé pour la simulation des objets déformables. Ensuite, la section 2.2, présente la contribution de ce travail.

2.1 Simulation d'objets déformables

L'approche présentée dans ce mémoire se base sur la méthode XPBD décrite dans la section 1.3.3. Les modèles utilisés dans ce type de simulation sont des maillages tétraédriques composés par N particules (noeuds) où chaque particule est décrite par une position $\vec{x}_i$, une vitesse $\vec{v}_i$ et un ensemble de contraintes qui l'interconnectent avec d'autres particules. La figure 1.5 présente un exemple de maillage utilisé pour une poutre en porte-à-faux.

2.1.1 Simulation avec la dynamique basée sur les positions

L'algorithme de XPBD a été utilisé pour simuler les forces internes d'un objet déformé qui permettent à l'objet de retourner à son état de repos. Ces forces d'élasticité sont simulées à travers les "forces" de contraintes, où la "force" pour chaque contrainte $C_i(\vec{x})$ est décrite par la direction $\nabla C_i(\vec{x})$ et une amplitude totale égale à λ_i. D'un point de vue mathématique, ces "forces" représentent des déplacements appliqués directement sur les positions des particules afin de résoudre un problème d'optimisation avec les contraintes $C(\vec{x})$.

Dans l'algorithme de XPBD, à chaque cadre de la simulation, les positions et les vitesses sont mises à jour selon l'algorithme 2.1. Comme dans l'algorithme 1.1 de PBD, notre algorithme commence par une intégration explicite sur les vitesses $\vec{v}$ suivie par une intégration implicite sur les positions $\vec{x}$ dans le but de calculer l'effet des forces extérieures tel que la gravité appliquée sur l'objet. Puis, l'algorithme 2.2 ProjectConstraints() est appelé pour simuler l'effet des forces

Algorithme 2.1 XPBD

Fonction : XPBD
Arguments : $\vec{x}^t$, $\vec{\lambda}_0$
Sortie : $\vec{x}^{t+1}$

1 Prédiction de vitesse $\vec{v}^{t+1} = \vec{v}^t + h\vec{a}^t$
2 Prédiction de position $\vec{x}^{t+1} = \vec{x}^t + h\vec{v}^{t+1}$
3 Initialiser les multiplicateurs de Lagrange $\vec{\lambda}_0 \leftarrow 0$
4 $\vec{x}_0 \leftarrow \vec{x}^{t+1}$
5 **for** $i = 0..maxIterations$ **do**
6 $\quad (\vec{x}_{i+1}, \vec{\lambda}_{i+1}) \leftarrow \text{ProjectConstraints}(\vec{x}_i, \vec{\lambda}_i)$
7 **end**
8 Mise à jour des vitesses des particules $\vec{v}^{t+1} = \frac{1}{h}\left(\vec{x}^{t+1} - \vec{x}^t\right)$

d'élasticité sur l'objet à travers la projection des contraintes de *strain*. Finalement les vitesses $\vec{v}$ sont mises à jour pour prendre en considération les forces d'élasticité.

2.1.2 Contrainte basée sur le vecteur de déformation

Afin d'assurer une simulation physiquement plausible, il est important que les contraintes $C(\vec{x})$ utilisées pour simuler les forces internes soient dérivées de la mécanique continue. Pour cela, les contraintes de la reformulation du FEM linéaire introduites par Servin *et al.* (2006) sont employées pour les modèles continus. Cette section explique comment la contrainte $C(\vec{x})$ est construite et présente l'algorithme XPBD utilisé pour la projection des contraintes $C(\vec{x})$.

Pour définir la contrainte $C(\vec{x})$, on commence par définir une énergie potentielle ayant la forme $\frac{1}{2}Kl^2$ où l décrit une quantité de déformation. En partant du fait que le *Green strain* fournit une

bonne mesure de déformation il est possible de définir une énergie potentielle $\psi(\vec{x})$ tel que

$$\psi(\vec{x}) = \frac{1}{2}\vec{\varepsilon}^{\intercal}\alpha^{-1}\vec{\varepsilon}, \tag{2.1}$$

$$\text{où } \vec{\varepsilon} = \begin{pmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ \varepsilon_{yz} \\ \varepsilon_{xz} \\ \varepsilon_{xy} \end{pmatrix} \tag{2.2}$$

et α^{-1} est la matrice de raideur qui contient les propriétés d'élasiticté du matériau utilisé. Cette matrice est aussi appélée l'inverse de conformité, et elle est donnée par

$$\alpha^{-1} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}. \tag{2.3}$$

Dans la matrice α^{-1}, les paramètres λ et μ sont les paramètres de Lamé. Le calcul de ces paramètres est donné par les équations 1.22 et 1.23.

En considérant que la densité $\psi(x)$ est constante sur tout le volume, (ce qui est aussi le cas dans le modèle linéaire de FEM), il est possible de calculer l'énergie totale dans un tétraèdre par l'intégration

$$U = V\psi(x) = \frac{1}{2}\left(V^{\frac{1}{2}}\vec{\varepsilon}\right)^{\intercal}\alpha^{-1}\left(V^{\frac{1}{2}}\vec{\varepsilon}\right). \tag{2.4}$$

Comme l'énergie potentielle pour une contrainte $C(\vec{x})$ peut être calculée par suit

$$U(\vec{x}) = \frac{1}{2} C(\vec{x})^{\intercal} \alpha^{-1} C(\vec{x}), \tag{2.5}$$

on peut identifier en examinant les termes entre l'équation 2.4 et l'équation 2.5 que la contrainte $C(\vec{x})$ qui fournira l'énergie potentielle $\psi(\vec{x})$ est donnée par

$$C(\vec{x}) = \left(V^{\frac{1}{2}} \vec{\varepsilon} \right) \tag{2.6}$$

Une fois que la fonction de la contrainte $C(\vec{x})$ est trouvée, il faut calculer les forces des contraintes qui permettent de projeter les positions des particules sur l'espace des contraintes. Ceci est établi par le mécanisme de projection qui permet de calculer d'une manière incrémentale les forces de contraintes. Ce mécanisme est défini par l'algorithme 2.2.

La projection des contraintes effectuée dans la fonction ProjectConstraints() implique la résolution d'un système d'équations situé à la ligne 8 de l'algorithme 2.2 où α est l'inverse de la matrice de raideur donnée par l'équation 2.3. Ce système d'équations peut être résolu par la méthode de Cholesky sans affecter énormément la performance, car ce système est de petite taille (6x6).

Le mécanisme de projection nécessite également l'évaluation du Jacobien de la contrainte $C(\vec{x})$. En effet, le Jacobien permet de déterminer la direction de déplacement des particules et d'optimiser les contraintes. L'expression du Jacobien est donnée par l'équation 2.7.

Algorithme 2.2 Projection des particules sur l'espace des contraintes

Fonction : ProjectConstraints
Arguments : $\vec{x}_k$, $\vec{\lambda}_k$
Sortie : $\vec{x}_{k+1}$, $\vec{\lambda}_{k+1}$

$\vec{x}_{k+1} \leftarrow \vec{x}_k$
$\vec{\lambda}_{k+1} \leftarrow \vec{\lambda}_k$
for C_j $j \in (1,...,cn)$ **do**
    Calculer la matrice de la configuration courante D_s du tétraèdre j
    Calculer la matrice de la configuration initiale D_m du tétraèdre j
    Calculer le gradient de déformation $F = D_s D_m^{-1}$
    Calcul du gradient $\nabla C_j(\vec{x})$
    Assembler le système d'équations $\left[\nabla C_j(\vec{x}) M^{-1} \nabla C_j(\vec{x})^\intercal + \frac{1}{h^2}\alpha\right] \Delta\vec{\lambda} = -C_j(\vec{x}) - \frac{1}{h^2}\alpha\vec{\lambda}$
    Résoudre pour $\Delta\vec{\lambda}$
    Calculer $\vec{\Delta x} = M^{-1} \nabla C_j(\vec{x})^\intercal \Delta\vec{\lambda}$
    Mettre à jour les multiplicateurs de Lagrange $\vec{\lambda}_{k+1} \leftarrow \vec{\lambda}_{k+1} + \Delta\vec{\lambda}$
    Mettre à jour les positions $\vec{x}_{k+1} \leftarrow \vec{x}_{k+1} + \vec{\Delta x}$
end

$$\frac{\partial \vec{\varepsilon}}{\partial X} = \frac{\partial}{\partial X} \begin{bmatrix} \varepsilon_{\mathrm{N}x} \\ \varepsilon_{\mathrm{N}y} \\ \varepsilon_{\mathrm{N}z} \\ \varepsilon_{\mathrm{S}x} \\ \varepsilon_{\mathrm{S}y} \\ \varepsilon_{\mathrm{S}z} \end{bmatrix} = \begin{bmatrix} \frac{\partial \vec{\varepsilon}_{\mathrm{N}}}{\partial X_0} & \frac{\partial \vec{\varepsilon}_{\mathrm{N}}}{\partial X_1} & \frac{\partial \vec{\varepsilon}_{\mathrm{N}}}{\partial X_2} & \frac{\partial \vec{\varepsilon}_{\mathrm{N}}}{\partial X_3} \\ \frac{\partial \vec{\varepsilon}_{\mathrm{S}}}{\partial X_0} & \frac{\partial \vec{\varepsilon}_{\mathrm{S}}}{\partial X_1} & \frac{\partial \vec{\varepsilon}_{\mathrm{S}}}{\partial X_2} & \frac{\partial \vec{\varepsilon}_{\mathrm{S}}}{\partial X_3} \end{bmatrix} \tag{2.7}$$

Le résultat du Jacobien est une matrice de dimensions 6x12 qui représente un mappage entre l'ensemble des six contraintes et les positions des quatre sommets d'un tétraèdre. Chaque bloc

dans cette matrice est de dimension 3x3 où le calcul de chaque bloc est donné par

$$\begin{aligned} \frac{\partial \vec{\varepsilon}_{\mathrm{N}}}{\partial X_i} &= \sqrt{V}\Lambda_{\mathrm{N}}^i F^{\intercal} + \frac{1}{2\sqrt{(V)}}\vec{\varepsilon}_{\mathrm{N}}\frac{\partial V}{\partial X_i} \\ &, \\ \frac{\partial \vec{\varepsilon}_{\mathrm{S}}}{\partial X_i} &= \sqrt{V}\Lambda_{S}^i F^{\intercal} + \frac{1}{2\sqrt{(V)}}\vec{\varepsilon}_{\mathrm{S}}\frac{\partial V}{\partial X_i} \end{aligned} \tag{2.8}$$

où

$$\begin{aligned} \Lambda_N^i &= \begin{bmatrix} y_{ix} & 0 & 0 \\ 0 & y_{iy} & 0 \\ 0 & 0 & y_{iz} \end{bmatrix}, \\ \Lambda_S^i &= \begin{bmatrix} 0 & y_{iz} & y_{iy} \\ y_{iz} & 0 & y_{ix} \\ y_{iy} & y_{ix} & 0 \end{bmatrix}, \end{aligned} \tag{2.9}$$

et y_{ix}, y_{iy}, y_{iz} sont les coefficients x,y,z de la ième ligne de la matrice de la configuration au repos du tétraèdre D_m. Les matrices D_s et D_m sont les matrices données par l'équation 1.16.

Le gradient du volume V par rapport aux positions $\vec{x}_1, \vec{x}_2, \vec{x}_3, \vec{x}_4$ est calculé avec les équations suivantes :

$$\begin{aligned} \frac{\partial V}{\partial \vec{x}_1} &= \frac{1}{6}(\vec{x}_2 - \vec{x}_0) \times (\vec{x}_3 - \vec{x}_0) \\ \frac{\partial V}{\partial \vec{x}_2} &= \frac{1}{6}(\vec{x}_3 - \vec{x}_0) \times (\vec{x}_1 - \vec{x}_0) \\ \frac{\partial V}{\partial \vec{x}_3} &= \frac{1}{6}(\vec{x}_1 - \vec{x}_0) \times (\vec{x}_2 - \vec{x}_0) \\ \frac{\partial V}{\partial \vec{x}_0} &= -\frac{\partial V}{\partial \vec{x}_1} - \frac{\partial V}{\partial \vec{x}_2} - \frac{\partial V}{\partial \vec{x}_3} \end{aligned} \tag{2.10}$$

Il convient aussi de souligner que l'avantage des contraintes basées sur le strain comparé aux contraintes énergétiques (Bender *et al.*, 2014) est un meilleur contrôle sur l'étirement et le cisaillement pour chaque tétraèdre sur les axes x,y,z, rendant ainsi le problème de XPBD mieux déterminé.

2.1.3 Implémentation paralèlle

Dans le cadre de ce projet de recherche, deux types de solveurs ont été implémentés : (i) un solveur parallèle exécuté sur GPU utilisant la méthode de Jacobi, et (ii) un solveur séquentiel exécuté sur CPU qui utilise la méthode de Gauss-Seidel. L'implémentation parallèle de l'algorithme 2.1 a été basée sur une version modifiée du solveur Jacobi de Bridson *et al.* (2002). La parallélisation peut être basée soit sur des particules, soit sur des contraintes. Dans ce mémoire, les deux types de paralléllisation ont été testés et il s'est avéré que la paraléllisation à base de contraintes est beaucoup plus efficace. En effet, dans la plupart des expérimentations, le nombre de contraintes est beaucoup plus important que le nombre de particules ce qui constitue un avantage pour le calcul parallèle notamment la distribution des contraintes sur les threads du GPU. Concernant l'implémentation du solveur à base de contraintes, au lieu d'utiliser les *atomic operations* qui permettent la synchronisation entre les threads GPU, les Δx_i pour chaque particule impliquée dans une contrainte ont été collectés pour chaque itération sur les contraintes. Une fois toutes les contraintes traitées, la moyenne des Δx_i correspondant aux particules est calculée afin d'obtenir le nouvel état de l'objet déformé.

2.2 Accéléreration du solveur avec la méthode d'Anderson

L'une des principales limitations lors de la résolution du problème d'optimisation de XPBD se manifeste dans le ralentissement de la convergence après quelques itérations. Ceci est principalement dû à la nature du solveur Gauss-Seidel utilisé. Pour résoudre ce problème, de nombreuses méthodes ont été proposées telles que les méthodes discutées dans la section 1.4. Étant donné que le solveur de XPBD peut être considéré comme une méthode de point fixe, la convergence de l'algorithme XPBD peut être accélérée par l'emploi de l'accélération

d'Anderson. En effet, aux lignes 11 et 12 de l'algorithme 2.2, XPBD produit une séquence d'itérations pour les positions ainsi que pour les multiplicateurs de Lagrange afin d'atteindre une solution fixe $(\vec{x}^*, \vec{\lambda}^*)$, étant donné la mise à jour

$$(\vec{x}^*, \vec{\lambda}^*) \leftarrow \text{ProjectConstraints}(\vec{x}^*, \vec{\lambda}^*),$$

où ProjectConstraint() constitue une itération de l'algorithme 2.2.

Dans ce projet de recherche, l'application de l'accélération d'Anderson a été étudiée à la fois sur la séquence des positions ainsi que sur les multiplicateurs de Lagrange produits par le solveur. C'est la première fois à notre connaissance que l'accélération d'Anderson est utilisée avec XPBD. Bien que ces deux approches aient amélioré le taux de convergence, l'accélération utilisant des multiplicateurs de Lagrange a abouti à des simulations plus stables que celles utilisant les positions. L'accélération utilisant uniquement les positions a entraîné des instabilités, même en utilisant une série de précautions. Cette section présente comment l'accélération d'Anderson de la section 1.4.3 a été adaptée au solveur XPBD pour améliorer le taux de convergence global.

2.2.1 Accélération appliquée sur les multiplicateurs de Lagrange

Comme décrit dans la section 1.4.3, la convergence de l'algorithme d'Anderson peut être identifiée par l'évaluation de l'erreur donnée par l'équation 1.34. Cette erreur doit converger vers zéro une fois que le solveur atteint la solution exacte. Dans le cas du simulateur XPBD implémenté, le calcul de l'erreur est donné par

$$\vec{e}(\Delta\vec{\lambda})_{k+1} = \Delta\vec{\lambda}_{k+1} - \Delta\vec{\lambda}_k, \tag{2.11}$$

où $\Delta\vec{\lambda}_k$ correspond à l'étape 9 de l'algorithme 2.2 et contstitue l'incrémentation des multiplicateurs de Lagrange obtenus après une itération de projection de contrainte.

L'idée principale dans notre approche consiste à utiliser l'accélération d'Anderson pour minimiser l'erreur définie par l'équation 2.11, et ce en trouvant une interpolation barycentrique entre l'erreur actuelle ainsi que les m dernières erreurs obtenues par le solveur de XPBD. Ce problème peut être reformulé en un problème de moindres carrés, où les coefficients barycentriques θ_j sont obtenus par la résolution de l'équation

$$\vec{\theta} = (\theta_1, \ldots, \theta_m) = \underset{\theta_1 \ldots \theta_m}{\arg\min} \left\| \vec{e}_k - \sum_{j=1}^{m} \theta_j (\vec{e}_{k-j+1} - \vec{e}_{k-j}) \right\|^2 . \tag{2.12}$$

L'équation 2.12 peut être résolue en utilisant l'équation normale correspondante, donnée par

$$(\Delta e^{\mathsf{T}} \Delta e)\, \theta = \Delta e^{\mathsf{T}} \vec{e}_k \tag{2.13}$$

où

$$\Delta e = [\vec{e}_k - \vec{e}_{k-1} | \vec{e}_{k-1} - \vec{e}_{k-2} | \ldots | \vec{e}_{k-m+1} - \vec{e}_{k-m}] . \tag{2.14}$$

La solution $\vec{\theta}^*$ de l'équation 2.13 ainsi que l'inverse de la matrice $A = \Delta e^{\mathsf{T}} \Delta e$ sont calculés à travers une décomposition orthogonale complète qui offre une bonne précision et peut également fournir une estimation de la solution pour les systèmes surdéterminés et sous-déterminés. Une fois la solution $\vec{\theta}^*$ est trouvée, il est possible d'estimer les prochaines incrémentations des multiplicateurs de Lagrange $\vec{\lambda}$ avec le calcul suivant

$$\Delta\vec{\lambda}_{k+1}^{\,\mathrm{AA}} = \Delta\vec{\lambda}_{k+1} - \sum_{j=1}^{m} \theta_j^* (\Delta\vec{\lambda}_{k-j+1} - \Delta\vec{\lambda}_{k-j}). \tag{2.15}$$

Une fois que les nouvelles incrémentations $\Delta\vec{\lambda}^{\,AA}$ sont trouvées, il est possible d'appliquer la technique de SOR avec un coefficient w_{SOR} sur les incrémentations $\Delta\vec{\lambda}^{\,AA}$ avant de mettre à jour les positions des particules. Ceci est réalisé comme suit :

$$\begin{aligned} \vec{\Delta\lambda}^{AA} &\leftarrow w_{SOR} \cdot \vec{\Delta\lambda}^{AA} \\ \vec{\Delta x} &= M^{-1} \nabla C(x)^{\intercal} \vec{\Delta\lambda}^{AA} \end{aligned} \tag{2.16}$$

Puis, les multiplicateurs de Lagrange sont mis à jour

$$\vec{\lambda}_{k+1} \leftarrow \vec{\lambda}_{k+1} + \vec{\Delta\lambda}^{AA}. \tag{2.17}$$

Bien que dans de nombreux cas l'accélération d'Anderson aide à améliorer le taux de convergence d'un solveur donné, il n'y a aucune preuve que cette méthode garantisse toujours une convergence globale ou locale. Dans le cas où elle provoque une stagnation, la simulation physique peut devenir instable et entraîner des oscillations.

2.3 Stratégies envisagées pour améliorer la convergence avec Anderson

Algorithme 2.3 Accélération d'Anderson

Fonction : Anderson
Arguments : $\vec{\Delta\lambda}_i$
Sortie : $\vec{\Delta\lambda}^{AA}$

1 Calculer l'erreur à l'itération i : $\vec{e}(\vec{\Delta\lambda}_i)$
2 Assembler les m dernières erreurs dans une matrice :
$\Delta e = [\vec{e}_i - \vec{e}_{i-1} | \vec{e}_{i-1} - \vec{e}_{i-2} | ... | \vec{e}_{i-m+1} - \vec{e}_{i-m}]$
3 Assembler le système d'équations $(\Delta e^{\intercal} \Delta e)\,\vec{\theta} = \Delta e^{\intercal} \vec{e}_k$
4 Tant que nombre de conditionnement de $(\Delta e^{\intercal} \Delta e) > 10^3$ et nombre de colonnes $cols > 2$:
5 Supprimer les anciennes colonnes de la matrice Δe
6 Résoudre pour $\vec{\theta}$ le système $(\Delta e^{\intercal} \Delta e)\,\vec{\theta} = \Delta e^{\intercal} \vec{e^k}$
7 Appliquer l'accélération d'Anderson sur $\vec{\Delta\lambda}$:
$\vec{\Delta\lambda}^{AA} = \vec{\Delta\lambda}_i - \sum_{j=1}^{m} \theta_j^* (\vec{\Delta\lambda}_{i-j+1} - \vec{\Delta\lambda}_{i-j})$

Algorithme 2.4 XPBD avec l'accélération d'Anderson

Fonction : XPBDAnderson
Arguments : $\vec{x}^t$, $\vec{\lambda}_0$
Sortie : $\vec{x}^{t+1}$

Prédiction de vitesse $\vec{v}^{t+1} = \vec{v}^t + h\vec{a}^t$
Prédiction de position $\vec{x}^{t+1} = \vec{x}^t + h\vec{v}^{t+1}$
Initialiser les multiplicateurs de Lagrange $\vec{\lambda}_0 \leftarrow 0$
$\vec{x}_0 \leftarrow \vec{x}^{t+1}$
for $i = 0..maxIterations$ **do**
    if $i >= m$ **then**
        $\vec{\Delta\lambda}^{\,\mathrm{AA}} \leftarrow w_{SOR} \cdot \mathrm{Anderson}(\vec{\Delta\lambda}_i)$
        $\vec{\Delta x}^{AA} = M^{-1}\nabla C_j(x)^{\intercal}\vec{\Delta\lambda}^{\,\mathrm{AA}}$
        $\vec{x}_{i+1} \leftarrow \vec{x}_i + \vec{\Delta x}^{AA}$
        $\vec{\lambda}_{i+1} \leftarrow \vec{\lambda}_i + \vec{\Delta\lambda}^{\,AA}$
        if $r(\vec{x}_{i+1}\ ,\ \vec{\lambda}_{i+1}) >= r(\vec{x}_i\ ,\ \vec{\lambda}_i)$ **then**
            $(\vec{x}_{i+1}\ ,\ \vec{\lambda}_{i+1}) = \mathrm{ProjectConstraints}(\vec{x}_i\ ,\ \vec{\lambda}_i)$
        end
        else
            $(\vec{x}_{i+1}\ ,\ \vec{\lambda}_{i+1}) = \mathrm{ProjectConstraints}(\vec{x}_{i+1}\ ,\ \vec{\lambda}_{i+1})$
        end
    end
    else
        $(\vec{x}_{i+1}\ ,\ \vec{\lambda}_{i+1}) = \mathrm{ProjectConstraints}(\vec{x}_i\ ,\ \vec{\lambda}_i)$
    end
end
Mise à jour des vitesses des particules $\vec{v}^{t+1} = \frac{1}{h}\left(\vec{x}^{t+1} - \vec{x}^t\right)$

Plusieurs mesures ont été proposées par Walker (2011) pour améliorer la stabilité de l'accélération. Ainsi la méthode Anderson n'est utilisée que lorsque certains critères sont satisfaits. Cependant ces critères peuvent varier selon le scénario. Tout d'abord, le résiduel

global $r(\vec{x}, \vec{\lambda})$ est utilisé comme indicateur pour évaluer si l'itération accélérée est acceptable ou non. Le résiduel à une itération k est donné par

$$r_k(\vec{x}, \vec{\lambda}) = C(\vec{x}) + \tilde{\alpha}\vec{\lambda}_k, \tag{2.18}$$

où $C(\vec{x})$ est la contrainte donnée par l'équation 2.6. À chaque itération, la norme du résiduel $||r(\vec{x}, \vec{\lambda})||_2$ est évalué avant et après l'accélération d'Anderson, et ce pour vérifier si la nouvelle prédiction produite par l'accélération augmente le résiduel. Dans le cas où le résiduel a augmenté, la solution prédite de l'accélération d'Anderson est annulée et la solution produite par le solveur XPBD est utilisée dans l'itération suivante (algorithme 2.4 : lignes : 11 -> 16). Cette stratégie est inspirée de Peng *et al.* (2018) où ils appliquent la même stratégie afin de maintenir une diminution constante de l'énergie en dynamique projective.

Nous utilisons une autre mesure pour maintenir la stabilité : l'accélération n'est utilisée que lorsque la fenêtre d'Anderson avec les m itérations précédentes est remplie. Ceci permet de fournir à l'algorithme d'Anderson plus d'informations sur les itérations précédentes produites par le solveur. (algorithme 2.4 : lignes : 6 à 17)

La taille de la fenêtre contenant les m itérations précédentes peut également affecter la convergence du système. En choisissant une grande fenêtre, une solution très ancienne est inclue et peut dévier la solution accélérée de la vraie solution. D'autre part une petite fenêtre peut également limiter le taux d'accélération ou dégrader le conditionnement. Dans ce projet de recherche, une taille de fenêtre $m = 5$ fournit une bonne performance dans la plupart des cas. Une autre stratégie souvent utilisée avec Anderson consiste à redémarrer périodiquement l'algorithme d'Anderson afin de réinitialiser la fenêtre avec des informations mises à jour. Ceci peut être amélioré encore en utilisant la méthode proposée par Yang *et al.* (2009). Cette méthode consiste à supprimer les anciennes informations contenues dans la fenêtre chaque fois où le nombre de conditionnement de la matrice $A = \Delta e^{\mathsf{T}} \Delta e$ dépasse un certain seuil $s = 10^3$ (algorithme 2.3 : lignes : 4 à 5). Le calcul du nombre de conditionnement pour une matrice A peut être réalisé par le calcul de la décomposition en valeur singulière de la matrice et ensuite

par le calcul du rapport entre la valeur singulière la plus grande et la valeur la plus petite. La valeur du rapport reflète la tendance de la matrice à être inversible et donc la capacité d'un solveur donné à produire une solution proche de la solution réelle.

CHAPITRE 3

RÉSULTATS ET COMPARAISONS

Ce chapitre présente les résultats obtenus lors du processus de validation de l'approche proposée. L'approche a été validée en exécutant plusieurs simulations avec différents paramètres. Les objets 3D utilisés dans ce projet de recherche ont été soigneusement sélectionnés pour qu'ils soient bien configurés. Dans le contexte du projet, la configuration d'un maillage renvoie à la distribution des tétraèdres dans le maillage simulé. Elle peut affecter le comportement de la déformation pendant la simulation. Une mauvaise configuration peut conduire à certains artefacts tel que l'artefact de *locking* où l'objet peut rester bloqué dans un certain état lorsqu'il est déformé. Une bonne configuration aide à ce que les déformations restent toujours plausibles lorsque l'objet est soumis à diverses déformations. Afin de valider le comportement de déformation obtenu avec XPBD, nous avons effectué des simulations similaires avec la méthode FEM en utilisant la bibliothèque VegaFEM (Sin *et al.*, 2013). Dans les expérimentations faites, les deux approches on produit approximativement le même comportement. Toutefois, en raison du schéma d'intégration implémenté, la méthode de FEM se distingue par une simulation beaucoup plus amortie comparée à celle de XPBD. Le

Tableau 3.1 Charactéristiques des maillages utilisés

Maillage	Nb Particules	Nb Contraintes	Dimensions
Poutre en porte-à-faux (résolution 1)	216	460	11.5x1.0x1.0 cm
Poutre en porte-à-faux (résolution 2)	512	1 395	10.3x1.0x1.0 cm
Poutre en porte-à-faux (résolution 3)	1 000	3 120	9.75x1.0x1.0 cm
Hippopotame	2 885	10 361	17.3x9.0x5.8 cm
Armadillo	14 779	54 855	13.5x8.6x10.4 cm

tableau 3.1 présente les spécifications des maillages utilisés. Parmi ces maillages, il existe trois maillages pour la poutre en porte-à-faux où la résolution spatiale des tétraèdres est variable. Dans toutes les expérimentations, seules les contraintes basées sur le *strain* ont été utilisées. Le tableau 3.2 contient la liste des paramètres utilisateur qui peuvent être ajustés.

Certaines propriétés telles que la densité ρ et le coefficient de Poisson v ont été fixées pour les expérimentations présentées aux figures 3.1, 3.2 et 3.3 afin de se rapprocher le plus possible des propriétés matérielles des organes humains. Les tableaux en annexe présentent des approximations de modules de Young pour de vrais matériaux.

Tableau 3.2 Paramètres de simulation utilisés

Maillage	Masse	Densité	Module de Young	Coefficient de Poisson
Poutre en porte-à-faux	11.5 g	1 000 kg.m^{-3}	[10^3,10^9] Pa	0.4
Armadillo	125.0 g	1 027 kg.m^{-3}	[10^3,10^9] Pa	0.4
Hippopotame	350.0 g	1 227 kg.m^{-3}	[10^3,10^9] Pa	0.4

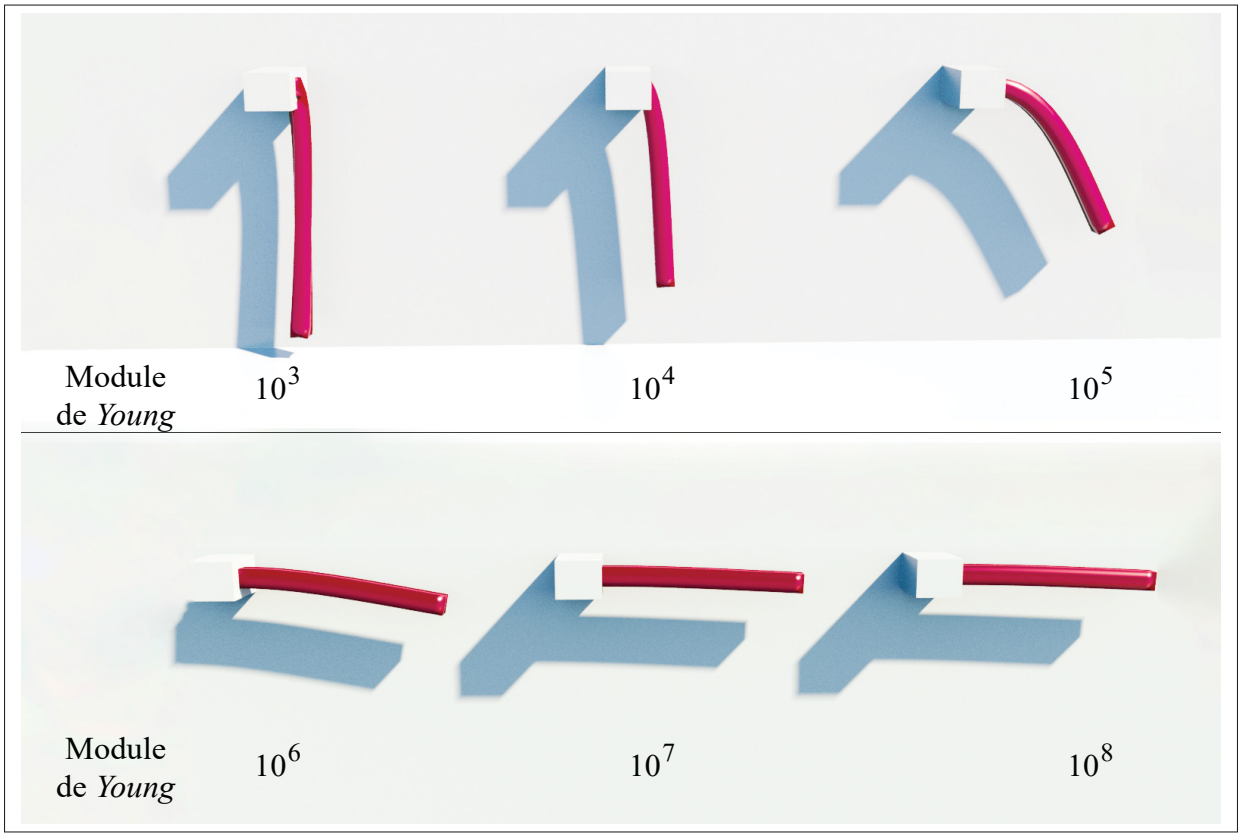


Figure 3.1 Déformation de la poutre en porte-à-faux pour différentes valeurs de module de Young

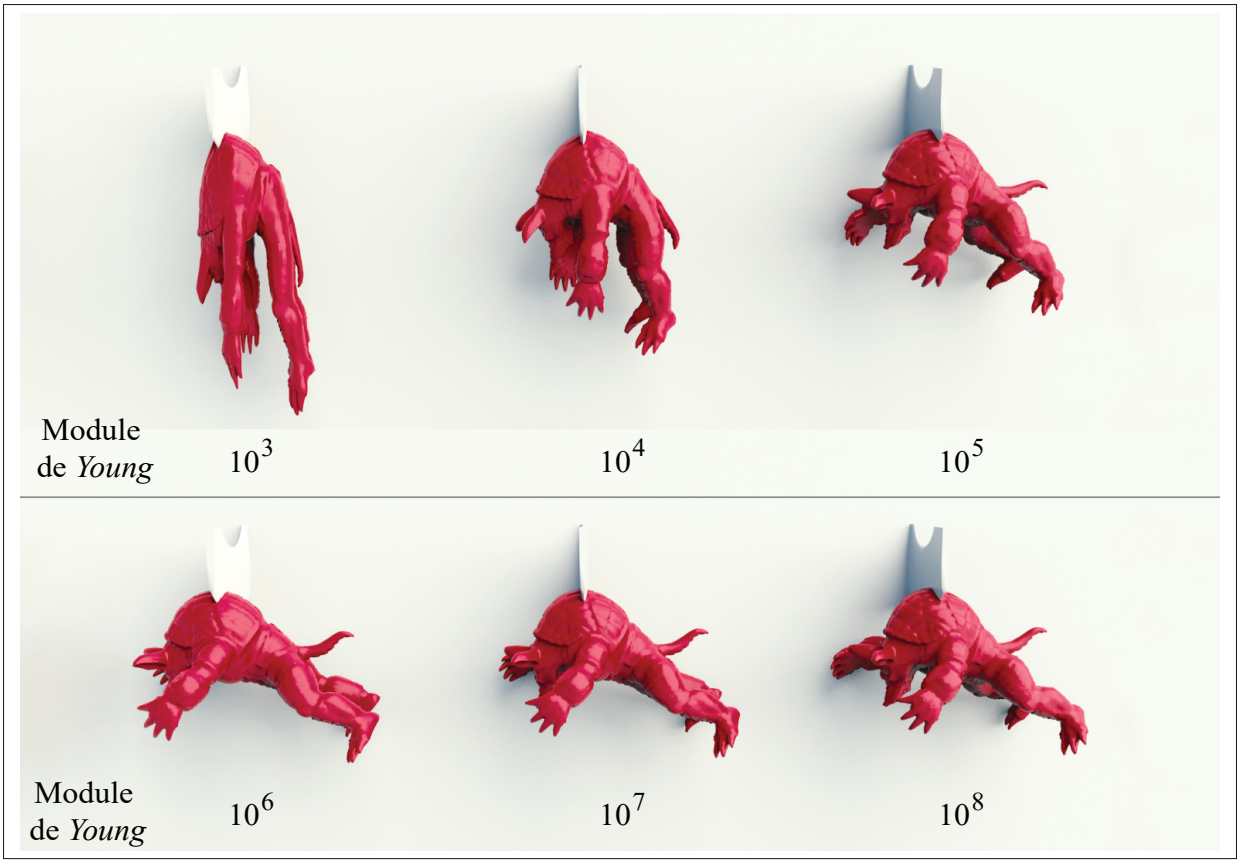


Figure 3.2 Déformation de l'armadillo pour différentes valeurs de module de Young

Les figures 3.1, 3.2 et 3.3 visualisent l'effet de la variation du module de Young sur la déformation des objets lorsqu'ils sont soumis à la force de gravité. Ces figures montrent bien que pour des valeurs faibles du module de Young, l'objet devient moins résistant aux forces externes et donc devient plus souple. D'autre part, pour les valeurs grandes de module de Young, les forces de contraintes deviennent plus importantes afin de résister plus aux forces externes.

La figure 3.4 visualise l'effet de la variation du coefficient de Poisson sur la déformation de l'objet où l'objet subit une compression latérale lorsqu'il est étiré. Il est montré que pour les valeurs de coefficient de Poisson proches de $v = 0.5$, l'objet subit une compression latérale plus importante.

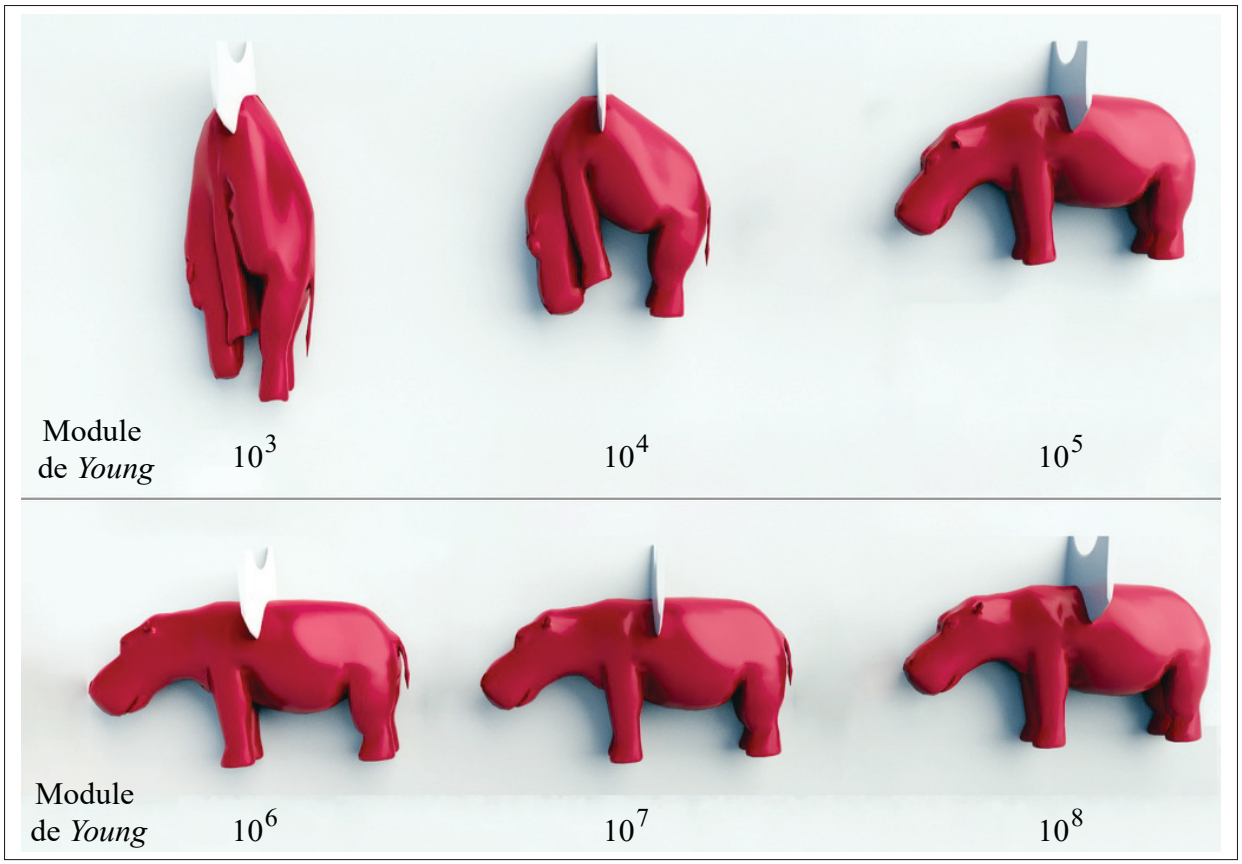


Figure 3.3 Déformation de l'hippopotame pour différentes valeurs de module de Young

Les figures 3.5, 3.6, 3.7, 3.8 et 3.9 comparent la performance des différentes techniques d'accélération numérique lorsqu'elles sont appliquées au solveur de XPBD. Dans ces figures, nous montrons les courbes de convergence pour chacun des techniques d'accélération présentés dans ce mémoire. La première courbe «Anderson SOR» montre la convergence de l'approche proposée qui consiste à appliquer l'accélération de Anderson et le SOR sur les multiplicateurs de Lagrange. La courbe de «Anderson» montre la convergence du solveur lorsque seule l'accélération d'Anderson est appliquée sur les multiplicateurs de Lagrange.

Les deux courbes «SOR» et «Lambda SOR» montrent respectivement la convergence du solveur lorsque l'on utilise le SOR sur les positions et le SOR sur les multiplicateurs de Lagrange. Finalement, la courbe «Chebyshev» montre l'impact de l'accélération de Chebyshev sur la convergence. Pour mesurer la performance de chaque technique d'accélération, la

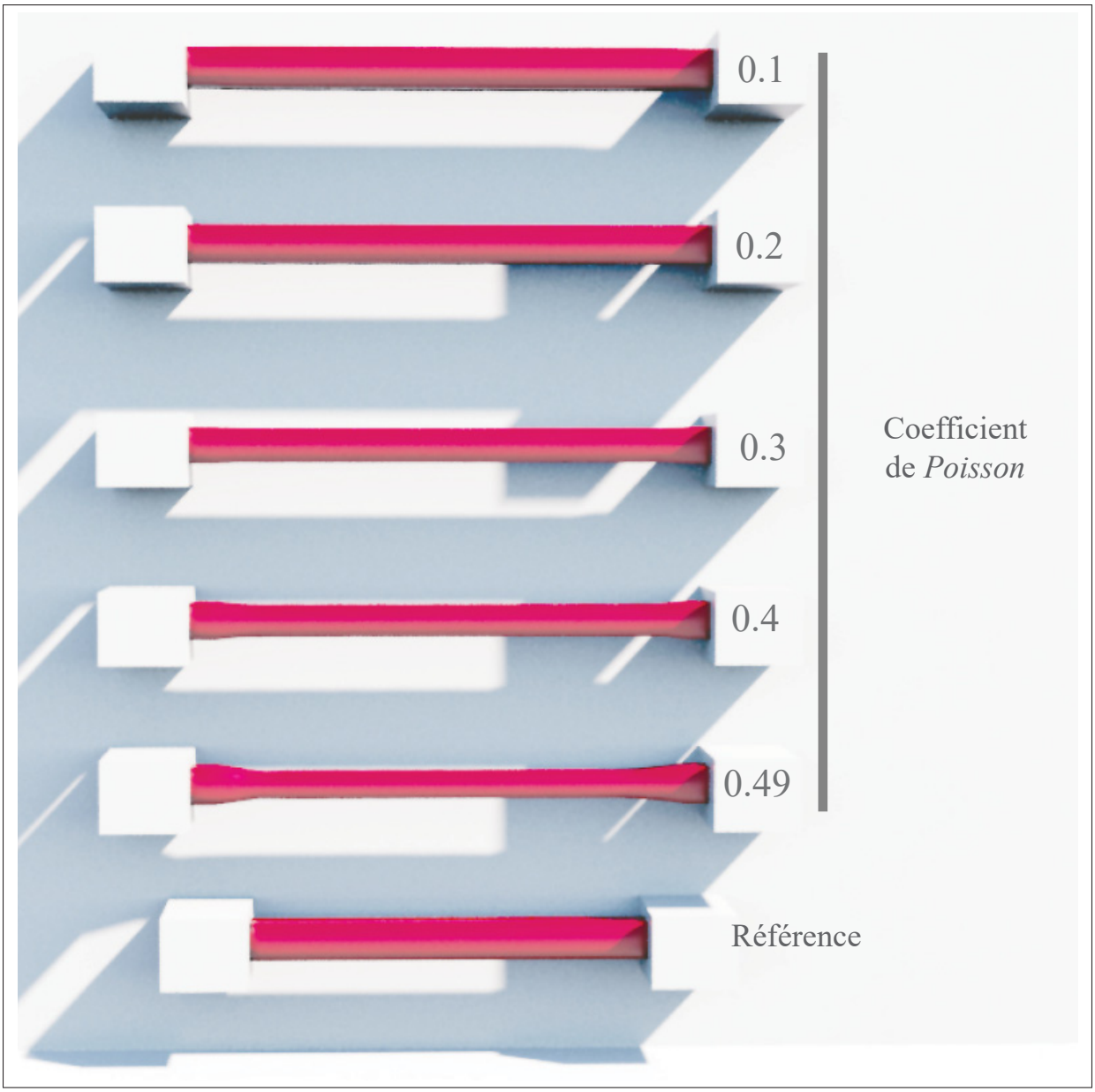


Figure 3.4 Effet du coefficient de Poisson *v* sur la déformation : L'objet subit une contraction latérale plus importante lorsque le coefficient de Poisson est augmenté

simulation démarre avec l'objet dans un état déformé. Ensuite, à chaque itération, le résidu donné par l'équation 2.18 est enregistré. Plus le taux de diminution de résidu est élevé, plus le taux de convergence est important. Comme le montrent les figures 3.5 et 3.8, le solveur diverge lorsque le SOR est utilisé à lui seul avec les multiplicateurs de Lagrange (Lambda SOR). En

fait, cette méthode divergeait dans tous les tests effectués, raison pour laquelle elle a été retirée des autres graphiques.

D'autre part, quand les accélérations de Chebyshev et SOR sont appliquées sur les positions, elles n'ont pas une grande amélioration sur le taux de convergence comparativement à celui de XPBD sans accélération. Par contre, l'accélération de Anderson améliore considérablement la convergence du solveur. La convergence est encore plus optimale lorsque l'algorithme de Anderson est combiné avec SOR. Afin de trouver la meilleure valeur pour le coefficient de SOR, nous avons mené de nombreuses expériences avec différentes valeurs de w_{SOR}. Il y était surprenant de constater qu'un coefficient de 10 pour w_{SOR} sur les $\Delta\lambda_{AA}$ avait le meilleur taux de convergence tout en maintenant la simulation stable. Dans ce cas, le taux de convergence est allé cinq fois plus rapide que celui du XPBD sans accélération.

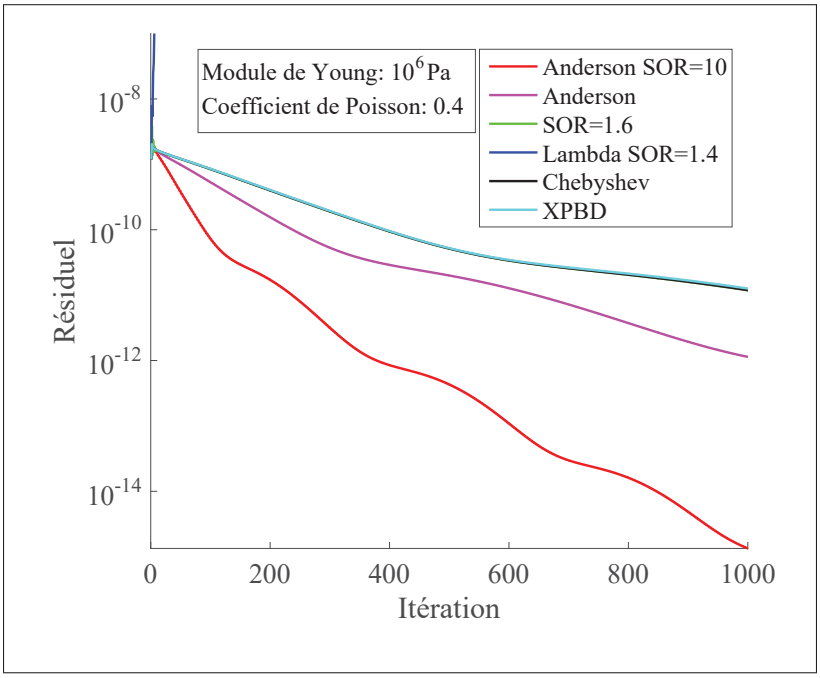


Figure 3.5 Convergence avec la poutre en porte-à-faux (460 contraintes) : courbe de Lambda SOR (en bleu foncé) diverge vers l'infini, courbe de SOR (verte) est superposée avec la courbe de XPBD (bleue)

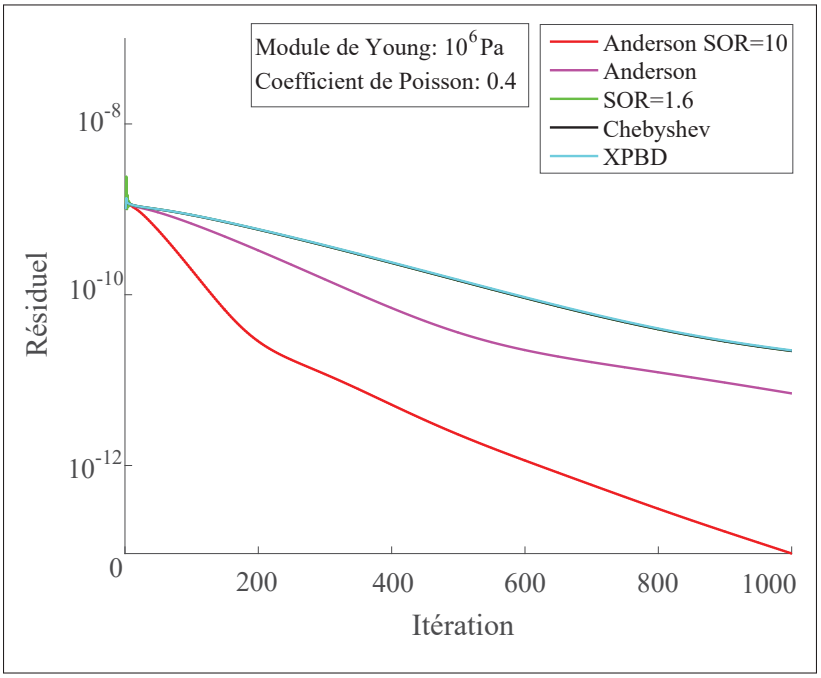


Figure 3.6 Convergence avec la poutre en porte-à-faux (1395 contraintes) : courbe de SOR (verte) est superposée avec la courbe de XPBD (bleue)

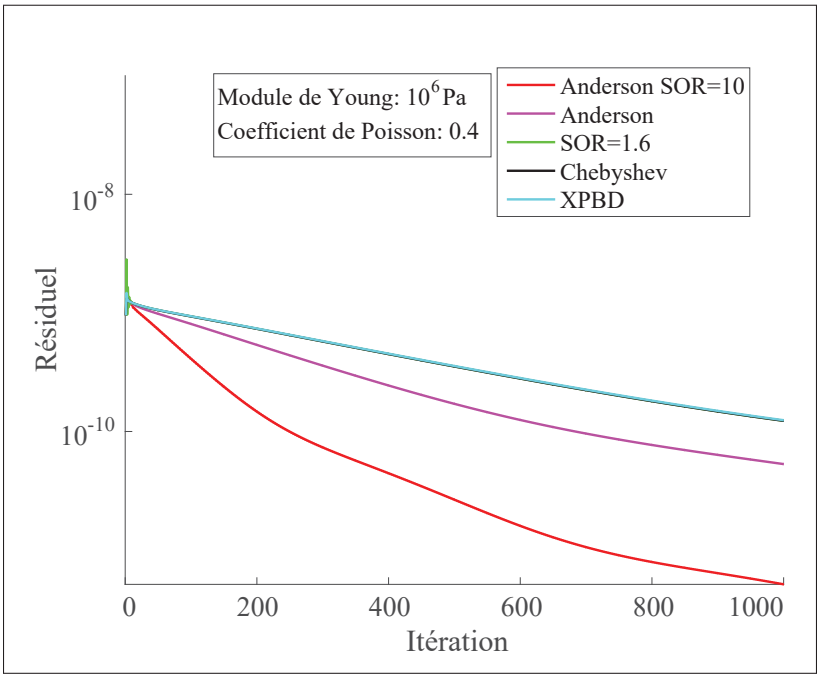


Figure 3.7 Convergence avec la poutre en porte-à-faux (3120 contraintes) : courbe de SOR (verte) est superposée avec la courbe de XPBD (bleue)

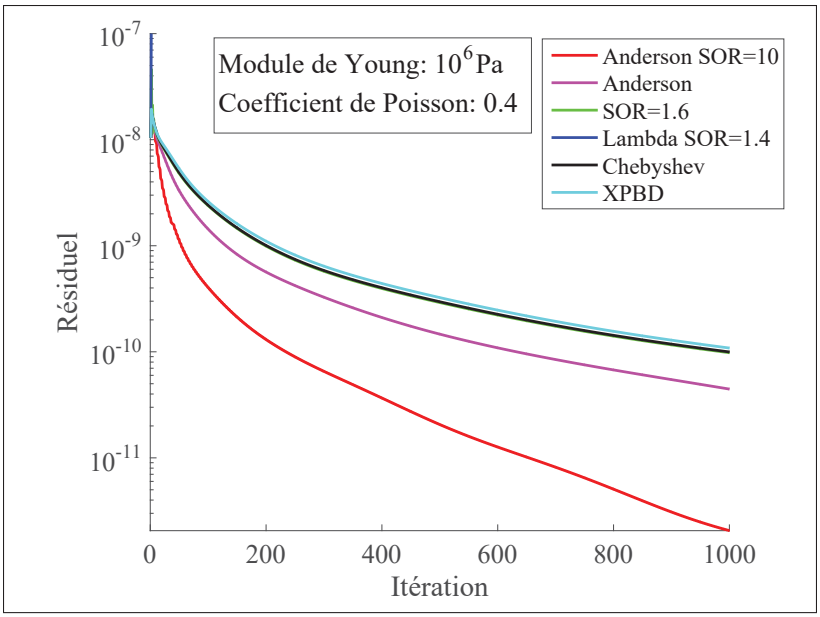


Figure 3.8 Convergence avec l'hippopotame : courbe de Lambda SOR (en bleu foncé) diverge vers l'infini - courbe de SOR (verte) est superposée avec la courbe de XPBD (bleue)

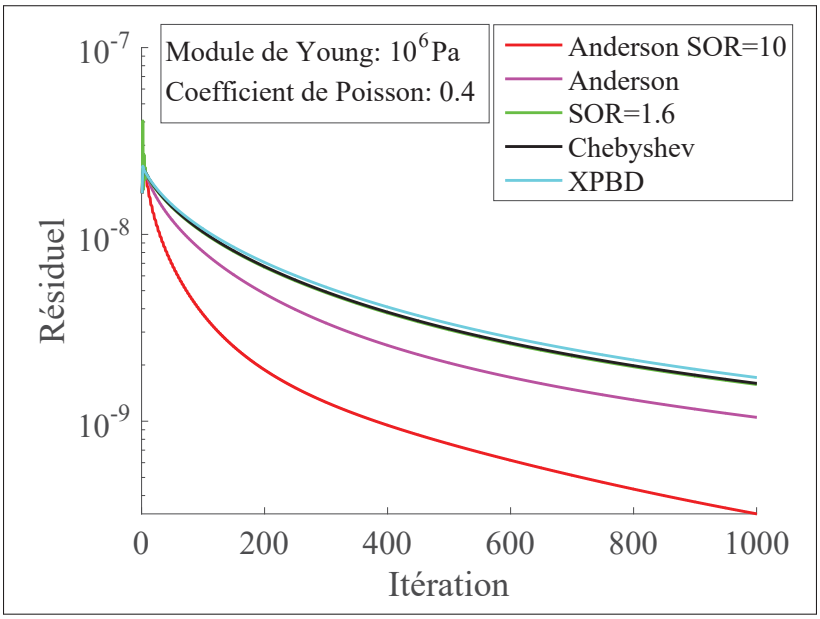


Figure 3.9 Convergence avec l'armadillo : courbe de SOR est superposée avec la courbe de XPBD

La figure 3.10 montre le temps de calcul des différentes étapes impliquées dans l'accélération de Anderson. Tous les temps de calcul présentés sont pour une implémentation séquentielle pure. Le graphique montre un coût pour la méthode développée deux fois supérieur à celui du solveur XPBD original. Et pourtant, notre méthode réussit à compenser le coût supplémentaire car elle permet d'améliorer le taux de convergence du solveur de 3 à 5 fois comme le montrent les graphiques 3.5, 3.6, 3.7, 3.8 et 3.9. D'autre part, une grande part du nouveau code d'Anderson peut être parallélisée, ce qui réduit le temps de calcul total. À titre d'exemple, les parties « CalculRésiduel » et « MisePosition » correspondent respectivement à l'étape de calcul du résidu et l'étape de la mise à jour des positions des particules par les multiplicateurs de Lagrange accélérés. Ces deux étapes qui représentent une surcharge supplémentaire peuvent profiter d'une parallélisation massive sur le GPU.

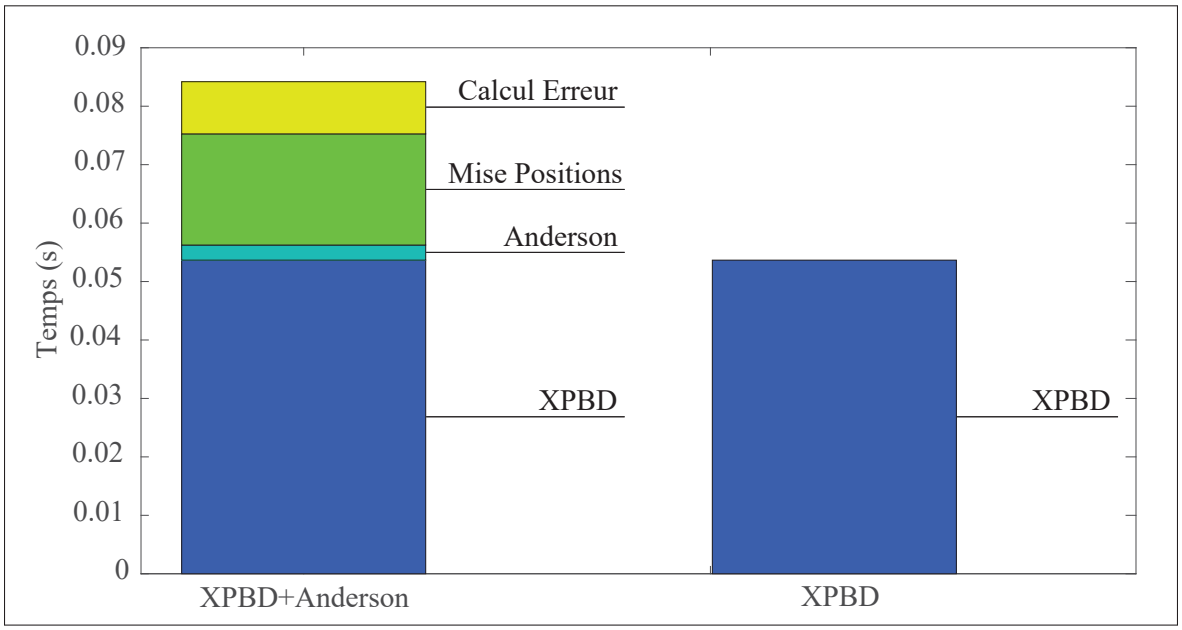


Figure 3.10 Temps de calcul avec Anderson vs XPBD. L'étape « XPBD » correpond à l'étape de projection de contraintes. L'étape « Anderson » correpond à l'algorithme d'accélération d'Anderson. L'étape « MisePositions » correspond à la mise à jour des positions avec les nouveaux multiplicateurs de Lagrange accéléré. L'étape « CalculErreur » correpond au calcul du résiduel

Comme démontré dans ce chapitre, notre approche améliore les performances globales du XPBD ceci en appliquant à la fois la technique d'accélération d'Anderson et le SOR sur les multiplicateurs de Lagrange. Ainsi, réduisant le nombre total d'itérations nécessaires pour simuler la déformation des corps non mous. D'autre part l'emploi, des stratégies présentées dans la section 2.3, a permis au solveur de simuler la déformation des corps mous sans affecter ni le comportement de la déformation du matériau ni la stabilité de la simulation.

CHAPITRE 4

LIMITATIONS

La limitation la plus importante dans l'approche proposée est le non support des grandes forces externes par l'implémentation actuelle, car celles ci peuvent engendrer l'inversion des tétraèdres qui peuvent à leur tour causer l'instabilité de la simulation. La raison principale de cette limitation réside dans le type des contraintes utilisées. En effet, les contraintes implémentées pour un tétraèdre ne garantissent pas la conservation d'un volume strictement positif. Ainsi, lorsqu'un objet subit une grande déformation, certains de ses tétraèdres peuvent subir un aplatissement ou une inversion et peuvent même demeurer inversés le long de la simulation.

Une autre limitation existe au niveau de l'implémentation parallèle du solveur. Tout d'abord, l'implémentation parallèle utilise un solveur basé sur Jacobi qui souffre d'un taux de convergence très lent par rapport à Gauss-Seidel. En effet, dans un solveur de Jacobi, lorsqu'on itère sur les contraintes, le résultat des projections précédentes n'est pas instantanément pris en compte ce qui ralentit la convergence du solveur. D'autre part, l'implémentation basée sur le solveur de Jacobi ne profite pas des méthodes d'accélération numérique pour améliorer la convergence. Ceci impose donc la nécessité d'utiliser un grand nombre d'itérations pour assurer un comportement plausible durant la simulation.

Un autre problème souvent rencontré lors de la simulation d'une déformation des corps mous est lié à la qualité des tétraèdres inscrits à l'intérieur du maillage. Par exemple, le fait d'avoir des tétraèdres étirés peut entraîner des instabilités ou des artefacts lors de la simulation. À notre connaissance, il n'existe toujours pas une méthode qui permet de produire des maillages volumétriques de bonne qualité à partir de maillages de surface. Il devient donc difficile de simuler la déformation pour n'importe quel maillage.

CONCLUSION

Dans ce projet de recherche, notre objectif principal était de trouver une approche efficace qui permettait la simulation réaliste des corps déformables. En premier lieu, nous avons examiné des méthodes bien établies ayant des temps de calcul faibles et permettant d'obtenir les résultats souhaités. Après avoir évalué ces méthodes, nous avons pu mettre en place un modèle de simulation utilisant la dynamique basée sur les positions qui offre le meilleur compromis entre temps de calcul et réalisme. Ensuite, nous avons visé à améliorer la convergence du modèle implémenté en utilisant les méthodes d'accélération numérique. Pour cela, nous avons proposé d'utiliser à la fois l'accélération d'Anderson et la méthode de sur-relaxation successive. Les résultats présentés dans le chapitre 3, ont montré que la méthode proposée a bien amélioré le taux de convergence de la méthode de XPBD même avec des maillages possédant des tétraèdres de petits volumes plus difficiles à simuler. En outre, la méthode d'accélération proposée a l'avantage de ne pas dépendre du type des contraintes utilisées, ce qui la rend adaptée à n'importe quel système de simulation utilisant le solveur de XPBD. Toutefois, tel que mentionné dans le chapitre précédent, les contraintes utilisées ne gèrent pas l'inversion des tétraèdres qui peut se manifester lorsque l'objet subit une grande déformation. Une solution simple à ce problème consiste à rajouter des contraintes de volume qui permettent d'imposer un volume positif chaque fois les tétraèdres à l'intérieur de l'objet sont inversés. Une autre méthode qui est inspirée de (Tournier *et al.*, 2015) consiste à employer la Hessienne de la contrainte lors de la résolution du système de XPBD afin d'avoir une meilleur estimation de la force de contrainte. Ceci permet d'améliorer davantage la convergence du solveur et donc peut réduire le risque d'inversion des tétraèdres (Macklin *et al.*, 2019).

D'autre part, il semble intéressant d'examiner le gain de convergence de l'implémentation parallèle avec l'approche proposée. Nous prévoyons aussi d'utiliser la technique de coloration de graphe (Fratarcangeli & Pellacini, 2013) qui permettrait d'améliorer davantage le taux de convergence de la méthode de XPBD.

ANNEXE I

MODULES DE YOUNG DES MATÉRIAUX

Tableau-A I-1 Mesures du module de Young pour différents matériaux
Tiré de Cambridge (2003, p. 11)

Matériaux	Young Modulus (GPa)
Acier	189 - 210
Béton	10 - 50
Nylons	2.62 - 3.2
Élastomères de silicone	0.005-0.02
Caoutchouc	0.0015 - 0.0025
Néoprène	0.0007 - 0.002
Mousse polymère flexible	0.0003 - 0.001

Tableau-A I-2 Mesures d'indentation et de traction du module de Young pour les orgranes
Tiré de McKeee (2011, p. 4)

Tissue	Indentation (kPa)	Traction (MPa)
Peau	~ 85	~ 30
Moelle spinale	~ 3	~ 2
Muscle	~ 7	~ 480
Tendon	–	~ 560
Tissue de sein	~ 8	–
Sclère	–	~ 2.7

BIBLIOGRAPHIE

Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4), 547–560.

Baraff, D. & Witkin, A. (1998). Large Steps in Cloth Simulation. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, (SIGGRAPH '98), 43–54.

Bender, J., Koschier, D., Charrier, P. & Weber, D. (2014). Position-based simulation of continuous materials. *Computers & Graphics*, 44, 1–10.

Bouaziz, S., Martin, S., Liu, T., Kavan, L. & Pauly, M. (2014). Projective dynamics : fusing constraint projections for fast simulation. *ACM Transactions on Graphics (TOG)*, 33(4), 1–11.

Bridson, R., Fedkiw, R. & Anderson, J. (2002). Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, (SIGGRAPH '02), 594–603.

Cambridge. (2003). *Materials Data Book* (éd. Cambridge University Engineering Department). Cambridge University Engineering Department.

Diziol, R., Bender, J. & Bayer, D. (2009). Volume conserving simulation of deformable bodies.

Fratarcangeli, M. & Pellacini, F. (2013). A GPU-based implementation of position based dynamics for interactive deformable bodies. *Journal of Graphics Tools*, 17(3), 59–66.

Irving, G., Teran, J. & Fedkiw, R. (2004). Invertible finite elements for robust simulation of large deformation. *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 131–140.

Liu, T., Bargteil, A. W., O'Brien, J. F. & Kavan, L. (2013). Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)*, 32(6), 1–7.

Macklin, M. & Müller, M. (2013). Position based fluids. *ACM Transactions on Graphics (TOG)*, 32(4), 1–12.

Macklin, M., Müller, M., Chentanez, N. & Kim, T.-Y. (2014). Unified particle physics for real-time applications. *ACM Transactions on Graphics (TOG)*, 33(4), 1–12.

Macklin, M., Müller, M. & Chentanez, N. (2016). XPBD : position-based simulation of compliant constrained dynamics. *Proceedings of the 9th International Conference on Motion in Games*, 49–54.

Macklin, M., Erleben, K., Müller, M., Chentanez, N., Jeschke, S. & Makoviychuk, V. (2019). Non-smooth newton methods for deformable multi-body dynamics. *ACM Transactions on Graphics (TOG)*, 38(5), 1–20.

McKee, C. T., Last, J. A., Russell, P. & Murphy, C. J. (2011). Indentation versus tensile measurements of Young's modulus for soft biological tissues. *Tissue Engineering Part B : Reviews*, 17(3), 155–164.

Müller, M., Heidelberger, B., Hennix, M. & Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication and Image Representation*, 18(2), 109–118.

Peng, Y., Deng, B., Zhang, J., Geng, F., Qin, W. & Liu, L. (2018). Anderson acceleration for geometry optimization and physics simulation. *ACM Transactions on Graphics (TOG)*, 37(4), 1–14.

Servin, M., Lacoursiere, C. & Melin, N. (2006). Interactive simulation of elastic deformable materials. *SIGRAD 2006. The Annual SIGRAD Conference ; Special Theme : Computer Games*, (019).

Sifakis, E. & Barbic, J. (2012). FEM simulation of 3D deformable solids : a practitioner's guide to theory, discretization and model reduction. Dans *ACM SIGGRAPH 2012 courses* (1–50).

Sin, F. S., Schroeder, D. & Barbič, J. (2013). Vega : non-linear FEM deformable object simulator. *Computer Graphics Forum*, 32(1), 36–48.

Tournier, M., Nesme, M., Gilles, B. & Faure, F. (2015). Stable constrained dynamics. *ACM Transactions on Graphics (TOG)*, 34(4), 1–10.

Walker, H. F. (2011). Anderson acceleration : Algorithms and implementations. *WPI Math. Sciences Dept. Report MS-6-15-50*.

Wang, H. (2015). A chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Transactions on Graphics (TOG)*, 34(6), 1–9.

Yang, C., Meza, J. C., Lee, B. & Wang, L.-W. (2009). KSSOLV—a MATLAB toolbox for solving the Kohn-Sham equations. *ACM Transactions on Mathematical Software (TOMS)*, 36(2), 1–35.

Zhang, J., Zhong, Y. & Gu, C. (2017). Deformable models for surgical simulation : a survey. *IEEE reviews in biomedical engineering*, 11, 143–164.

Zhang, J., Peng, Y., Ouyang, W. & Deng, B. (2019). Accelerating ADMM for efficient simulation and optimization. *ACM Transactions on Graphics (TOG)*, 38(6), 1–21.