**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 en la simulation d’un tir d’une boite de conserve remplie d’air en chute libre par une balle. 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 indique la vitesse du centre de masse ainsi que la vitesse angulaire après la collision entre la boite et la balle.

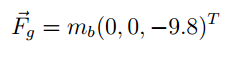
Nous nous intéresserons à quatre scenarios distincts :

* La boite de conserve n’as pas de vitesse angulaire et le lancer de la balle ne possède que des vitesses selon l’axe des z 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 selon 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.

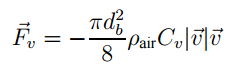
Dans le présent rapport, un bref rappel des équations nécessaires à la simulation sera fait. Ensuite, notre simulation développée sur MATLAB sera présentée. Aussi, nous présenterons nos résultats et en ferons une analyse détaillée. Enfin, nous conclurons par une discussion sur les problèmes que nous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

# II – Équations importantes

1. Équations du mouvement à résoudre
2. Équations qui contrôlent la simulation
3. Détection de collision et vitesses finales des objets

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

 La force de frottement visqueux est donnée par la multiplication du coefficient de frottement visqueux (air) avec la densité de l’air à 30 °C (Cv), la vitesse () multiplié avec sa norme et la moitié de l’aire de la balle ( d2b/8).

Équation 2

# III – Méthode de résolution des équations du mouvement

Les équations de mouvement de la balle et du cylindre sont réalisées à l’aide de Runge Kutta d’ordre 4. Nous fournissions à Runge Kutta les paramètres de pas et de vitesse.

1. Justification du pas

Le bon pas a été déterminé expérimentalement. L’idée étant de trouvé un pas suffisamment petit pour que la précision minimale soit respecté mais également suffisamment grand pour que le temps d’exécution ne soit pas trop important. Avec un pas de 0,0001 nous obtenons une précision de 0,1585 mm ce qui nous apparaît suffisant.

1. 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. C’est-à-dire que nous calculons de quelle distance nous avons arrêté la balle trop tard. Nous affichons le résultat en mm 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. D’abord 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. Pour ce faire 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

.

# VII - Discussions sur le devoir

Dans le cadre de ce second laboratoire, nous avons eu quelques défis à surmonter. Tout d’abord, nous avions remarqué que les calculs pour l’ensemble des simulations étaient très semblables, nous avons donc décidé de découper tout cela dans des classes spécifiques et utiliser l’héritage pour dupliquer le moins possible. Or, MATLAB est avant tout un langage de calcul avant d’être un langage de programmation oriente objet. Nous avons donc dû nous adapter à celui-ci afin d’en tirer le plus d’avantages possibles et de rendre notre code plus lisible et facile d’accès.

Nous avons également rencontré des difficultés en ce qui a trait à la vérification de collision. Notre modèle initial détectait les collisions en vérifiant si plusieurs petites balles contenues dans le cylindre collisionnait avec la balle lancé. Cette modélisation du cylindre comme étant plusieurs sphères combinée n’était pas assez précise et nous avons dût laisser tomber cette méthode. Nous avons néanmoins conservé la pré-validation de collision avec une sphère englobant le cylindre.