

Modélisation stochastique des projectiles balistiques

rapport de Projet

Janvier 2023

Auteurs: Ahmed Mansour Bourassine

ahmed-mansour.bourassine@polytechnique.edu



Contents

1	Introduction	2
2	Étude des forces	3
2.1	Poids et gravité	4
2.2	Frottement de l'air et variation de la densité	5
2.3	accélération Normale	5
3	Modélisation de la trajectoire	6
3.1	équation différentielle	6
3.2	Résolution numérique	9
3.3	Implémentations dans python	11
4	API et trajectoire probabilstique	12

1 Introduction

La modélisation stochastique consiste à prendre en compte l'incertitude et l'aléatoire dans l'analyse et la prédiction de phénomènes complexes. Dans le cas d'un projectile, il peut être intéressant d'utiliser une approche stochastique pour tenir compte des variations de la vitesse, du vent, de la position ou de l'angle de lancement, qui peuvent affecter la trajectoire du projectile.

En utilisant des outils statistiques et probabilistes, il est possible de développer un modèle mathématique permettant de représenter la distribution des trajectoires possibles pour un projectile lancé dans un environnement donné. Ce modèle peut être utilisé pour simuler des scénarios différents et prévoir les chances de succès ou d'échec d'une mission, par exemple.

Le projet que je souhaite réaliser me permettra donc de découvrir les techniques de modélisation stochastique utilisées et de les mettre en pratique. Il faut bien choisir les hypothèses et les paramètres de mon modèle en comparant les résultats des simulations avec des mesures expérimentales.

2 Étude des forces

Dans cette partie, nous allons nous intéresser à l'étude des forces et de la trajectoire du Modèle 155mm Howitzer HE, un obusier de l'artillerie lourde utilisé par de nombreux pays dans le monde. Nous disposons de données réelles sur ce projectile, qui nous permettront de valider et de tester notre modèle de modélisation stochastique.

Le Modèle 155mm Howitzer HE est un projectile de grande taille et de poids important, conçu pour être utilisé dans des missions de destruction de cibles en profondeur. Sa trajectoire est influencée par de nombreux facteurs, tels que la vitesse de lancement, l'angle de tir, la résistance de l'air ou encore l'effet de la gravité.

En utilisant notre modèle, nous allons étudier l'influence de ces facteurs sur la trajectoire du Modèle 155mm Howitzer HE et sur les forces qui agissent sur lui pendant son vol. Nous allons également comparer les résultats de notre modèle aux données réelles sur ce projectile, afin de vérifier sa validité et sa précision.



Figure 1: Modèle 155mm Howitzer HE .

on va étudier trois force (et une quatrième à qui on va dédier tout une partie après):

- P : le poids dû à la gravité terrestre.
- T : l'accélération Normale qui augmente avec la vitesse horizontale.
- f : le frottement de l'air

la quatrième force est celle du vent F_w qui doit être modéliser de façon stochastique puisque sa direction et sa norme sont plus aléatoire que ceux des autres forces.

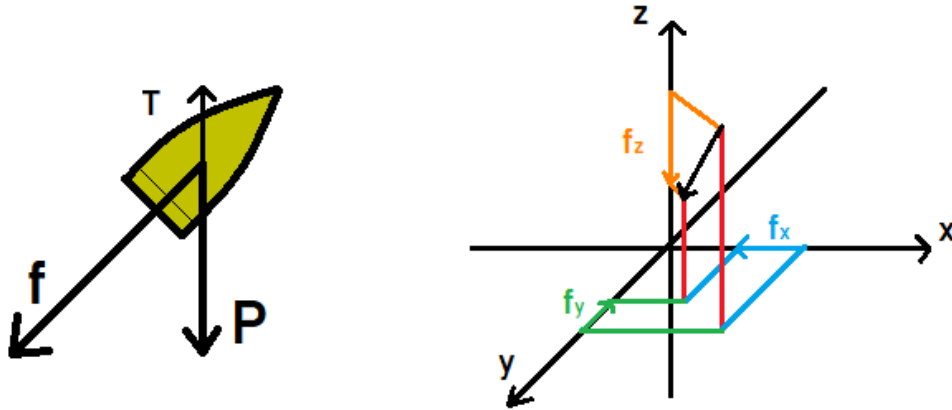


Figure 2: les forces subit et leur projection sur les axes.

2.1 Poids et gravité

$$\|\vec{P}\| = m \cdot g$$

On va considérer que g n'est pas constante ; en faite elle varie en fonction de la hauteur mais le variation est généralement insignificatif. Dans notre cas puisque le projectile peut atteindre 11km d'élévation il faut prendre en compte l'évolution de g au cours de la trajectoire :

$$g(h) = \frac{R_T^2}{(R_T + h)^2}$$

avec $h = \int_0^t V_z dt$ et $R_T =$ rayon de la terre

$$\Rightarrow g(t) = \frac{R_T^2}{\left(R_T + \int_0^t V_z dt\right)^2}$$

2.2 Frottement de l'air et variation de la densité

Pour \vec{F} (frottement) aussi, on va créer un modèle dépendant de la hauteur.

$$\|\vec{F}\| = \text{Coeff} \cdot V^2 = \alpha \cdot AD(h, g) \cdot V^2$$

$$\alpha = \text{coeff de projectile} = \frac{1}{2} \times 0,25 \times \text{Surface du projectile}$$

Il faut mentionner que le 0.25 est la coeff de frottement en régime supersonique de notre projectile ; il est spécifique au type de munition qu'on utilise (155mm Howitzer HE).

Air Density (AD) affecte énormément la friction que le projectile va rencontrer. Plus on monte dans l'atmosphère plus la densité de l'air diminue. Le modèle prend en considération le fait que l'atmosphère n'est isotherme et donc on doit prendre en compte le Lapse-Rate (changement de température en fonction de la hauteur) ; la diminution suit un régime linéaire dans les premiers 10km avec une coeff entre $6 \cdot 10^{-3}\text{C/m}$ et $7 \cdot 10^{-3}\text{C/m}$

$$AD(h, g) = \text{airdensity} = \rho_0 \times \left(\frac{T_{SL}}{T_{SL} + L_{rate} \times h} \right)^{\left(1 + \frac{g \cdot M}{R \cdot L_{rate}} \right)}$$

TSL = temperature at Sealevel

$$L_{rate} = 6,5 \cdot 10^{-3}\text{C/m}$$

R = universal gas constant

M = masse molaire - air

P_0 = density at sea level

2.3 accélération Normale

on ajoute \vec{N} l'accélération que subit le projectile ; il faut soustraire la force dû à sa vitesse acquise par la rotation terrestre qu'on va noter V_{rt} . ainsi ;

$$\|\vec{N}\| = \left[\left(V_{rt} + \sqrt{V_x^2 + V_y^2} \right)^2 - V_{rt}^2 \right] \cdot \frac{1}{R_T + \int_0^t V_z dt}$$

3 Modélisation de la trajectoire

Le premier point (qu'on a déjà traité) concerne l'étude des forces agissant sur un projectile et la manière de les intégrer dans notre modèle mathématique. Nous avons présenté les forces de base (gravité, résistance de l'air,...) .

Dans cette partie on va introduire le vecteur vent comme une force aléatoire. On met ainsi en équation l'influence de ces forces sur la trajectoire du projectile et on en déduit l'équation différentielle, qui résulte de l'intégration des équations de force dans notre modèle. En plus on va formaliser l'incertitude liée au vent sous forme de processus aléatoire.

Enfin, le troisième point (dans la suite) concerne la résolution de l'équation différentielle, pour laquelle nous allons utiliser des méthodes numériques (par exemple, la méthode de Runge-Kutta). L'implémentations de l'algorithme de résolution sera en Python et on testera la précision de la solution obtenue.

3.1 équation différentielle

En écrivant ainsi les équation-diff associé à l'accélération dans les trois axes on obtient les deux systèmes suivants :

Pendant la montée :

$$\left\{ \begin{array}{l} \frac{\partial V_z}{\partial t} = -g(h) - \frac{\alpha}{m} \cdot AD(g, h) \cdot V_z^2 + \|\vec{N}\| + \frac{\|\vec{F}_w(x, y, h)\|_z}{m} \\ \quad = -\frac{R_T^2}{\left(R_T + \int_0^t V_z dt\right)} - \frac{\alpha}{m} \cdot AD(g, h) \cdot V_z^2 + \frac{\left(V_{rt} + \sqrt{V_x^2 + V_y^2}\right)^2 - V_{rt}^2}{R_T + \int_0^t V_z dt} + \frac{\|\vec{F}_w(x, y, h)\|_z}{m} \\ \frac{\partial V_x}{\partial t} = -\frac{\alpha}{m} AD(g, h) V_x^2 + \frac{\|\vec{F}_w(x, y, h)\|_x}{m} \\ \frac{\partial V_y}{\partial t} = -\frac{\alpha}{m} AD(g, h) V_y^2 + \frac{\|\vec{F}_w(x, y, h)\|_y}{m} \end{array} \right.$$

Pendant la descente :

$$\left\{ \begin{array}{l} \frac{\partial V_z}{\partial t} = -g(h) + \frac{\alpha}{m} \cdot AD(g, h) \cdot V_z^2 + \|\vec{N}\| + \frac{\|\vec{F}_w(x, y, h, t)\|_z}{m} \\ \quad = -\frac{R_T^2}{(R_T + \int_0^t V_z dt)} + \frac{\alpha}{m} \cdot AD(g, h) \cdot V_z^2 + \frac{(V_{rt} + \sqrt{V_x^2 + V_y^2})^2 - V_{rt}^2}{R_T + \int_0^t V_z dt} + \frac{\|\vec{F}_w(x, y, h, t)\|_z}{m} \\ \frac{\partial V_x}{\partial t} = -\frac{\alpha}{m} AD(g, h) V_x^2 + \frac{\|\vec{F}_w(x, y, h, t)\|_x}{m} \\ \frac{\partial V_y}{\partial t} = -\frac{\alpha}{m} AD(g, h) V_y^2 + \frac{\|\vec{F}_w(x, y, h, t)\|_y}{m} \end{array} \right.$$

(La différence entre les deux est le signe de la coefficient du frottement car il est toujours opposé au sens du mouvement)

m = masse de projectile

$\vec{F}_w(x, y, h, t)$ = vecteur vent qu'on va définir sa variation comme un vecteur aléatoire

$\frac{\partial \|\vec{F}_w(x, y, h, t)\|}{\partial t} \sim \mathcal{N}(\text{moyenne de vitesse, variance à définir})$

soit φ_x et φ_y les variation des angles que le vecteur vent fait avec le vecteur de la moyenne. alors:

φ_x = angle de déviation horizontale $\sim \mathcal{N}(0, \text{à définir})$

φ_y = angle de déviation verticale $\sim \mathcal{N}(0, \text{à définir})$

\vec{F}_w est la composante stochastique de l'équation différentielle et elle doit être évalué en tout moment

Voilà un exemple de la distribution statistique de la vitesse de vent qu'on peut approximer à l'aide d'une variable aléatoire Normale :

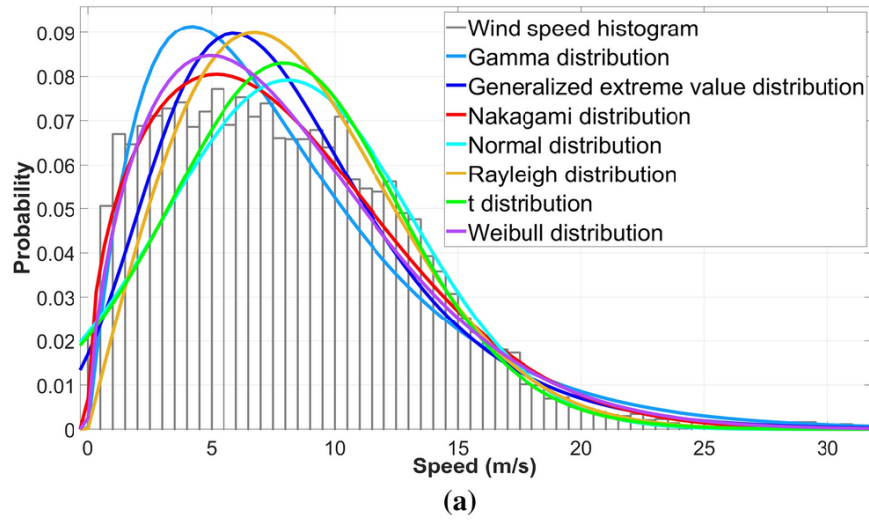


Figure 3: différent valeurs de la vitesse du vent.

Même chose pour l'angle par rapport l'axe des x et des y :

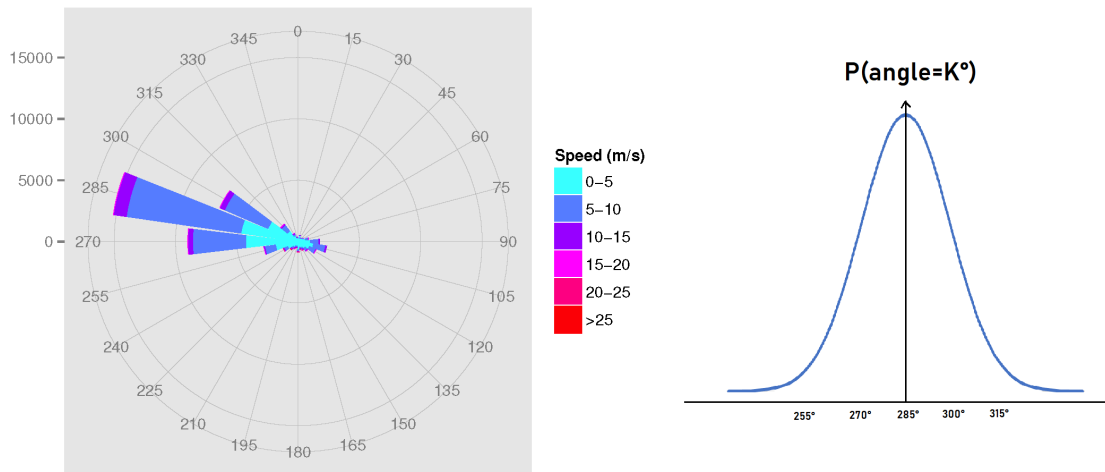


Figure 4: différent valeurs de la vitesse du vent.

3.2 Résolution numérique

Pour résoudre l'équation différentielle qui décrit la trajectoire d'un projectile, il est souvent nécessaire d'utiliser des méthodes numériques. Ces méthodes permettent de traiter de manière précise et rapide des équations complexes qui ne peuvent pas être résolues analytiquement.

L'une des méthodes numériques les plus simples et les plus utilisées est la méthode d'Euler explicite. Cette méthode consiste à découper la durée de simulation en petits intervalles de temps (appelés "pas de temps") et à estimer les valeurs des variables à chaque étape en utilisant la dérivée de l'équation différentielle.

La méthode d'Euler explicite est simple à mettre en œuvre et permet de trouver rapidement une solution approximative de l'équation différentielle. Toutefois, elle présente des limites en termes de précision et peut donner des résultats instables dans certains cas. Il existe donc d'autres méthodes numériques plus précises et robustes, comme la méthode de Runge-Kutta, qui peuvent être utilisées pour résoudre des équations différentielles complexes.

quant on formalise la rédaction on obtient la suite $(y)_{n \in \mathbb{N}}$ tel que $y_n = \begin{pmatrix} y_x \\ y_y \\ y_2 \end{pmatrix}_n$
on définit aussi le pas $= \varepsilon$ (l'unité est en ms)

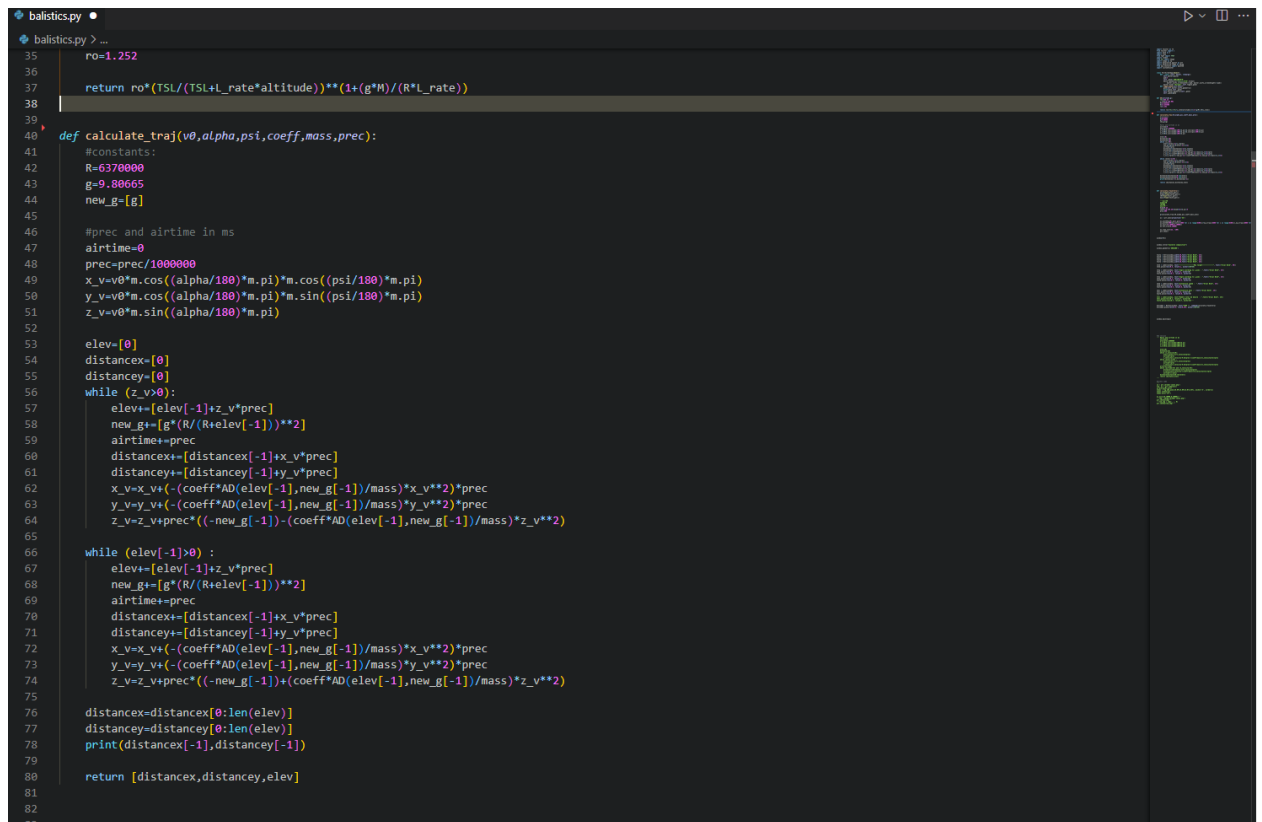
$$\left\{ \begin{array}{l} (y_z)_{n+1} = (y_z)_n + \varepsilon \left[-\frac{R_T^2}{(R_T + h_n)^2} - \frac{\alpha}{m} \cdot AD(g_n, h_n) (y_z)_n^2 + \|\vec{N}_a\|_n + \frac{(\|F_w\|_z)_n}{m} \right] \\ (y_x)_{n+1} = (y_x)_n + \varepsilon \left[-\frac{\alpha}{m} \cdot AD(g_n, h_n) (y_x)_n^2 + \frac{(\|F_w\|_x)_n}{m} \right] \\ \{(y_y)_{n+2} = (y_y)_n + \varepsilon \left[\frac{-\alpha}{m} \cdot AD(g_n, h_n) (y_y)_n^2 + \frac{(\|F_{w_x}\|_y)_n}{m} \right] \} \\ \\ h_{n+1} = h_n + \varepsilon \cdot (y_z)_n \\ (\text{pos}_x)_{n+1} = (\text{pos}_x)_n + \varepsilon (y_x)_n \\ (\text{pos}_y)_{n+1} = (\text{pos}_y)_n + \varepsilon (y_y)_n \\ \\ \|F_w\|_{n+1} = \|F_w\|_n + \varepsilon(x) \text{ où } x \sim \mathcal{N}(0, \text{déviation à définir}) \\ \\ g_n = \frac{R_T^2}{(R_T + h_n)^2} \\ \\ \|\vec{N}_a\|_n = \left[\left(V_{rt} + \sqrt{(y_x)_n^2 + (y_y)_n^2} \right)^2 - V_{rt}^2 \right] \times \frac{1}{R_T + h_n} \\ \\ \text{vector position } P_n = \begin{pmatrix} (\text{pos}_x)_n \\ (\text{pos}_y)_n \\ h_n \end{pmatrix} \end{array} \right.$$

En utilisant une méthode numérique appropriée, il est possible de trouver une solution précise et fiable de l'équation différentielle qui décrit la trajectoire d'un projectile, et de simuler ainsi de manière réaliste les conditions de lancement d'un projectile.

3.3 Implémentations dans python

Pour appliquer ça en Python, il faut d'abord importer les bibliothèques nécessaires, comme NumPy et Matplotlib. Ensuite, il faut définir les paramètres de la simulation, comme le pas de temps, la durée de la simulation et les valeurs initiales des variables.

Il faut créer une boucle qui itère sur le nombre de pas de temps de la simulation et à chaque étape de la boucle, utiliser l'équation différentielle et les valeurs précédentes des variables pour calculer les nouvelles valeurs des variables, puis réinjecter ces valeurs dans le modèle et passer au pas de temps suivant. Enfin, il faut afficher ou enregistrer les résultats de la simulation, comme par exemple les valeurs des variables en fonction du temps ou la trajectoire du projectile sur un graphique.



```
35 ro=1.252
36
37 return ro*(TSL/(TSL+rate*altitude))**(1+(g*M)/(R*I_rate))
38
39
40 def calculate_traj(v0,alpha,psi,coeff,mass,prec):
41     #constants:
42     R=6370000
43     g=9.80665
44     new_g=[g]
45
46     #prec and airtime in ms
47     airtime=0
48     prec=prec/1000000
49     x_v=v0*m.cos((alpha/180)*m.pi)*m.cos((psi/180)*m.pi)
50     y_v=v0*m.cos((alpha/180)*m.pi)*m.sin((psi/180)*m.pi)
51     z_v=v0*m.sin((alpha/180)*m.pi)
52
53     elev=[0]
54     distancex=[0]
55     distancey=[0]
56     while (z_v>0):
57         elev+=elev[-1]+z_v*prec
58         new_g=[g*(R/(R+elev[-1]))**2]
59         airtime+=prec
60         distancex+=distancex[-1]+x_v*prec
61         distancey+=distancey[-1]+y_v*prec
62         x_v=x_v+(-(coeff*AD(elev[-1],new_g[-1])/mass)*x_v**2)*prec
63         y_v=y_v+(-(coeff*AD(elev[-1],new_g[-1])/mass)*y_v**2)*prec
64         z_v=z_v+prec*((-new_g[-1])-(coeff*AD(elev[-1],new_g[-1])/mass)*z_v**2)
65
66     while (elev[-1]>0):
67         elev+=elev[-1]+z_v*prec
68         new_g=[g*(R/(R+elev[-1]))**2]
69         airtime+=prec
70         distancex+=distancex[-1]+x_v*prec
71         distancey+=distancey[-1]+y_v*prec
72         x_v=x_v+(-(coeff*AD(elev[-1],new_g[-1])/mass)*x_v**2)*prec
73         y_v=y_v+(-(coeff*AD(elev[-1],new_g[-1])/mass)*y_v**2)*prec
74         z_v=z_v+prec*((-new_g[-1])+(coeff*AD(elev[-1],new_g[-1])/mass)*z_v**2)
75
76     distancex=distancex[0:len(elev)]
77     distancey=distancey[0:len(elev)]
78     print(distancex[-1],distancey[-1])
79
80     return [distancex,distancey,elev]
```

Figure 5: le code en python.

J'ai également implémenté un GUI (Graphical User Interface) qui permet de tester différentes valeurs de masse, de vitesse et d'angle de lancement d'un projectile, et de visualiser la trajectoire correspondante.

Le GUI est constitué d'une interface graphique qui permet de saisir les valeurs de masse, de vitesse et d'angle de lancement du projectile. Il est également possible de charger des données météorologiques en temps réel à partir d'une API de météorologie, ce qui permet de prendre en compte les conditions météorologiques réelles dans la simulation.

4 API et trajectoire probabilstique

l'utilisation d'API météorologiques peut grandement améliorer la précision de mes simulations et prévisions. En accédant à des données météorologiques en temps réel, je peux prendre en compte les effets des conditions atmosphériques, comme le vent, l'humidité et la température, sur la trajectoire du projectile. Cela peut aider à améliorer la réalisme et la fiabilité de mes résultats, et les rendre plus pertinents aux conditions actuelles.

Par exemple, en accédant à des données de vent en temps réel à partir d'une API météorologique, je peux prédire de manière plus précise l'influence du vent sur la trajectoire du projectile. Cela peut être particulièrement important pour des applications comme les simulations de formation militaire, où la capacité à prédire de manière précise la trajectoire d'un projectile est cruciale.

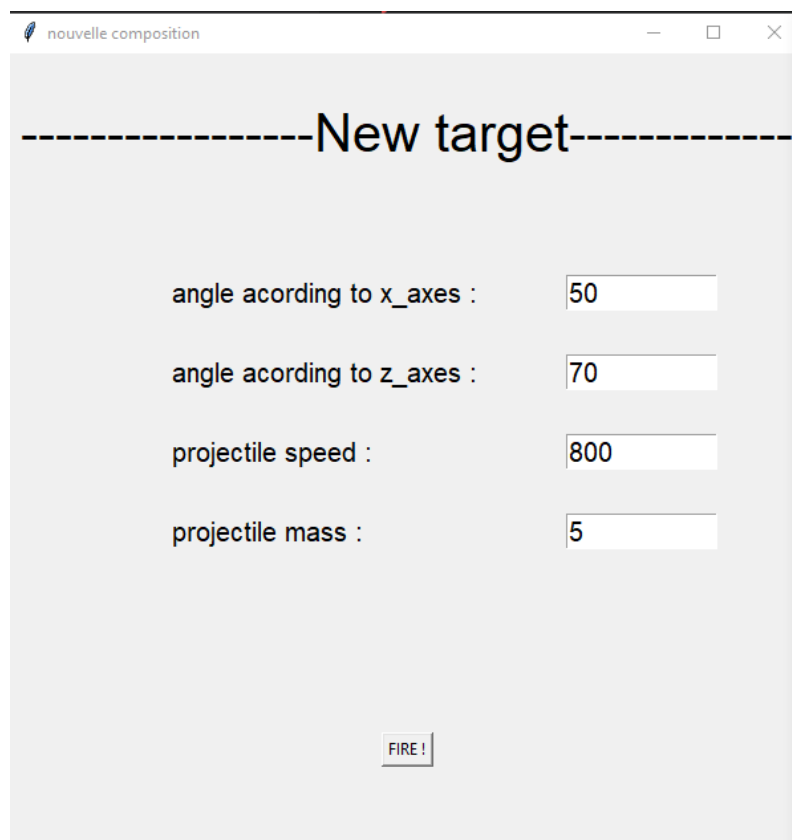
en utilisant les données de 09/01 à Paris 12:00 :

vent : 5m/s

température : 11°C

pression atmosphérique : 1016.5hPa

Voici un test:



nouvelle composition

-----New target-----

angle acording to x_axes :

angle acording to z_axes :

projectile speed :

projectile mass :

Figure 6: paramètre.

ces paramètres produit ce résultats:

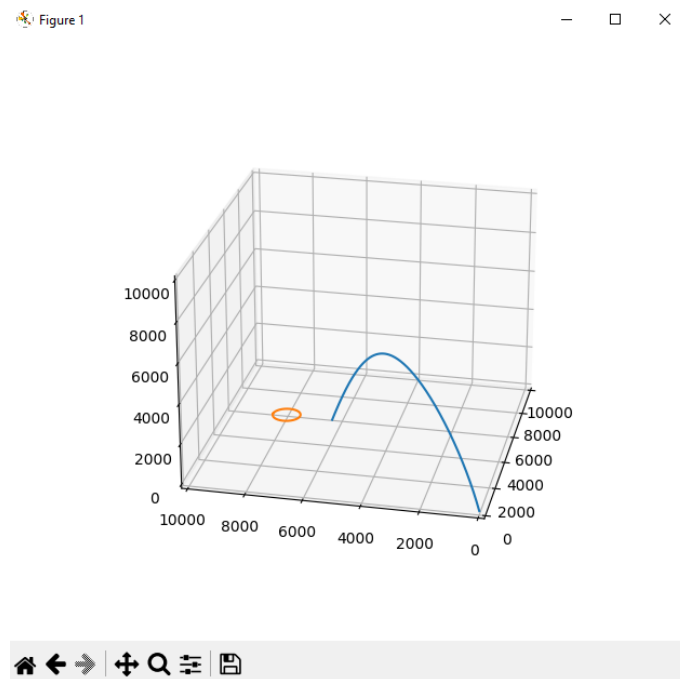


Figure 7: trajectoire.

la trajectoire reste probabiliste mais la déviation possible est pratiquement invisible à l'oeil nue. (on parle de ± 40 m pour des distances de 15km)

en effet on peut même ajouter une composante aléatoire pour la vitesse initiale de la projectile et l'angle de l'envoi.

J'ai constaté que les données sur les déviations liées à la fabrication des munitions étaient limitées et difficiles à obtenir, ce qui a impacté la qualité de mes résultats.

Dans l'avenir, on peut obtenir de meilleures données afin de pouvoir affiner le modèle et obtenir des prévisions encore plus précises. j'espère également pouvoir étendre le modèle à d'autres types de projectiles et à d'autres conditions météorologiques, afin de le rendre encore plus polyvalent et utile.

Voilà le Github du code:

https://github.com/scihelios/stochastic_balistics