**PHS4700**

**Physique pour les applications multimédia**

Automne 2015

PAGE COUVERTURE **OBLIGATOIRE** POUR TOUS LES DEVOIRS

**Numéro de devoir : 03**

**Numéro de l’équipe : 14**

|  |
| --- |
| Nom: Rose Prénom : Alexandre matricule: 1580973  Signature : |
| Nom: Mainville Prénom : David matricule: 1636075    Signature : |
| Nom: Gosselin Prénom : Antoine matricule: 1588443    Signature : |
| Nom: Farvacque Prénom : Dylan matricule: 1684271  Signature : |

Table des matières

[I – Description du problème 2](#_Toc435460407)

[II – Équations importantes 3](#_Toc435460408)

[III – Méthode de résolution des équations du mouvement 4](#_Toc435460409)

[IV – Description du logiciel 5](#_Toc435460410)

[V – Résultats obtenus 6](#_Toc435460411)

[VI – Analyse des résultats obtenus 7](#_Toc435460412)

[VII - Discussions sur le devoir 8](#_Toc435460413)

# I – Description du problème

Pour ce présent devoir, notre tâche consiste à simuler le lancer d’une balle de tennis qui a pour cible une boite de conserve remplie d’air en chute libre. Notre simulation devra être capable de fournir les composantes en X, Y et Z des vitesses et centres de masse selon le temps des différents objets. Enfin, en cas de collision entre la balle et la boite, notre simulation devra indiquer la vitesse du centre de masse ainsi que la vitesse angulaire après la collision entre la boite et la balle.

Nous nous intéresserons à quatre scenarios distincts :

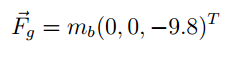
* La boite de conserve n’a pas de vitesse angulaire et le lancer de la balle ne possède que des vitesses selon l’axe des x et des z,
* Toujours avec une boite conserve sans vitesse angulaire, le lancer de la balle aura aussi une vitesse en y
* Dans le troisième scenario, la balle n’aura qu’une vitesse en x et en y et la boite de conserve aura une vitesse angulaire autour de l’axe des y
* Enfin, dans le dernier scenario, la boite aura toujours une vitesse angulaire selon l’axe des y et le lancer de la balle aura une vitesse non nulle selon tous les axes.

Dans le présent rapport, un bref rappel des équations nécessaires à la simulation sera fait. Ensuite, notre simulation développée sur MATLAB sera présentée. Aussi, nous présenterons nos résultats et en ferons une analyse détaillée. Enfin, nous conclurons par une discussion sur les problèmes que nous avons dû surmonter au cours du devoir en ce qui a trait à la programmation et aux simulations.

# II – Équations importantes

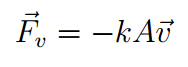
## Équations du mouvement à résoudre

La force gravitationnelle sur la balle est donnée par une seule multiplication de constante sur l’axe de la hauteur (dans la situation présente l’axe des Z).



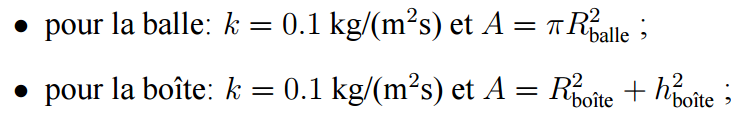
Équation 1

Nous utilisons comme équation pour le frottement visqueux la formule donnée dans l’énoncé.



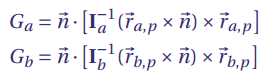
Équation 2

Les valeurs de k sont aussi données dans l’énoncé du travail



## Équations qui contrôlent la simulation

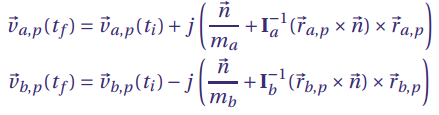
## Détection de collision et vitesses finales des objets

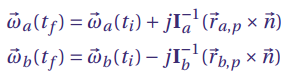


http://puu.sh/lq0NA/38ba9c2357.png

http://puu.sh/lq0O1/2230f858ee.png

http://puu.sh/lq0P8/7d114e1d54.png





# III – Méthode de résolution des équations du mouvement

Les équations de mouvement de la balle et du cylindre sont approximées numériquement à l’aide de Runge Kutta d’ordre 4. Nous fournissions à Runge Kutta les paramètres de pas et de vitesse. Avec cette méthode d’approximation numérique il est possible pour chaque itération de calculer la somme des forces qui est égale à la masse multiplié l’accélération, puis à l’aide des vitesses initiales de trouver les vitesses finales et les positions finales.

Pour calculer les forces de frottement visqueux, nous avons utilisé la formule donnée dans l’énoncé avec la constante *k* donnée dont seule la vitesse varie à chaque itération.

Pour la rotation de la boite, nous incrémentons une variable contenant son vecteur de rotation d’une valeur égale à la vitesse angulaire divisé par le pas à chaque itération. Donc, lorsque nous avons besoin d’effectuer des calculs sur la boite qui a subi une rotation, nous pouvoir calculer la matrice de rotation globale à l’aide des trois composantes de rotation en x, y et z. C’est la fonction *Rotation* dans notre programme qui est responsable d’obtenir les trois composantes de la matrice de rotation et de les multiplier.

## Justification du pas

Le bon pas a été déterminé expérimentalement. L’idée étant de trouver un pas suffisamment petit pour que la précision minimale de 1mm soit respectée mais suffisamment grand pour que le temps d’exécution ne soit pas trop important. Avec un pas de 0,00005 nous obtenons une erreur maximale de 0,4734 mm dans toutes les simulations. On peut donc dire que notre précision est de 0,5 mm, ce qui est amplement suffisant pour respecter les contraintes du laboratoire.

## Description des vérifications effectuées pour assurer la précision

Pour valider la précision de notre résultat lorsque nous obtenons une collision, nous calculons la profondeur d’enfoncement de la balle. Avec la méthode que nous utilisons pour calculer le point de collision, ses coordonnées se trouvent obligatoirement à la surface de la boite. Ainsi, nous pouvons calculer la distance entre le centre de masse de la balle et le point de collision. La différence de cette distance avec le rayon de la balle correspond donc à l’enfoncement de la balle dans la boite ou encore l’erreur maximale sur la position réelle du point de collision. Nous affichons le résultat en millimètres c’est pourquoi la formule ci-dessous multiplie le résultat par 1000.

(rayon\_balle - norm(pointCollision - balle.CentreDeMasse(1:3,end)))\*1000

Équation du calcul de précision (en mm)

# IV – Description du logiciel

Les simulations sont entièrement réalisées à l’aide du logiciel MATLAB. Tout d’abord on initialise nos objets, soit la balle et le cylindre. On définit chacune des propriétés disponibles dans l’énoncé pour ces deux objets et l’on calcule leur centre de masse pour les futures équations.

On définit ensuite un pas suffisamment petit afin d’obtenir la précision minimale requise. On définit également une rotation par pas en fonction de la vitesse angulaire initiale et du pas.

Une fois les diverses initialisations effectuées, on commence le début des simulations qui est une boucle while qui s’arrête lorsque la balle touche le sol (balle.CentreDeMasse(3,end) - balle.Rayon <= 0) ou qu’une collision avec le cylindre est confirmée. La boucle de simulation commence par incrémenter le temps de la simulation d’un pas. Ensuite on vérifie si le temps de lancer la balle est atteint ; Si c’est le cas on déplace la balle. On utilise RK4 afin de mettre à jour la position et la vitesse de la balle en fonction du temps courant. On effectue les mêmes opérations pour le déplacement du cylindre, sans toutefois avoir besoin de valider qu’un certain temps est atteint puisque le cylindre est en mouvement à partir du temps 0. Cependant en plus d’être déplacé, le cylindre est rotationné à l’aide des matrices de rotations vu au TP1.

Suite au déplacement des deux objets, on effectue un test de collision. Afin de ne pas effectuer de validation complexe lorsque ce n’est pas nécessaire le test vérifie d’abord s’il y a collision entre la balle et une sphère englobant le cylindre. Si ce n’est pas le cas le test retourne 0, sinon il poursuit avec une validation plus sophistiqué. La validation supplémentaire effectue des projections dans le plan. Premièrement, on change le référentiel pour que la boite soit droite et son centre soit à l’origine. Ensuite on projette dans le plan « xy ». Si la distance entre le centre de la balle et le centre du cylindre est inférieure à la somme des deux rayons on poursuit avec une validation en « xz ». Si le centre de la balle est à la même hauteur que le cylindre (c’est-à-dire entre les deux extrémités de celui-ci) alors il y a collision. Sinon on vérifie si les positions en x et y de la balle sont à l’intérieur du cylindre. Si c’est le cas on vérifie que le z de la balle est moindre que la hauteur du cylindre divisé par deux. Si le dernier cas n’est pas concluant il reste à valider si la collision a lieu avec le rebord du cylindre. En un premier temps il faut trouver le point le plus près de la balle sur le rebord du cylindre. Pour y arriver, on fait la projection du vecteur allant de l’origine au centre de la balle, puis on le normalise. Ceci nous donne un vecteur unitaire ayant la direction du point le plus proche de la sphère. Il suffit de multiplier par le rayon du cylindre pour obtenir les coordonnées. Ensuite, on calcule la distance entre le centre de la balle et le point le plus près du cylindre. Si cette distance est inférieure au rayon de la balle alors il y a collision.

Suite au test de collision, on réagit en mettant fin à la boucle si le résultat est positif. De plus, dans un tel cas, on note la position de collision et l’on affiche la distance ainsi que le point de collision.

Enfin, la dernière étape de la boucle de simulation est la validation que la balle n’a pas atteint le sol. Si c’est le cas on met fin à la simulation.

Une fois la simulation terminée (que ce soit par collision avec le cylindre ou le sol) on affiche le graphique des positions selon le temps.

# V – Résultats obtenus

# VI – Analyse des résultats obtenus

.

# Graphique de la trajectoire

En rouge, la trajectoire de la boite en chute libre. En bleu, la trajectoire de la balle lancée.

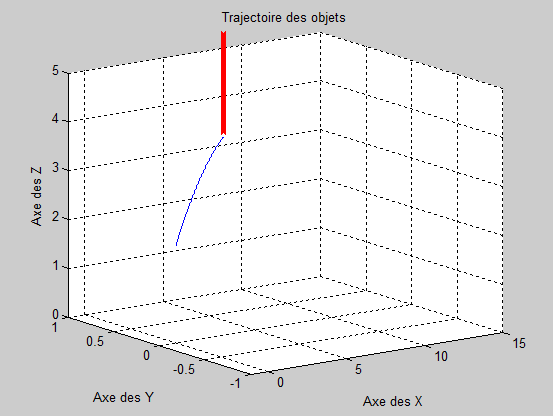


Figure 2: Simulation 1

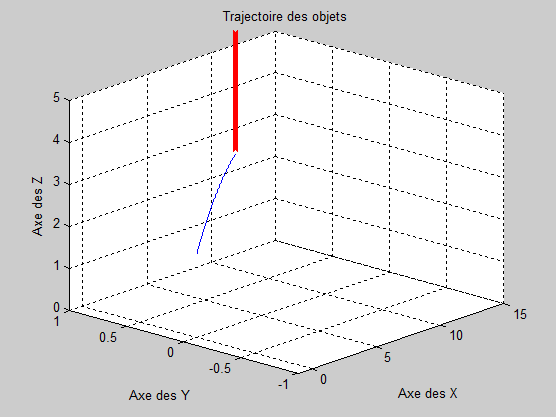


Figure 1: Simulation 2

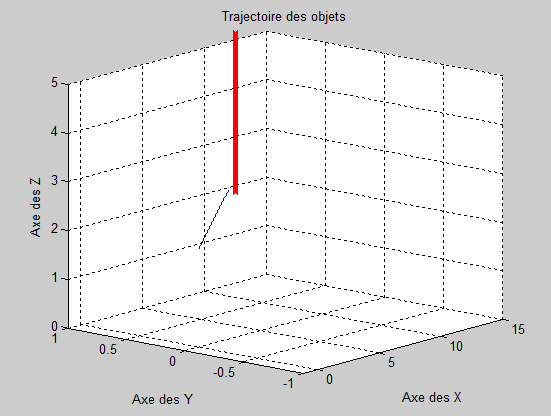


Figure 3: simulation 4

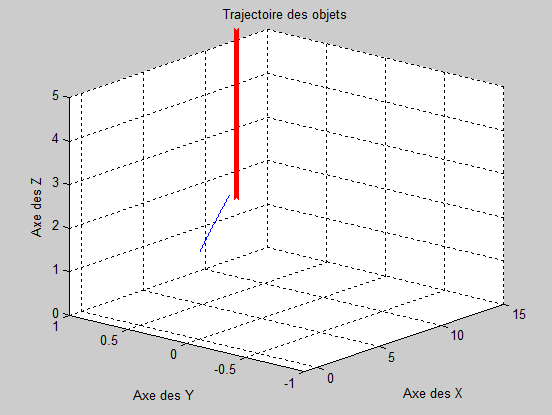


Figure 4: Simulation 3

# VII - Discussions sur le devoir

Comme lors du second laboratoire, nous avons tenté de mettre de l’ordre dans notre code en créant des objets contenant les propriétés physiques de la balle et de la boite.

Nous avons rencontré des difficultés en ce qui a trait à la vérification de collision. Notre model initial détectait les collisions en vérifiant si plusieurs petites balles contenues dans le cylindre entraient en collision avec la balle lancée. Cette méthode était facile à implémenter puisque la détection de collision entre deux sphère est relativement simple. Cette modélisation du cylindre comme étant plusieurs sphères combinées n’était pas assez précise et nous avons dû laisser tomber cette méthode. Nous avons néanmoins conservé la pré-validation de collision avec une sphère englobant le cylindre, ce qui allège les calculs faits à chaque itération.