

Animation efficace de solides en contact par modèle physique

Galizzi Olivier, Faure François

iMAGIS-GRAVIR/IMAG (CNRS,INRIA,INPG,UJF) 655 av. de l'Europe, 38330 Montbonnot

Olivier.Galizzi@imag.fr

Résumé : *Dans cet article, nous proposons une nouvelle approche au problème du calcul de réponses aux collisions permettant de manipuler interactivement plusieurs centaines de solides en contact. La méthode proposée est de type correctrice, en ce sens qu'elle intervient après le calcul des nouveaux états des solides selon les lois de Newton sans tenir compte des collisions. Des contraintes sont ensuite appliquées, permettant de supprimer les interpénétrations et de corriger vitesses et accélérations des solides. Les trois corrections se font de façon itérative à l'aide d'un nouvel algorithme dérivé des algorithmes d'optimisation de la famille des gradients conjugués. Un tel algorithme de résolution nous permet de plus de régler un compromis entre précision et rapidité des calculs et ainsi de réaliser des animations peu précises mais rapides ou au contraire plus exactes mais également plus lentes.*

Mots-clés : simulation solides contacts

1 Introduction

La simulation de solides par modèle physique couvre un vaste champ d'application allant des effets spéciaux cinématographiques, aux programmes de jeux vidéos en passant par toutes sortes de simulations de type éboulement rocheux ou destruction d'empilements. Les scènes de la vie quotidienne sont également composées, pour la plupart, d'un grand nombre d'objets solides, articulés ou non et soumis aux lois de la gravité. Les méthodes d'animation par modèle physique permettent la conception d'animation réalistes de ce type de scènes, difficiles à générer par la main d'un animateur. C'est pourquoi ce domaine a été déjà largement exploré durant les quinze dernières années. Cependant les méthodes actuelles de simulation de solides par modèle physique se heurtent à la complexité des algorithmes utilisés (en $O(n^3)$ voire $O(n^4)$ où n est le nombre de solides en jeu dans la simulation). Leurs performances deviennent donc catastrophiques dès lors que ce nombre s'accroît de trop, et elles deviennent alors inutilisables pour des applications temps réel. Une nouvelle approche est donc nécessaire afin de palier à ce problème qu'est la trop grande complexité des scènes, avec pour objectif, de réaliser des simulations perceptuellement convaincantes, c'est à dire où les trajectoires des solides sont réalistes à l'œil même si elles ne sont pas physiquement parlant exactes.

Les problèmes majeurs auxquels se sont heurtés toute une génération de chercheur sont principalement la stabilité des simulations, notamment aux empilements de solides, mais également la lenteur des algorithmes utilisés étant donné leur complexité. Cependant de grandes avancées ont déjà été faites depuis les premières méthodes dites de pénalité présentées en 1988 par Moore et Wilhelms dans [?] où des ressorts de longueur à vide nulle étaient placés entre deux solides en collisions permettant ainsi de les faire ressortir de cet état incohérent. Les plus grandes avancées restent pourtant récentes. Ainsi c'est en 2000 seulement dans [?] que Mirtich fut capable de gérer un grand nombre de solides en un temps raisonnable mais il supposait qu'ils étaient relativement bien espacés et répartis dans l'espace. En 2001, dans [?] Milenkovic proposa, lui, de synchroniser toutes les collisions ainsi que leur traitement à la fin de chaque pas de temps. Tous les états des solides en collisions étaient alors corrigés en même temps à l'aide d'algorithmes d'optimisation minimisant une énergie cinétique. Pour la première fois quasiment, une solution très stable au problème de l'animation de solides par modèle physique fut proposée. Ainsi Milenkovic fut capable d'empiler dix cubes les uns sur les autres. Cependant la méthode utilisée restait encore relativement lente et non interactive (1 image par seconde).

Stabilité ou rapidité personne n'a encore été capable de concilier ces deux aspects importants de la simulation de solides par modèle physique, d'où notre travail sur ces deux points essentiels.

2 Prérequis

2.1 Notations

Voici les quelques conventions et notations adoptées pour la rédaction de cet article :

- les lettres en caractères gras majuscules représentent des matrices
- les lettres en caractères gras minuscules représentent des vecteurs
- les lettres en caractères standard minuscules représentent des scalaires

De plus on pose $\mathbf{0}$ comme étant le vecteur ou la matrice nul de la taille approprié.

Voici également les conventions utilisées pour manipuler les différents composants des solides notés S_i :

- le centre de gravité est noté o_i
- les points situés sur la surface du solide sont notés $p_1, p_2 \dots, p_n$
- les coordonnées, la vitesse et l'accélération d'un point p_i sont notées respectivement $\mathbf{p}_i, \dot{\mathbf{p}}_i, \ddot{\mathbf{p}}_i$

2.2 La mécanique du solide

L'animation par modèle physique de solides peut être mathématiquement formulée comme étant l'intégration sur le temps des équations différentielles suivantes :

$$\dot{\mathbf{o}}_i = \frac{do_i}{dt} \quad , \quad \ddot{\mathbf{o}}_i = \frac{d\dot{o}_i}{dt} \quad (2.1)$$

$$m\ddot{\mathbf{x}} = \sum \mathbf{f}_i \quad , \quad \mathbf{I}_M \dot{\boldsymbol{\omega}} = \sum \mathbf{c}_i \quad (2.2)$$

Les équations 2.1 traduisent la relation entre positions et vitesses ainsi qu'entre vitesses et accélération des solides. Les relations 2.2 traduisent les lois de Newton-Euler, où m est la masse du solide considéré, $\sum \mathbf{f}_i$ la somme des forces exercées sur le solide (gravité, forces de contraintes, forces élastiques ou visqueuses ...), \mathbf{I}_M la matrice d'inertie du solide et $\sum \mathbf{c}_i$ la somme des couples appliqués au solide.

Nous modélisons la vitesse d'un solide au sein de son repère local par le vecteur de dimension six noté $\dot{\mathbf{x}}_i = [\mathbf{o}_i^T \quad \omega_i^T]^T$ ou \mathbf{o}_i dénote la vitesse à l'origine et ω_i la vitesse angulaire du solide S_i . De même l'accélération peut être exprimée comme le vecteur $\ddot{\mathbf{x}}_i = [\dot{\mathbf{o}}_i^T \quad \dot{\omega}_i^T]$. La vitesse et l'accélération d'un point p_i lié au solide S_i s'expriment comme le montrent les relations 2.3 et 2.4.

$$\dot{\mathbf{p}}_1 = \dot{\mathbf{o}}_1 + \omega_1 \times \mathbf{o}_1 \mathbf{p}_1 \quad (2.3)$$

$$\ddot{\mathbf{p}}_1 = \ddot{\mathbf{o}}_1 + \dot{\omega}_1 \times \mathbf{o}_1 \mathbf{p}_1 + \omega_1 \times (\omega_1 \times \mathbf{o}_1 \mathbf{p}_1) \quad (2.4)$$

$$\mathbf{n} \cdot \ddot{\mathbf{p}}_1 = \mathbf{n} \cdot \ddot{\mathbf{o}}_1 + (\mathbf{o}_1 \mathbf{p}_1 \times \mathbf{n}) \cdot \dot{\omega}_1 + \mathbf{n} \cdot \omega_1 \times (\omega_1 \times \mathbf{o}_1 \mathbf{p}_1) \quad (2.5)$$

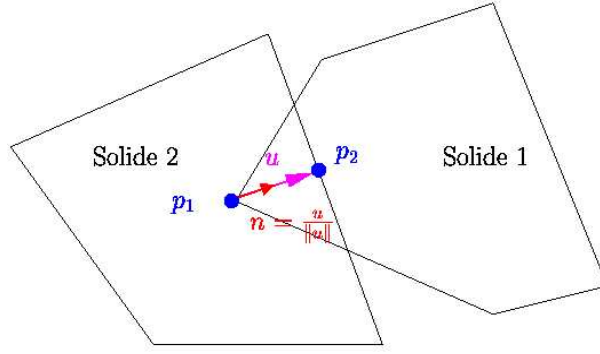
$$\mathbf{n} \cdot (\ddot{\mathbf{p}}_1 - \ddot{\mathbf{p}}_2) = \mathbf{j}_1 \ddot{\mathbf{x}}_1 - \mathbf{j}_2 \ddot{\mathbf{x}}_2 + \mathbf{c}_1 - \mathbf{c}_2 \quad (2.6)$$

L'accélération du point p_i projetée sur une direction de contrainte \mathbf{n} (voir équation 2.5) peut également s'écrire $\mathbf{n} \ddot{\mathbf{p}}_i = \mathbf{j}_i \ddot{\mathbf{x}}_i + \mathbf{c}_i$ ou $\mathbf{j}_i = [\mathbf{n}^T \quad (\mathbf{o}_i \mathbf{p}_i \times \mathbf{n})^T]$. Dans le cas de solides en collisions, cette direction \mathbf{n} et celle portée par le vecteur d'extraction modélisé lors de la détection de collisions (voir figure 1). À partir de cette formulation, il est facile de trouver l'accélération de pénétration des deux solides S_1 et S_2 aux points p_1 et p_2 comme le montre la relation 2.6.

3 Formalisation des contraintes

3.1 Ecriture des contraintes

Comme mentionné dans l'introduction, nous utilisons des contraintes afin de corriger les états de solides et remettre le système dans un état cohérent. Dans cette section, nous montrons comment poser les contraintes pour la



Deux solides en collision

FIG. 1 – Modélisation d'une collision. Dû à l'intégration discrète du temps, les solides ne sont pas en contact mais en interpénétration. Le vecteur u tel qu'il est représenté est appelé vecteur d'extraction.

correction des accélérations des solides. L'objectif est d'annuler l'accélération de pénétration entre deux solides S_1 et S_2 en collision. Pour cela on va donc chercher, en partant des valeurs de $\ddot{\mathbf{p}}_1$ et $\ddot{\mathbf{p}}_2$, les valeurs corrigées $\ddot{\mathbf{p}}_{1_{\text{corr}}}$ et $\ddot{\mathbf{p}}_{2_{\text{corr}}}$ satisfaisant la contrainte 3.1.

$$\mathbf{n}(\ddot{\mathbf{p}}_{1_{\text{corr}}} - \ddot{\mathbf{p}}_{2_{\text{corr}}}) = 0 \quad (3.1)$$

On va donc chercher les $\Delta\ddot{\mathbf{p}}_1$ et $\Delta\ddot{\mathbf{p}}_2$ tels que en posant :

$$\begin{aligned} \ddot{\mathbf{p}}_{1_{\text{corr}}} &= \ddot{\mathbf{p}}_1 - \Delta\ddot{\mathbf{p}}_1 \\ \ddot{\mathbf{p}}_{2_{\text{corr}}} &= \ddot{\mathbf{p}}_2 - \Delta\ddot{\mathbf{p}}_2 \end{aligned}$$

on ait la condition 3.1 à vrai. Pour cela, on veut donc que :

$$\mathbf{n}(\ddot{\mathbf{p}}_{1_{\text{corr}}} - \ddot{\mathbf{p}}_{2_{\text{corr}}}) = \mathbf{n}(\ddot{\mathbf{p}}_1 - \Delta\ddot{\mathbf{p}}_1 - \ddot{\mathbf{p}}_2 + \Delta\ddot{\mathbf{p}}_2) = 0$$

C'est à dire que :

$$\mathbf{n}(\Delta\ddot{\mathbf{p}}_1 - \Delta\ddot{\mathbf{p}}_2) = \mathbf{n}(\ddot{\mathbf{p}}_1 - \ddot{\mathbf{p}}_2)$$

Soit en utilisant la relation 2.6 :

$$\mathbf{j}_1\ddot{\mathbf{x}}_1 + \mathbf{c}_1 - \mathbf{j}_2\ddot{\mathbf{x}}_2 - \mathbf{c}_2 = \mathbf{j}_1\Delta\ddot{\mathbf{x}}_1 - \mathbf{j}_2\Delta\ddot{\mathbf{x}}_2$$

En remarquant que : $\ddot{\mathbf{x}}_1 = \mathbf{M}_1^{-1}\mathbf{f}_{\text{ext}}$ et $\ddot{\mathbf{x}}_2 = \mathbf{M}_2^{-1}\mathbf{f}_{\text{ext}}$ Et en notant :

$$\begin{aligned} M &= \begin{bmatrix} \mathbf{M}_1 & 0 \\ 0 & \mathbf{M}_2 \end{bmatrix} \\ J &= [\mathbf{j}_1 \quad -\mathbf{j}_2] \\ \Delta\ddot{\mathbf{x}} &= \begin{bmatrix} \Delta\ddot{\mathbf{x}}_1 \\ \Delta\ddot{\mathbf{x}}_2 \end{bmatrix} \\ c &= \mathbf{c}_1 - \mathbf{c}_2 \end{aligned}$$

On peut écrire :

$$\mathbf{J}\mathbf{M}^{-1}\mathbf{f}_{\text{ext}} + \mathbf{c} = \mathbf{J}\Delta\ddot{\mathbf{x}} \quad (3.2)$$

Or on sait que :

$$\Delta\ddot{\mathbf{x}}_1 = \underbrace{\mathbf{M}_i^{-1} \underbrace{\mathbf{J}_i^T f_{12}}_{\text{force } f_{12} \text{ exprimée en } o_1}}_{\text{variation d'accélération induite en } o_1 \text{ par } f_{12}} \quad (3.3)$$

Et de même pour $\Delta\ddot{\mathbf{x}}_2$, où f_{12} (respectivement $-f_{12}$) est une force appliquée au point p_1 (respectivement p_2) selon une direction qui est la direction n de pénétration.

Le problème n'est plus alors de trouver $\Delta\ddot{\mathbf{p}}_1$ et $\Delta\ddot{\mathbf{p}}_2$ mais le scalaire f_{12} vérifiant le système d'équation suivant :

$$\begin{aligned} \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_{12} &= -\mathbf{J}\mathbf{M}^{-1}\mathbf{F}_{\text{ext}} + \mathbf{c} \\ &= -\text{accélération de pénétration de } p_1 \text{ et } p_2 \\ &= -\text{erreur à corriger} \end{aligned} \quad (3.4)$$

On peut noter que la force f_{12} ne travaille pas, c'est à dire qu'elle ne génère pas de mouvement. Elle est là, uniquement pour garantir la contrainte posée (i.e. annuler l'accélération de pénétration) : elle correspond, en fait, au multiplicateur de lagrange de cette même contrainte.

3.2 Cas général

Dans le cas général de n solides en jeu dans la simulation, les matrices \mathbf{J} et \mathbf{M} sont de la forme 3.5 et 3.6.

$$\mathbf{J} = \begin{bmatrix} \bullet & \bullet & \mathbf{j}_i & \bullet & -\mathbf{j}_k & \bullet \\ \bullet & \mathbf{j}_l & \bullet & \bullet & -\mathbf{j}_m & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \mathbf{j}_n & \bullet & -\mathbf{j}_o \\ \mathbf{j}_p & \bullet & \bullet & -\mathbf{j}_p & \bullet & \bullet \end{bmatrix} \quad (3.5)$$

$$\mathbf{M} = \text{diag}(M_1, M_2, \dots, M_n) \quad (3.6)$$

La matrice \mathbf{M} est de taille $6s \times 6s$ ou s est le nombre de solides, et \mathbf{J} a pour taille $c \times 6s$ ou c est le nombre de contraintes. Les blocs nuls sont représentés par des \bullet . Sur chaque ligne de \mathbf{J} seulement \mathbf{j}_i et \mathbf{j}_k sont non nuls lorsque les solides S_i et S_k ont un point de collision détecté. Chaque ligne correspond donc à une contrainte et les blocs non nuls aux solides contraints.

Dans le cas général la correction des accélérations revient à résoudre un système linéaire de la forme $\mathbf{A}\mathbf{f} = \mathbf{b}$ ou $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ et \mathbf{b} égal à $\mathbf{J}\mathbf{M}^{-1}\mathbf{F}_{\text{ext}} + \mathbf{c}$. Le vecteur \mathbf{f} est composé de forces exprimées en $kg.m.s^{-2}$ et correspond au vecteur des multiplicateurs de lagrange associés aux contraintes (de façon similaire à la méthode de Baraff exposée dans [?] mais appliqué ici au cas des solides en contacts et non pas seulement au cas des solides articulés).

4 Résolution de $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} \geq \mathbf{b}$

4.1 Gradient bi-conjugué modifié

La matrice $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ de notre système est relativement large. Elle se compose en effet, d'autant de colonnes qu'il y a de solides en jeu dans la simulation, et, d'autant de ligne qu'il y a de contraintes, donc de collisions. Ses dimensions exèdent donc souvent plusieurs centaines de lignes et colonnes. Pour être à même de résoudre un tel système plusieurs fois par seconde afin d'atteindre notre objectif de temps réel, une approche classique est impossible. Nous avons donc développé un nouvel algorithme basé sur l'algorithme d'optimisation du gradient bi-conjugué (voir [?] chapitre *Sparse Linear Systems* pages 83 à 89) qui résoud un système d'équations linéaires $\mathbf{A}\mathbf{f} + \mathbf{b} = \mathbf{0}$ par minimisation itérative de $(\mathbf{A}\mathbf{f} + \mathbf{b})^2$. Cet algorithme itératif effectue comme seul calcul coûteux, deux produits matrices vecteurs par itérations, ce qui nous permet d'exploiter au mieux le fait que \mathbf{J} et \mathbf{M}^{-1} sont creuses alors que \mathbf{A} ne l'est pas. Nous effectuons pour cela, le produit $\mathbf{A}\mathbf{f}$ en trois étapes $O(n)$ en ne calculant jamais explicitement \mathbf{A} .

Cependant des conditions supplémentaires doivent être ajoutées au système afin de prendre en compte du sens physique des grandeurs manipulées (d'où l'inégalité à la place du signe égal dans le titre de la section). Considérons un ensemble fini de collisions entre deux solides S_1 et S_2 , modélisées chacune par deux points \mathbf{p}_{11} et \mathbf{p}_{12} et un vecteur d'extraction $\mathbf{n}_1 = \mathbf{p}_{11} - \mathbf{p}_{12}$ donnant une direction normale au contact et selon lequel le mouvement est

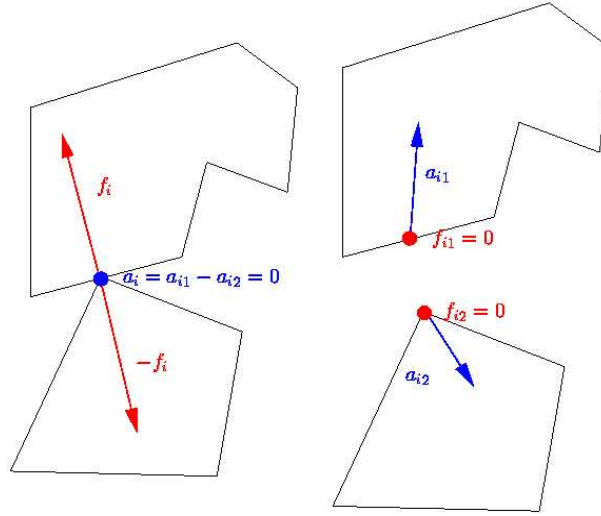


FIG. 2 – Dans le cas sans frottement deux cas de figures exclusifs sont possible. S_1 et S_2 ne peuvent pas se rapprocher ($a_i < 0$) mais sont libres de s'éloigner l'un de l'autre ($a_i \geq 0$). D'un autre coté, les forces de contacts f_i sont toujours répulsives (par convention on dira $f_i \geq 0$). Bien évidemment, soit les deux corps sont dans un état de contact actif et leurs accélération de pénétration est nulle mais il existe une force répulsive donc positive qui maintient le contact et empêche une plus profonde pénétration (à gauche), soit les deux corps ne sont plus en contact et aucune force n'agit entre eux mais leur accélération relative est positive et non nulle (à droite).

contraint. Soit $a_i = \mathbf{n}_i \cdot (\ddot{\mathbf{p}}_{i_1} - \ddot{\mathbf{p}}_{i_2})$ l'accélération de pénétration de S_1 et S_2 et f_i la force de contact agissant entre ces mêmes corps selon l'axe \mathbf{n}_i (voir figure 1).

Pour chaque collision i , les trois contraintes expliquées figure 2 se résument respectivement par les trois conditions suivantes :

$$a_i \geq 0 \quad f_i \geq 0 \quad f_i a_i = 0$$

Or, puisque a_i et f_i sont contraintes positives pour tout i , on a :

$$f_i a_i = 0 \Leftrightarrow \mathbf{f}^T \mathbf{a} = 0$$

Le problème du calcul des forces de contact peut donc se ramener à la formulation suivante, connue sous le nom de problème linéaire complémentaire (LCP) :

$$\begin{cases} \mathbf{A}\mathbf{f} + \mathbf{b} & \geq 0 \\ \mathbf{f} & \geq 0 \\ \mathbf{f}^T(\mathbf{A}\mathbf{f} + \mathbf{b}) & = 0 \end{cases} \quad (4.1)$$

C'est lors de la résolution de ce LCP qu'entre en jeu notre nouvel algorithme de type gradient bi-conjugué, qui peut prendre en compte ces nouvelles conditions. Nous utilisons pour cela la notion d'ensemble actif. Considérons un ensemble de n collisions. Nous dirons pour chaque contact qu'il est soit *actif* et $f_i \geq 0$, $a_i = 0$ soit *inactif* et $a_i \geq 0$, $f_i = 0$ selon les conditions qu'il vérifie. Nous construisons donc deux partitions, nommées A pour *actif* et IA pour *inactif*. Au début de chaque résolution, nous avons choisi pour heuristique de mettre tous les contact dans la classe A . Les itérations du gradient, ne sont maintenant effectuées que sur les équations appartenant à la classe A . Ces deux classes peuvent cependant évoluer lors de la résolution du système, donc au fil des itérations. En effet, comme le montre la figure 3, l'algorithme de gradient cherche à faire le meilleur compromis, afin de satisfaire au mieux toutes les équations. Cependant, il n'est pas toujours possible de les satisfaire toutes en même temps. A chaque itération, et pour chaque contact, nous appliquons donc l'automate représenté figure 4. Ainsi si pour chaque contact de la classe A si la force de contact associée est devenue attractive ($f_i < 0$), nous passons ce contact dans la classe IA et nous n'en tenons plus compte. Au contraire, si pour un contact de la classe IA , nous détectons que l'accélération de pénétration devient négative ($a_i < 0$), alors nous repassons ce contact dans la classe A . Bien évidemment, pour chaque itération où il y a eu un changement de classe, l'algorithme de gradient doit être

redémarré, car la dimension du système a changé par l'ajout ou la suppression d'une ou plusieurs contraintes (i.e. d'équations). Comme solution initiale du nouveau système, nous utilisons alors simplement la valeur calculée lors de la dernière itération.

L'avantage de notre méthode est qu'elle intervient au coeur même de l'algorithme du gradient. Nous construisons en effet, les partitions A et IA à la volée et n'avons ainsi pas besoin de résoudre un système d'équations linéaires entier à chaque itération pour les mettre à jour, comme cela a été le cas jusqu'à présent (voir algorithmes ci-dessous).

Jusqu'à présent : résolution d'un système entier avant de pouvoir mettre à jour les partitions A et IA . chaque résolution de $\mathbf{A}\mathbf{f} + \mathbf{b} = \mathbf{0}$ est en $O(n^2)$ et l'on fait au minimum $O(n)$ itérations d'où un algorithme au mieux en $O(n^3)$.

Procédure résoudreSysteme1
 initialisation
 $\mathbf{f}_{old} = \mathbf{0}$
Tantque pas resolu **faire**
 $\mathbf{f}_{new} = \text{resoudre } \mathbf{A}\mathbf{f} + \mathbf{b} = \mathbf{0} \text{ sur } A \quad (\geq O(n^3))$
 $\mathbf{f}_{new} = \mathbf{f}_{old} + \alpha(\mathbf{f}_{new} - \mathbf{f}_{old})$
 $(A, IA) = \text{mise à jour de } (A, IA)$

Où α est le plus grand pas qu'il est possible d'effectuer tout en respectant les inégalités.

Notre algorithme : Mise à jour de A et IA à chaque itération du gradient bi-conjugué. À chaque itération (1 itérations = chercher direction + longueur pas) on met à jour les partitions. On effectue donc $O(n)$ itérations en $O(n)$ d'où un algorithme en $O(n^2)$ au maximum.

Procédure résoudreSysteme2
 initialisation
 $\mathbf{f}_{old} = \mathbf{0}$
Tantque pas resolu **faire**
 $\mathbf{d} = \text{nouvelle direction de recherche} \quad (O(n))$
 $\beta = \text{longueur du pas à effectuer}$
 $\mathbf{f}_{new} = \mathbf{f}_{old} + \beta\mathbf{d}$
 $(A_{new}, IA_{new}) = \text{mise à jour de } (A, IA)$
 Si $A_{new} \neq A$ **alors**
 réinitialisation
 $(A, IA) = (A_{new}, IA_{new})$

Où β est la longueur du pas standard du gradient bi-conjugué.

5 Performance et convergence

5.1 Convergence

Etant donné que les forces solutions du système $\mathbf{A}\mathbf{f} = \mathbf{b}$ correspondent aux multiplicateurs de Lagrange associés aux contraintes, l'algorithme converge vers une solution unique si et seulement si la matrice $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ est définie positive. Ce n'est pas toujours le cas, notamment lorsque se créent des cycles (par exemple une chaîne fermée de solides articulés), pourtant nous n'avons pas remarqué de problèmes liés à cette non unicité de la solution dans les scènes que nous avons manipulées.

Quoi qu'il en soit, et c'est le point principal, notre algorithme nécessite très peu d'itérations pour converger vers un résultat visuellement correct. En effet, un algorithme de type gradient conjugué trouve la solution exacte d'un système en autant d'itérations qu'il y a d'inconnues. Or dans notre cas, pour un système contenant près de 300 inconnues (i.e. 300 collisions), une dizaine d'itérations peuvent suffire pour obtenir un résultat très correct. Chaque

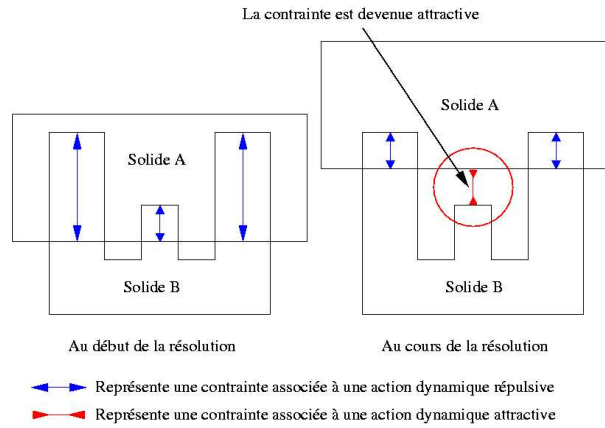


FIG. 3 – L’algorithme de gradient conjugué cherche un meilleur compromis satisfaisant au mieux toutes les conditions. Dans cet exemple au cours de la résolution une force deviendra inévitablement attractive : notre algorithme va donc le transférer dans la classe IA .

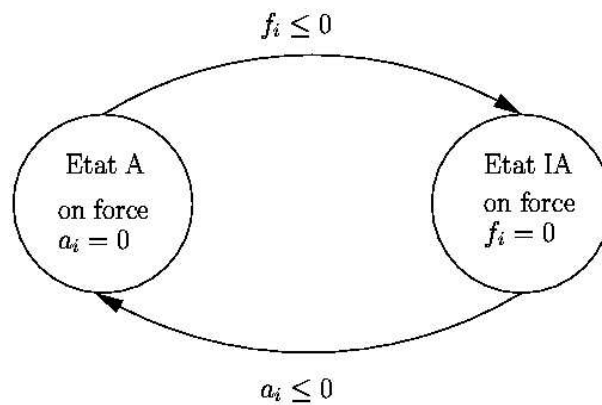


FIG. 4 – Automate de transition entre les classes A et IA

itération s’effectuant en temps linéaire, notre algorithme qui a une complexité au pire en $O(n^2)$ voit donc cette complexité descendre très près de $O(n)$.

5.2 Performance

Voici quelques résultats de performance que nous avons obtenus pour différents types de simulation (des captures d’écran des simulations correspondantes sont fournies en annexe A) : on notera que les solides sont constitués d’assemblages de sphères pour simplifier la détection de collisions, ce qui n’affecte en rien la validité de notre méthode étant donné que la dynamique d’un solide ne dépend pas de sa géométrie.

6 Application à la correction des positions et vitesses

Afin de calculer la correction à effectuer sur les accélérations des solides nous avons procédé comme suit :

- \mathbf{f} = forces f_i à appliquer au niveau de chaque collision selon l’axe de contrainte
- $\mathbf{J}^T \mathbf{f}$ = forces f_i exprimées au centre de gravité o_i des solides
- $\mathbf{M}^{-1} \mathbf{J}^T \mathbf{f}$ = variations d’accélérations linéaires et angulaires induites en o_i par les f_i
- $\mathbf{JM}^{-1} \mathbf{J}^T \mathbf{f}$ = variations d’accélérations de pénétration induites aux points de collisions par les f_i

	# solides - sphères	# collisions	fps
Pendule	11 - 61	10	300
125 sphères	126 - 126	250	45
216 sphères	217 - 217	400	20
343 sphères	344 - 344	700	10
Chaîne	13 - 361	30	50
Tourniquet	126 - 602	700	10
Pile	14 - 405	120	30

TAB. 1 – Complexité de quelques scènes et efficacité de la méthode (captures d'écran sont disponibles annexe A).

Et l'on résoud le système suivant :

$$\mathbf{JM}^{-1}\mathbf{J}^T\mathbf{f} + \mathbf{c} = -\text{erreur sur les accélérations}$$

Pour la correction des vitesses, on a montré que l'on peut procéder de même, en résolvant le système suivant :

$$\begin{aligned} \mathbf{JM}^{-1}\mathbf{J}^T\boldsymbol{\pi} + \mathbf{c} &= \mathbf{J}\dot{\mathbf{x}} \\ &= -\text{erreur sur les vitesses} \end{aligned} \quad (6.1)$$

où $\boldsymbol{\pi}$ est un vecteur d'impulsions ($kg.m.s^{-1}$) et va permettre d'appliquer une variation instantanée de vitesse sur les solides selon le même schéma que précédemment :

$$\begin{aligned} \boldsymbol{\pi} &= \text{impulsions } \pi_i \text{ à appliquer au niveau de chaque collision selon l'axe de contrainte} \\ \mathbf{J}^T\boldsymbol{\pi} &= \text{impulsions } \pi_i \text{ exprimées au centre de gravité } o_i \text{ des solides} \\ \mathbf{M}^{-1}\mathbf{J}^T\boldsymbol{\pi} &= \text{variations de vitesses linéaires et angulaires induites en } o_i \text{ par les } \pi_i \\ \mathbf{JM}^{-1}\mathbf{J}^T\boldsymbol{\pi} &= \text{variations de vitesse de pénétration induites aux points de collisions pas les } \pi_i \end{aligned}$$

Pour ce qui est de la correction des positions, les contraintes sont maintenant géométriques et l'on va travailler sur des approximations. En effet, afin de pouvoir corriger l'orientation et la positions des solides il est nécessaire de linéariser les rotations. Nous manipulons donc, dans ce dernier cas, un vecteur $\boldsymbol{\delta}$ d'action de déplacement ($kg.m$) qui vont induire des déplacements instantanés sur les solides afin de les faire ressortir de leur état d'interpénétration. Ceci peut se formuler comme suit :

$$\begin{aligned} \mathbf{JM}^{-1}\mathbf{J}^T\boldsymbol{\delta} &= \mathbf{n}(\mathbf{p}_1 - \mathbf{p}_2) \\ &= -\text{profondeurs pénétrations} \end{aligned} \quad (6.2)$$

On remarquera que la correction que nous apportons sur les positions, vitesses et accélérations, se calcule de manière analogue et résolvant un système d'équations linéaires du type

$$\mathbf{JM}^{-1}\mathbf{J}^T\boldsymbol{\lambda} = -\text{erreur}$$

où $\boldsymbol{\lambda}$ est respectivement un vecteur d'actions de déplacement, un vecteur d'impulsions et un vecteur d'accélérations. Nous avons donc un seul et même algorithme pour les trois corrections.

La gestion du rebond se fait via le modèle de Poisson, simplement en remplaçant le vecteur $\boldsymbol{\pi}$ calculé, par le vecteur $(1 + \epsilon)\boldsymbol{\pi}$, où $0 \leq \epsilon \leq 1$ est le coefficient de rebond.

7 Extension aux solides articulés

L'extension du simulateur à la gestion des solides articulés ne pose pas de problème particulier avec notre méthode. Par exemple, une liaison de type point-point se modélise avec trois contraintes scalaires dont chaque direction est

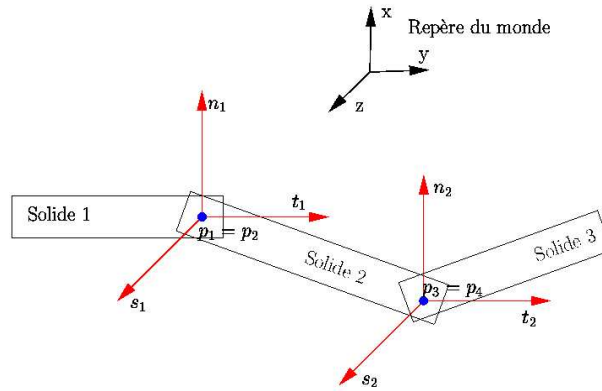


FIG. 5 – Les repères locaux des contraintes point sur point sont alignés avec les axes du monde. Les points p_1 et p_2 appartenant respectivement aux solides 1 et 2 doivent être maintenus confondus et leur vitesse ainsi que leur accélération relative doivent être nulles. Idem pour les points p_3 et p_4 .

alignée sur un des trois axes principaux \mathbf{n} , \mathbf{t} et \mathbf{s} d'un repère orthogonal (voir figure 5). Pour simplifier les calculs le repère choisi est tout simplement le repère du monde.

La résolution de ces contraintes se fait en même temps que celles associées aux collisions et notre algorithme reste globalement inchangé, à la différence près que ces contraintes doivent toujours rester dans la classe *ACTIF*. En effet, une articulation n'a pas lieu de se désolidariser et donc les deux points p_1 et p_2 de se décoller.

8 Extension au frottement adhérent

Un exemple simple de frottement adhérent est celui d'une boule roulant sans glisser sur un plan : c'est une simplification du frottement de Coulomb où les objets peuvent glisser les uns sur les autres. Il suffit pour le prendre en compte dans notre méthode, de construire non plus une contrainte normale par collision mais trois : une normale et deux dans le plan tangent au contact. On procède ensuite de la même façon qu'auparavant mais avec trois fois plus de contraintes. On garantit ainsi, en plus du cas sans frottement que la vitesse et accélération de pénétration des deux solides en jeu est nulle non seulement dans la direction \mathbf{n} , mais aussi dans les deux directions tangentielles orthogonales \mathbf{t} et \mathbf{s} .

9 Répartition de la charge de calcul

Nous nous sommes posé la question de savoir quelle est la répartition optimale en nombre d'itérations sur les positions, vitesses et accélération, à distribuer, selon un budget de calcul exprimé lui aussi en nombre d'itérations. En partant du principe que notre critère de qualité définissant une bonne simulation est la distance de pénétration moyenne des solides après correction des positions, on va parcourir l'espace constitué des différentes répartitions et trouver le minimum de la fonction ainsi parcourue. L'espace des paramètres de réglage a trois dimensions : nombres d'itérations sur les positions, sur les vitesses et sur les accélérations mais leur somme étant constante le domaine de définition de notre fonction est sur deux dimensions et a une forme triangulaire comme le montre la figure 6. La figure 7, montre les résultats obtenus pour la répartition optimale de 30 itérations pour le *tourniquet* (voir annexe A). On remarque que si on ne corrige pas du tout les positions ou les vitesses, l'erreur diverge. Par contre on remarque également que trop corriger les positions (voir zone 1) ne garantit pas forcément un bon résultat car les erreurs engendrées sur les vitesses deviennent tellement grandes qu'elles se répercutent irrémédiablement sur la correction des positions. Tout miser sur la correction des accélérations (voir zone 2) engendre le même problème. La vallée traversant le centre du triangle contient des répartitions beaucoup plus satisfaisantes avec un point optimal (en magenta) correspondant à la répartition optimale pour cet exemple.

Cette rapide étude de la répartition montre qu'il est nécessaire de corriger un minimum les positions afin essentiellement de supprimer les dérives dues aux erreurs numériques et à l'intégration discrète du temps. Cependant, l'essentiel des calculs doit être concentré sur la correction des vitesses.

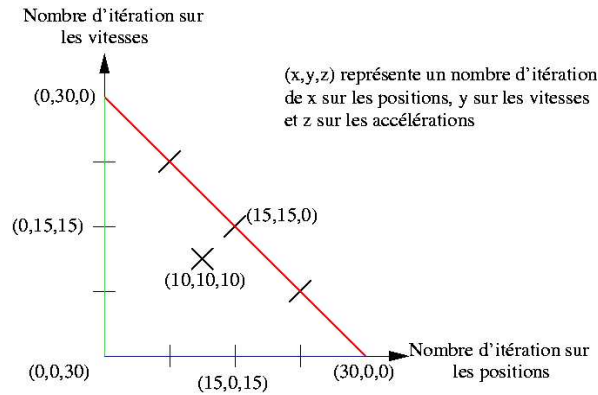


FIG. 6 – Domaine de définition de la fonction représentant l'erreur sur les positions en fonction du nombre d'itérations attribuées pour la correction des positions, des vitesses et des accélérations

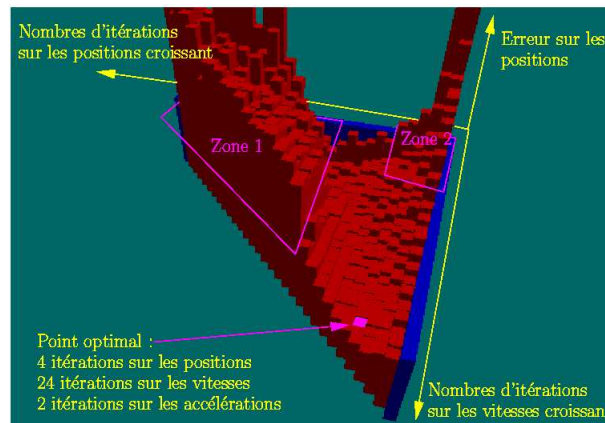


FIG. 7 – Graphique montrant l'erreur commise sur la correction de position en fonction de la répartition de 30 itérations. En bleu on peut voir l'ensemble des paramètres faisant diverger le programme et en magenta le point optimal

10 Conclusion

L'algorithme présenté ici fonctionne très bien pour de nombreux types de simulations et peut être utilisé pour des applications diverses et variées, comme la simulation d'éboulis ou de chute de pierre, pour les jeux de constructions ou encore les scènes "alimentaires" (objets en contact dans un bol...). La rapidité du calcul de réponse aux collisions en fait un outil parfait pour la simulation temps réel d'un grand nombre de corps (plusieurs centaines) avec un grand nombre de collisions (plusieurs centaines également). Le système de compromis entre efficacité et rapidité de calcul accroît cette possibilité et l'étend même à la simulation non temps réel d'un très grand nombre de corps (plusieurs milliers) facilité par une complexité en mémoire linéaire (car les matrices sont creuses et sont donc stockées sous forme de graphe).

Les tests montrent que la répartition du temps de calcul optimal est relativement non conventionnelle et nécessite des corrections en position ce qui est très rarement fait dans les méthodes de simulation de solides par modèle physique. Cependant, et cela confirme la tendance dans ce domaine, la correction sur les vitesses est prépondérante d'autant plus que c'est elle qui donne son réalisme à la simulation.

11 Travaux futurs

Trois principaux objectifs restent à atteindre pour compléter notre méthode.

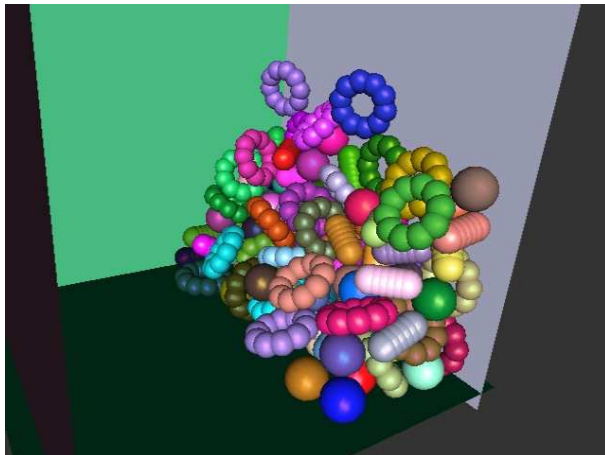
Tout d'abord il serait important d'avoir une implémentation du frottement gérant le modèle de Coulomb qui est un

modèle plus réaliste que celui implémenté ici. Pour cela il sera nécessaire de discrétiser le cône de Coulomb en une pyramide à base polygonale pour pouvoir appliquer des contraintes linéaires sur les composantes normales et tangentielles des forces. Cela conduira bien entendu à un frottement anisotropique. Des travaux ont déjà été menés dans ce sens sans pour autant obtenir des résultats tout à fait satisfaisants et gérant tous les cas de figure que nous avons établis.

Dans un deuxième temps, il sera utile de finir l'étude de répartition optimale des paramètres afin d'en dégager une loi permettant de construire un unique paramètre de réglage du compromis entre temps et précision de calcul. Ainsi un non spécialiste pourra facilement utiliser et régler notre méthode pour sa propre application qui pour l'instant nécessite le réglage de plusieurs paramètres.

Finalement il faudra penser à intégrer à la méthode de résolution de collisions, un module de détection de collision entre polyèdres qui soit performant. Pour l'instant avec notre détection tout à fait naïve, c'est en effet, entre 50 et 80 pourcent du temps de calcul total qui est passé à déterminer et modéliser les collisions.

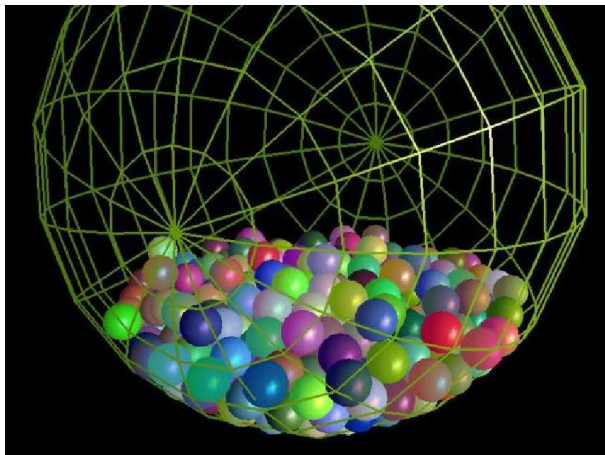
A Quelques images



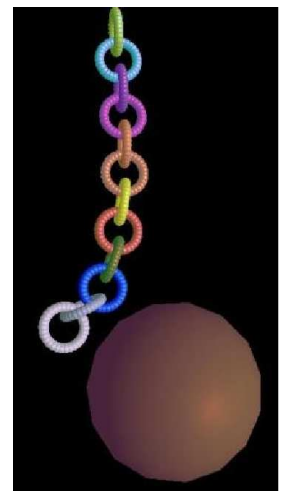
Tourniquet : 36 barres de 6 sphères, 10 tores de 10 sphères et 36 sphères dans un cube



Pile : 13 tores de 30 sphères



316 spheres : 316 spheres dans une sphère creuse



Chaîne : 12 tores de 30 sphères



Pendule : pendule articulé composé de 12 barres rigides de 6 sphères chacunes

Références

- [Bar89] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3) :223–232, 1989.
- [Bar91] David Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4) :31–40, 1991.
- [Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. *Computer Graphics*, 28(Annual Conference Series) :23–34, 1994.
- [Bar96] David Baraff. Linear-time dynamics using Lagrange multipliers. *Computer Graphics*, 30(Annual Conference Series) :137–146, 1996.
- [BtDP⁺02] B. Brogliato, AA ten Dam, L Paoli, F Genot, and M Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems, march 2002.
- [Fau99] Francois Faure. Fast physically-based simulation of nonpenetrating rigid bodies. unpublished, january 1999.
- [Fle00] R. Fletcher. *Practical Methods of Optimization*. Wiley, may 2000.
- [Hah88] James K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4) :299–308, 1988.
- [MC94] Brian Mirtich and John F. Canny. Impulse-based dynamic simulation, 1994.
- [MC95] Brian Mirtich and John F. Canny. Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, pages 181–188, 1995.
- [Mir96a] B. Mirtich. Hybrid simulation : combining constraints and impulses, 1996. Technical Report, Department of Computer Science, University of California, Berkeley.
- [Mir96b] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools : JGT*, 1(2) :31–50, 1996.
- [Mir98] B. Mirtich. Rigid body contact : Collision detection to force computation, 1998. Technical Report TR-98-01, Mitsubishi Electrical Research Laboratory.
- [Mir00] Brian Mirtich. Timewarp rigid body simulation. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 193–200. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
- [MS01] Victor J. Milenkovic and Harald Schmidl. Optimization-based animation. *Computer Graphics Proceedings*, (Annual Conference Series) :37–46, 2001.
- [MW88] Matthew Moore and James Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4) :289–298, 1988.
- [PTVF93] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : The Art Of Scientific Computing*. Cambridge University Press, january 1993.
- [RB94] E. Rimon and J. Burdick. Mobility of bodies in contact - I : A new 2nd order mobility index for multiple-finger grasps, 1994. Submitted to IEEE Trans. on Robotics and Automation.
- [SS98] J. Sauer and E. Sch. A constraint based approach to rigid body dynamics for virtual reality applications, 1998. Proceedings of the ACM Symposium on Virtual Reality Software and Technology.
- [WB97] Andrew Witkin and David Baraff. Physically based modeling : Principles and practice, 1997. Siggraph '97 Course notes.