****

**PHS4700**

**Physique pour les applications multimédia**

Automne 2016

PAGE COUVERTURE **OBLIGATOIRE** POUR TOUS LES DEVOIRS

**Numéro de devoir : 03**

**Numéro de l’équipe : 16**

|  |
| --- |
| Nom: Tremblay Prénom : David matricule: 1748125  Signature : |
| Nom: Cech Prénom : Jean Paul matricule: 1794611  Signature : |
| Nom: Desrochers Prénom : Pascal matricule: 1689838  yeaboi2  Signature : |
| Nom: Zhong Prénom : Zihui matricule: 1687994  Signature : |

Table des matières

[1 Mise en situation 3](#_Toc467104406)

[2 Rappel théorique 4](#_Toc467104407)

[2.1 Force gravitationnelle 4](#_Toc467104408)

[2.2 Vitesses finales après une collision 4](#_Toc467104409)

[2.3 Équations pour déterminer si une collision se produit 6](#_Toc467104410)

[3 Méthode de résolution 7](#_Toc467104411)

[4 Résultats 8](#_Toc467104412)

[5 Analyse des résultats 13](#_Toc467104413)

[5.1 Tirs 1-3 (Collision) 13](#_Toc467104414)

[5.2 Tirs 4-6 13](#_Toc467104415)

[6 Conclusion 14](#_Toc467104416)

Liste des figures

[Figure 1: Équation de la force gravitationnelle 4](#_Toc467104417)

[Figure 2: Équations des facteurs Ga et Gb 4](#_Toc467104418)

[Figure 3: Équation de α 4](#_Toc467104419)

[Figure 4: Équation de la vitesse relative 5](#_Toc467104420)

[Figure 5: Équation de j 5](#_Toc467104421)

[Figure 6: Vitesses finales après la collision 5](#_Toc467104422)

[Figure 7: Vitesses angulaires finales après la collision 5](#_Toc467104423)

[Figure 8: Trajectoires du tir 1 10](#_Toc467104424)

[Figure 9: Trajectoires du tir 2 10](#_Toc467104425)

[Figure 10: Trajectoires du tir 3 11](#_Toc467104426)

[Figure 11: Trajectoires du tir 4 11](#_Toc467104427)

[Figure 12: Trajectoires du tir 5 12](#_Toc467104428)

[Figure 13: Trajectoires du tir 6 12](#_Toc467104429)

Liste des tableaux

[Tableau 1: Valeurs initiales des six tirs 8](#_Toc467104430)

[Tableau 2: Résultats des simulations des tirs 9](#_Toc467104431)

# Mise en situation

Pour ce troisième devoir, il nous est demandé de réaliser une simulation des trajectoires d'un bloc cubique ainsi que d'une balle. Le bloc cubique est éjecté d'un canon et la balle est lancée afin d'essayer d'entrer en collision avec le bloc pour le faire dévier. Six tirs doivent être simulés, chacun ayant un temps de lancer de la balle, une vitesse initiale pour chaque objet ainsi qu'une vitesse angulaire pour le bloc cubique qui sont préétablis. Notre simulation doit pouvoir fournir les composantes *x, y* et *z* du centre de masse des deux objets dans le temps afin d'obtenir leurs trajectoires. À la fin de la simulation, celle-ci doit indiquer si une collision a eu lieu et déterminer la vitesse du centre de masse et la vitesse angulaire pour les deux objets. Seulement la force gravitationnelle affecte les objets lors de la simulation (pas de force de frottement).

Dans les prochaines sections de ce devoir, les équations physiques nécessaires pour réaliser le travail demandé seront brièvement décrites. Ensuite, les méthodes de résolution utilisées dans le logiciel seront discutées et une analyse de la précision de la méthode pour s'assurer que nos erreurs maximales sont dans les normes acceptables sera effectuée. Les résultats seront ensuite illustrés sous forme de tableaux et de graphiques et leur validité sera discutée. Finalement, une brève conclusion énumérant quelques problèmes survenus lors la réalisation de ce devoir servira de conclusion.

# Rappel théorique

Toutes les équations nécessaires à la réalisation de ce travail sont présentées et brièvement discutées dans les sous-sections qui suivent.

## Force gravitationnelle

La force gravitationnelle représente l'effet de la gravité terrestre sur le bloc cubique ainsi que sur la balle. Son effet est seulement sur la hauteur (l'axe des *z* dans le cas présent). -9,8 représente l'accélération gravitationnelle sur la Terre qui est de 9,8 m/s2.

https://i.gyazo.com/653dad468405b0b9c39ca48a545d8206.png

Figure 1: Équation de la force gravitationnelle

## Vitesses finales après une collision

Pour calculer nos vitesses des centres de masses et vitesses angulaires, nous avons besoin de calculer les facteurs Ga et Gb. La normale au plan de collision , le moment d'inertie  *I* et le vecteur y sont impliqués.

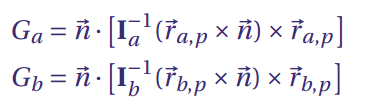


Figure 2: Équations des facteurs Ga et Gb

Les facteurs G mentionnés ci-haut sont utilisés dans l'équation de la figure 3 pour calculer le . ma et mb représentent les masses des deux objets qui entrent en collision.

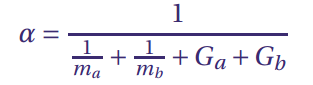


Figure 3: Équation de α

L'équation suivante sert à déterminer la vitesse relative des objets juste avant la collision. représente encore une fois la normale au plan de collision.

https://i.gyazo.com/c97a58e46976f17e8e527e178bf76ba6.png

Figure 4: Équation de la vitesse relative

Nous pouvons déterminer notre composante normale à l'impulsion *j* à l'aide des trois équations ci-haut. représente le coefficient de restitution entre le bloc et la balle. Celui-ci nous est fourni et est de 0,8.

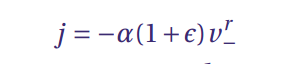


Figure 5: Équation de j

Les vitesses finales du bloc et de la balle après la collision sont données par les équations de la figure 2. est la normale au point de contact où la collision se produit et *I*  est le moment d'inertie. Les autres variables impliquées sont calculées à l'aide des équations illustrées au début de la sous-section 2.2.

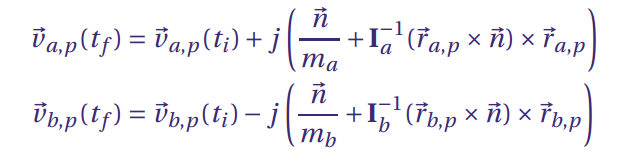


Figure 6: Vitesses finales après la collision

Les vitesses angulaires finales de nos deux objets sont données par les équations suivantes. Les vitesses angulaires des objets avant la collision ainsi que le moment d'inertie y sont impliqués.

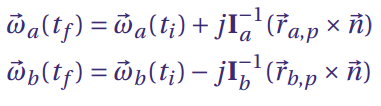


Figure 7: Vitesses angulaires finales après la collision

## Équations pour déterminer si une collision se produit

Les deux conditions d'arrêt de la simulation sont :

* Une collision se produit entre le bloc et la balle
* Le bloc ou la balle touche le sol

Pour déterminer si une des deux conditions se produit, nous créons une sphère de rayon:

correspondant à la somme du rayon maximal du bloc et de la balle. Ensuite, nous vérifions si la distance entre les objets est inférieure à ce rayon. Si c'est le cas, on se retrouve en présence d'une éventuelle collision et on doit donc vérifier plus précisément ce qui se produit. On réduit notre pas dans le temps pour obtenir une plus grande précision lors de la collision et on effectue une rotation du système d'axe. Pour la détection de collision, nous utilisons la méthode des plans séparateurs, s’il a un plan possible qui sépare les 2 solides, il n’y a pas de collision. Cette méthode est plus difficile dans le cas présent, car on utilise d’habitude les faces des 2 solides et nous sommes en présence d’une sphère, qui a théoriquement une infinité de faces. Pour ce faire, nous avons choisi de séparer cette collision en 3 cas distincts; la collision sphère-face du cube, sphère-arrête du cube et sphère-coin du cube. Dans chaque cas on évalue un plan qui est perpendiculaire à la droite liant les points les plus proches de la sphère et du plan/ligne/coin. Si au moins un plan séparateur est trouvé, il n’y a pas de collision, sinon il a collision.

Pour satisfaire la condition d’arrêt, un des plans calculés doit avoir le point de la sphère le plus proche du plan situé à +/- 1 mm du plan et les points au centre des solides situé sur les bords opposés du plan. C’est ainsi qu’on garantit la précision du point de collision, et comme la vitesse est exacte, la précision de l'ensemble de la simulation.

# Méthode de résolution

Nous avons réalisé la programmation de notre simulation sur le logiciel MATLAB. La méthode de résolution pour obtenir les résultats est décrite à la suite. De plus, une analyse de la précision de cette méthode incluant une vérification que les résultats ne contiennent pas d'erreurs dépassant l'erreur maximale permise (1mm) y est présentée. Notre simulation se termine si une collision se produit ou si un des deux objets touche le sol avant qu'une collision se produise.

        La classe gravity.m permet d'obtenir la position d'un objet à l'aide d'une position initiale, une vitesse initiale et un . La seule force impliquée dans cette classe est la force gravitationnelle. C'est donc la méthode gravity() qui sera utilisée pour obtenir la position du bloc et de la balle tout au long de la simulation.

        La classe rotation.m, quant à elle, sert à représenter la rotation de nos objets lors de la simulation. La rotation est égale à la vitesse angulaire multipliée par notre . La rotation est effectuée à l'aide d'une approximation numérique utilisant des quaternions. Les quaternions sont calculés dans les classes QARotation.m, QConjugue.m et QProduit.m

La classe Distsq.m est une petite classe servant à déterminer la distance à la puissance 2 entre nos deux objets. Elle est utilisée dans la classe Devoir3.m pour détecter si nos objets sont proches un de l'autre afin de diminuer notre pour obtenir une meilleure précision pour la collision.

La classe CollisionDetection.m sert à déterminer si une collision entre nos objets se produit. La détection de la collision s'effectue à l'aide de la méthode des plans de divisions à partir des plans du bloc cubique dans planeSide.m.

        Finalement, Devoir3.m regroupe les classes ci-haut pour effectuer les calculs demandés et pour obtenir les résultats désirés. Il est intéressant de mentionner nos au fil de la simulation. Tout d'abord, nous utilisons une incrémentation de temps de 0,002 entre chaque mesure de position jusqu'à ce que les objets soient à proximité (la distance étant mesurée dans Disq.m). Ensuite, on réduit notre pas dans le temps afin d'obtenir une mesure plus précise lors de la collision. On détectera la collision à l'aide de la méthode des plans de divisions. Lorsque celle-ci se produit, notre pas est réduit de moitié et sera réduit de moitié à chaque collision.

# Résultats

Les résultats de nos six coups effectués dans notre simulation sont illustrés dans le tableau 2 ci-dessous. Tout d'abord, un premier tableau indique les valeurs initiales de chacun des coups afin d'améliorer la lisibilité du tableau des résultats. Par la suite, les trajectoires de nos objets peuvent être observées dans des graphiques en 3 dimensions.

Tableau 1: Valeurs initiales des six tirs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tir** | **Temps initial**  **(s)** | **Vitesse initiale du bloc**  **(m/s)** | **Vitesse angulaire initiale du bloc**  **(rad/s)** | **Vitesse initiale de la balle**  **(m/s)** |
| **1** | 0.545454 | [-2, -3, 5] | [0, 0, 0] | [5, 2, 0.642424] |
| **2** | 0.545454 | [-2, -3, 5] | [0, 0, 15] | [5, 2, 0.642424] |
| **3** | 0.071429 | [0, -6, 3] | [0, 0, 0] | [7, 0, 0.40834] |
| **4** | 0.071429 | [0, -6, 3] | [0, 0, 15] | [7, 0, 0.40834] |
| **5** | 0.6 | [-2, -3, 5] | [-5, -5, 0] | [5, 2, 0.642424] |
| **6** | 0.1 | [-2, -3, 5] | [0, 0, 0] | [5, 2, 0.1] |

Tableau 2: Résultats des simulations des tirs

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tir** | **Temps final**  **(s)** | **Résultat** | **blocf** | **ballef** | **Position** |
| **1** | 0.8097 | 0 |  |  | Bloc :  Balle : |
| **2** | 0.8102 | 0 |  |  | Bloc :  Balle : |
| **3** | 0.4918 | 0 |  |  | Bloc :  Balle : |
| **4** | 0.7501 | -1 |  |  | Bloc :  Balle : |
| **5** | 1.1822 | 1 |  |  | Bloc :  Balle : |
| **6** | 0.7460 | -1 |  |  | Bloc :  Balle : |
|  |  |  |  |  |  |

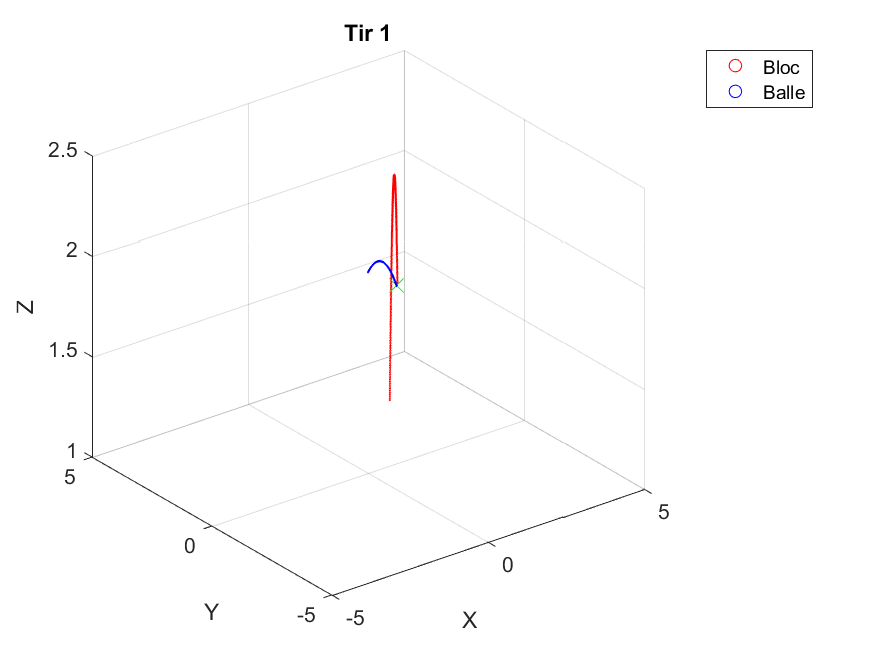


Figure : Trajectoires du tir 1

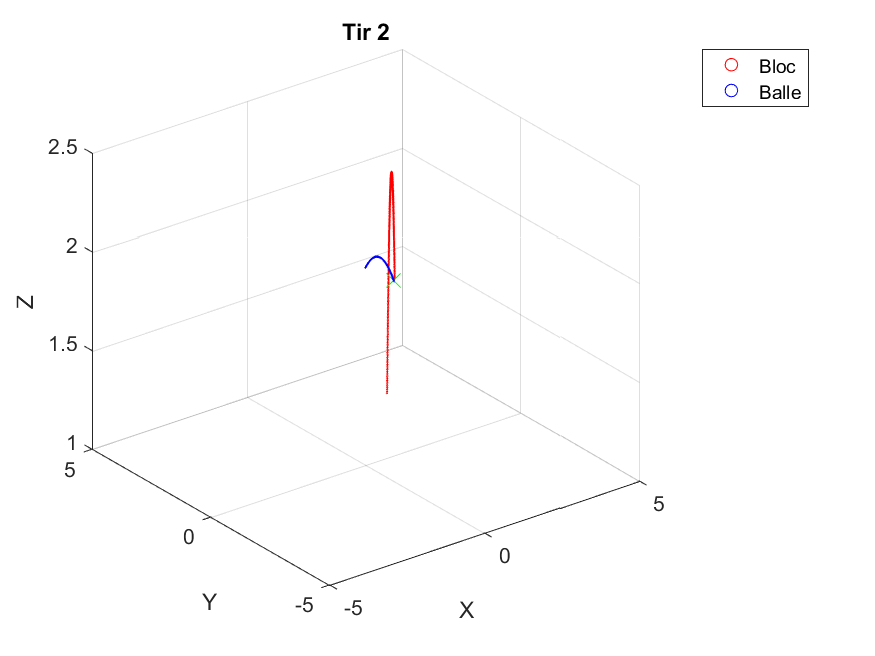


Figure : Trajectoires du tir 2

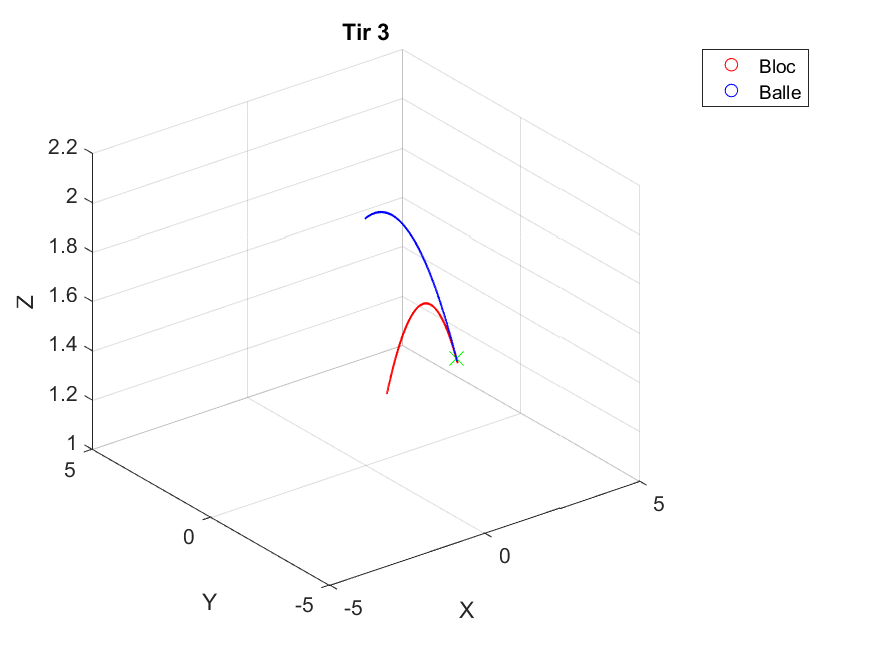


Figure : Trajectoires du tir 3

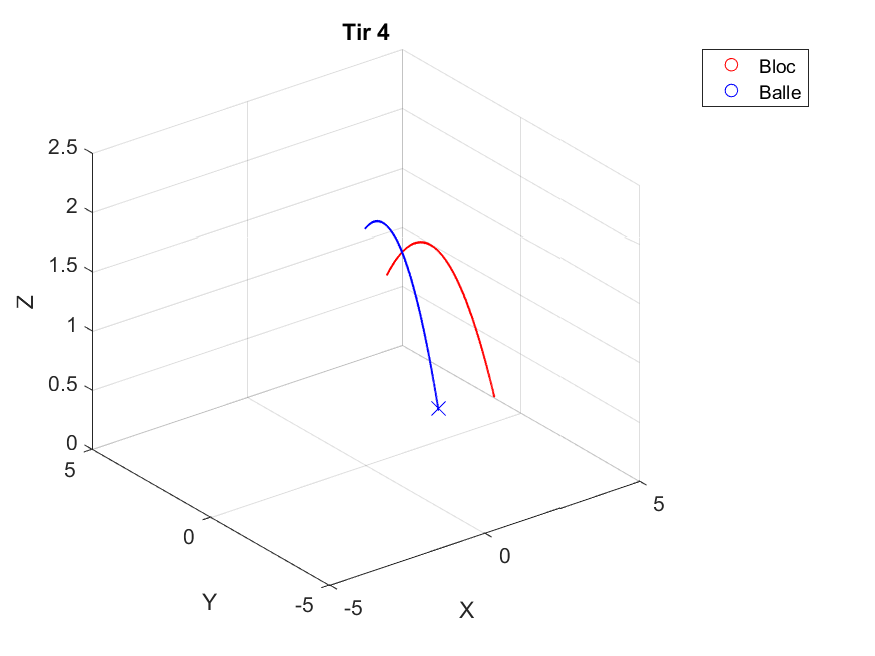


Figure : Trajectoires du tir 4

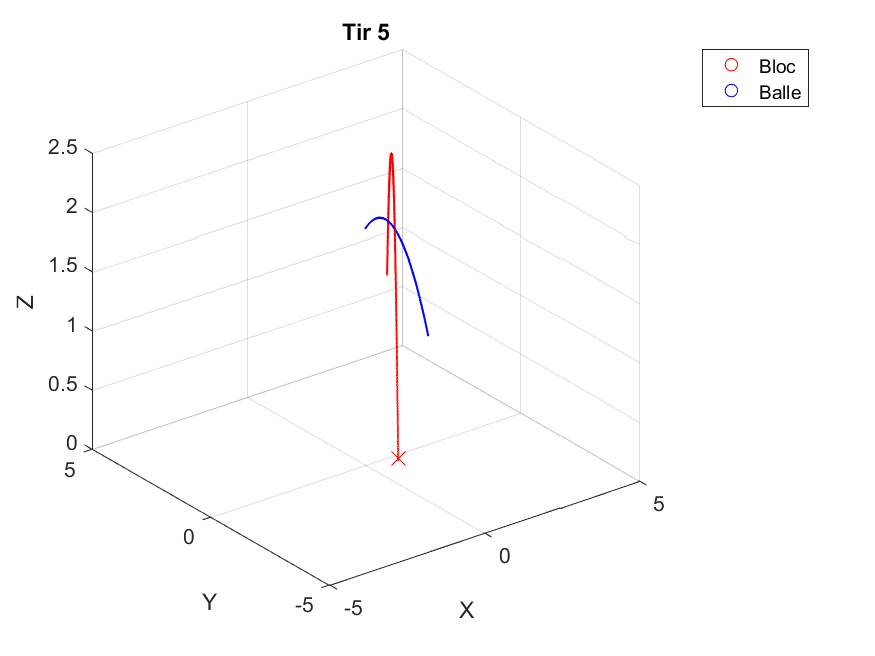


Figure : Trajectoires du tir 5

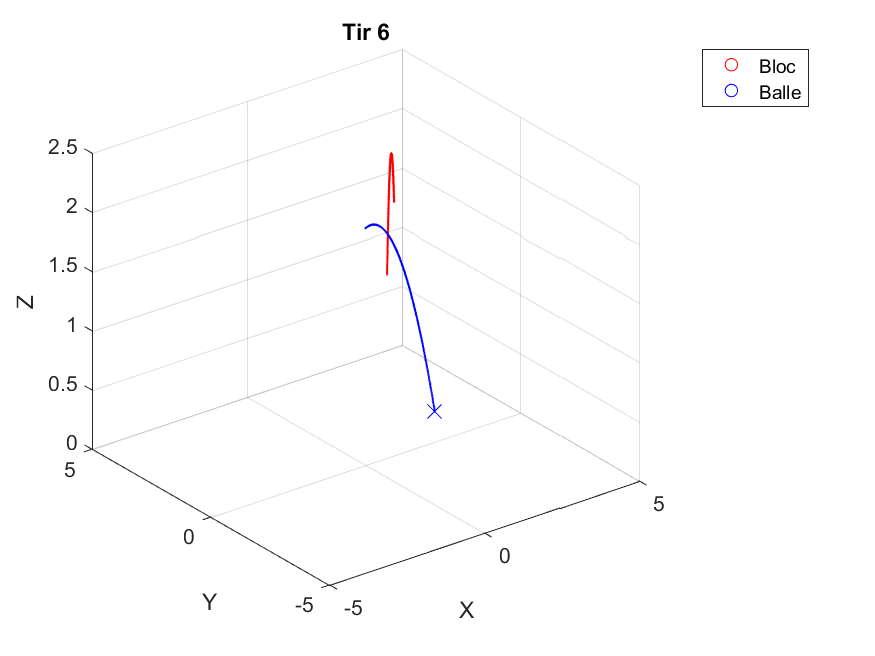


Figure : Trajectoires du tir 6

# Analyse des résultats

Une analyse des résultats obtenus ci-haut est réalisée pour les 6 tirs simulés dans les sous-sections 5.1 et 5.2. Une validation ainsi qu'une courte explication des résultats seront présentées pour chaque tir.

## Tirs 1-3 (Collision)

Comme on peut voir dans le tableau et dans les graphiques des trajectoires, la balle est entrée en collision avec le bloc. Premièrement avant la collision on veut voir que la composante z de la vitesse linéaire de balle a diminué en raison de la force de gravité qui agit dessus. Il en est de même pour la vitesse linéaire initiale du bloc. Après la collision, on peut voir que les composantes x et y des vitesses linéaires des deux solides ont diminuée. En effet, la collision n’est pas parfaitement élastique (e=0.8), donc il y a de l’énergie qui a pu se perdre sous forme de chaleur mais la quantité de mouvement par rapport à la ligne d’impact est conservée. On peut remarquer que la vitesse linéaire en z reste inchangée dans tous les cas de collision. Cela peut être expliqué par le fait que l’axe Z est perpendiculaire à la ligne d’impact. De plus, puisqu’on ne tient pas compte des frottements entre les deux objets, la seule force qui va agir sur la balle est l’impulsion du point de collision vers le centre de masse de la sphère et donc ne va pas générer de moment. Cela explique pourquoi la balle n’a pas de vitesse angulaire après la collision.

## Tirs 4-6

Dans les tirs 4-5-6, on peut voir sur les graphiques des trajectoires qu’aucune collision entre les deux objets n’a eu lieu. Comme dans les tirs précèdent, la vitesse en z diminue en raison de la force de gravité. Dans le cas du tir 4, les vitesses linéaires des deux solides sont pareilles au tir 3. Par contre la vitesse angulaire initiale du bloc le fait dévier de trajectoire et donc la collision n’aura pas lieu. C’est le même principe dans le cas du tir 5. Pour le tir 6, la balle à une vitesse trop grande en z, et donc va passer par-dessus le bloc comme on peut le voir dans le schéma.

# Conclusion

Pour ce troisième livrable, plusieurs difficultés ont dû être surmontées dans le codage de la simulation pour obtenir les résultats désirés. Tout d'abord, notre balle est de forme sphérique et possède donc une infinité de faces. Nous avons donc dû coder les cas pour les collisions sphère-faces, sphères-arrêtes et sphères-coins pour les plans de séparations utilisés dans la détection de collision. Ensuite, les vitesses avant et après la collision doivent être par rapport aux vitesses relatives au point de collision et non par rapport aux centres de masse de nos deux objets comme nous avions fait au départ. Aussi, l'utilisation de quaternions afin de générer une matrice de rotation en 3D était ardue et nous a causé quelques soucis. Finalement, au niveau de l'analyse, il était parfois difficile de valider les résultats avec certitudes.