

EC222 : Modélisation et Simulation

Organisation

Gaëtan Hello

Université d'Evry Val d'Essonne
UFR Sciences et Technologies
gaetan.hello@univ-evry.fr

L1 SPI 2019-20

- 1 Contexte de l'enseignement
- 2 Contenu des cours/TD
- 3 Travaux Pratiques
- 4 Evaluation

1 Contexte de l'enseignement

2 Contenu des cours/TD

3 Travaux Pratiques

4 Evaluation

Compétences visées :

- ▶ connaissances de base en modélisation, programmation et simulation pour des applications potentielles dans différents champs disciplinaires (GE, GI, GM...),
- ▶ capacité à utiliser, modifier et créer des programmes informatiques simples pour la résolution numérique de problèmes en sciences physiques et de l'ingénieur.

Pré-requis :

- ▶ bases en analyse (dérivation, intégration, ...),
- ▶ bases en algèbre (opérations sur vecteurs et matrices, ...),
- ▶ bases en algorithmie (variables, boucles, structures conditionnelles, ...).

- 1 Contexte de l'enseignement
- 2 Contenu des cours/TD**
- 3 Travaux Pratiques
- 4 Evaluation

Thèmes abordés (6 séances de 2H) :

- ▶ organisation de l'enseignement
- ▶ résolution numérique de problèmes d'optimisation
 - principes de l'optimisation continue
 - méthodes de la dichotomie et de Newton-Raphson (résoudre $f(x) = 0$)
 - applications (problèmes physiques, méthode des moindres-carrés)
- ▶ résolution numérique d'équations différentielles
 - notions de modèles et modélisations
 - schémas d'Euler explicite et implicite
- ▶ optimisation combinatoire
 - problèmes typiques (voyageur de commerce...)
 - notions de complexité des algorithmes
- ▶ méthodes de Monte-Carlo

- 1 Contexte de l'enseignement
- 2 Contenu des cours/TD
- 3 Travaux Pratiques**
- 4 Evaluation

- ▶ utilisation et rédaction de programmes informatiques simples en langage *python* (version 3.X) pour la résolution numérique de problèmes de modélisation en lien avec le cours,
- ▶ utilisation de la distribution *Anaconda* proposée par la compagnie *Continuum Analytics* (free, multiplateforme, contenu riche, IDE conviviale...),
- ▶ 3 séances de 4H évaluées lors d'un examen sur table dédié,
- ▶ sujets de TP seront disponibles en ligne.

Python

Le langage de programmation *Python* s'avère particulièrement adapté au développement d'applications scientifiques/techniques simples car il s'agit d'un langage :

- ▶ de haut-niveau,
- ▶ interprété,
- ▶ multiparadigme,
- ▶ à la syntaxe simple,
- ▶ doté de nombreuses bibliothèques et simple à étendre,
- ▶ free,
- ▶ opensource,
- ▶ populaire et largement documenté...

Problème :

estimer l'altitude maximale atteinte par un boulet de canon projeté d'une position $(x_0, y_0) = (0, 0)$ à une vitesse de $v_0 = 100 \text{ m} \cdot \text{s}^{-1}$ selon un angle $\alpha = 30^\circ$ dans un champ de pesanteur d'intensité $g = 9.81 \text{ m} \cdot \text{s}^{-2}$

Modélisation mathématique :

L'application du principe fondamental de la dynamique au boulet de canon conduit à des équations différentielles découplées simples à résoudre.

Solution analytique de la trajectoire :

$$x(t) = v_0 \cdot \cos(\alpha) \cdot t + x_0$$

$$y(t) = -\frac{1}{2} \cdot g \cdot t^2 + v_0 \cdot \sin(\alpha) \cdot t + y_0$$

Programmation avec Python - Anaconda : exemple

```
import numpy          #bibliothèque pour les tableaux
import math          #bibliothèque de fct mathématiques
import matplotlib    #bibliothèque pour afficher des courbes

#=== paramètres physiques =====
m=10.                #masse
g=9.81               #accélération pesanteur
v0=100.              #vitesse initiale
alpha=30.            #angle initial en degrés
deltaT=0.1           #pas de temps
x0=0.                #position initiale
y0=0.                #position initiale

#=== paramètres numériques =====
nInstants=101       #nb d'instants où est calculée la trajectoire
```

Programmation avec Python - Anaconda : exemple

```
#=== initialisations =====
tab_t=numpy.zeros(nInstants)
tab_x=numpy.zeros(nInstants)
tab_y=numpy.zeros(nInstants)
alpha=alpha/180.*math.pi; #passage de l'angle en radian

#=== on remplit les tableaux =====
for i in range(nInstants):
    tab_t[i]=i*deltaT;
    tab_x[i]=v0*math.cos(alpha)*tab_t[i]+x0;
    tab_y[i]=-1./2.*g*tab_t[i]**2+v0*math.sin(alpha)*tab_t[i]+y0;

#=== affichage de la courbe x(t) =====
matplotlib.pyplot.plot(tab_x,tab_y)
```

Programmation avec Python - Anaconda : exemple

The image shows the Spyder Python IDE interface. On the left, a Python script is open, defining parameters for a projectile launch and plotting its trajectory. The script includes comments in French and uses NumPy and Matplotlib. A central window titled 'Figure 1' displays a plot of the trajectory, showing a parabolic path of a projectile over a distance of 900 meters. The vertical axis is labeled 'y (m)' and ranges from 0 to 140. The horizontal axis is labeled 'x (m)' and ranges from 0 to 900. On the right, the 'Explorateur de variables' (Variable Explorer) shows a table of variables defined in the script, including numerical values and arrays. Below the variable explorer, the 'Console' window shows the execution output, including file paths and warnings.

```
m=10. #masse
g=-9.81 #acceleration pesanteur
v0=100. #vitesse initiale
alpha=30. #angle initial en degres
deltaT=0.1 #pas de temps
x0=0. #position initiale
y0=0. #position initiale
16
17=== parametres numeriques =====
18nInstants=101 #nb d'instants ou est d
19
20=== initialisations=====
21alpha=alpha/180*math.pi; #passage de
22tab_t=numpy.zeros(nInstants)
23tab_x=numpy.zeros(nInstants)
24tab_y=numpy.zeros(nInstants)
25
26=== on remplit les tableaux =====
27for i in range(nInstants):
28    tab_t[i]=i*deltaT;
29    tab_x[i]=v0*math.cos(alpha)*tab_t
30    tab_y[i]=1./2.*g*tab_t[i]**2+v0*math.sin(alpha)*tab_t[i]+y0;
31
32=== affichage de la courbe x(t) =====
33matplotlib.pyplot.plot(tab_x,tab_y,label='y(x)',linewidth=3,color=[0,0,1])
34matplotlib.pyplot.xlabel('% (m)')
35matplotlib.pyplot.ylabel('% (m)')
36matplotlib.pyplot.title('Trajectoire du boulet de canon soumis a son poids')
37matplotlib.pyplot.grid(b=None, which='major', axis='both');
38matplotlib.pyplot.legend()
39matplotlib.pyplot.savefig("graphe.pdf")
40
```

Nom	Type	Taille	Valeur
alpha	float	1	0.5235987755982988
deltaT	float	1	0.1
g	float	1	-9.81
i	int	1	100
indiceColonne	int	1	0
j	int	1	999
m	float	1	10.0
matriceCompteur	float64 (3L 61L)		array([[0., 311., 55.
matriceProba	float64 (3L 3L)		array([[0.9, 0.3, 0.8],
n1	int	1	868
n2	int	1	80
n3	int	1	52
nEtudiants	int	1	1000
nInstants	int	1	101
nPeriodes	int	1	60

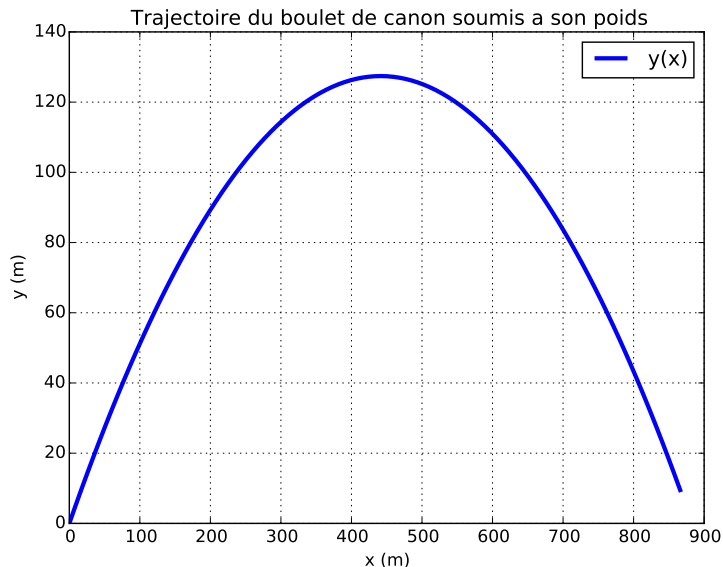
```
Console 486(A)
22/01/2015 22:40 <REP>
22/01/2015 22:40 <REP>
22/01/2015 22:40 1 064 trajectoire.py
1 fichier(s) 1 064 octets
2 Rép(s) 456 999 473 152 octets libres

In [24]:
runfile("D:/SCIENCE/ENS/1415/S2/UE222/COURS/1/PP/trajectoire.p
y", wdir="D:/SCIENCE/ENS/1415/S2/UE222/COURS/1/PP")
C:\Python34\AnacondaLib\site-
packages\matplotlib\axes\_axes.py:475: UserWarning: No
labelled objects found. Use labels'... kwarg on individual
plots.
warnings.warn("No labelled objects found. ")

In [25]:
runfile("D:/SCIENCE/ENS/1415/S2/UE222/COURS/1/PP/trajectoire.p
y", wdir="D:/SCIENCE/ENS/1415/S2/UE222/COURS/1/PP")

In [26]:
```

Programmation avec Python - Anaconda : exemple



Recherche de l'altitude maximale :

Quel code ajouter à la suite de l'existant pour identifier l'altitude maximale ?

```
...  
#=== on remplit les tableaux  
for i in range(nInstants):  
    tab_t[i]=i*deltaT;  
    tab_x[i]=v0*math.cos(alpha)*tab_t[i]+x0;  
    tab_y[i]=-1./2.*g*tab_t[i]**2+v0*math.sin(alpha)*tab_t[i]+y0;  
  
#=== affichage de la courbe y(x)  
matplotlib.pyplot.plot(tab_x,tab_y)
```

Recherche de l'altitude maximale :

Quel code ajouter à la suite de l'existant pour identifier l'altitude maximale ?

```
...  
#=== on remplit les tableaux  
for i in range(nInstants):  
    tab_t[i]=i*deltaT;  
    tab_x[i]=v0*math.cos(alpha)*tab_t[i]+x0;  
    tab_y[i]=-1./2.*g*tab_t[i]**2+v0*math.sin(alpha)*tab_t[i]+y0;  
  
#=== affichage de la courbe y(x)  
matplotlib.pyplot.plot(tab_x,tab_y)  
  
#=== recherche altitude max  
alti_max=y0  
for i in range(nInstants):  
    if(tab_y[i]>alti_max):  
        alti_max=tab_y[i]
```


- 1 Contexte de l'enseignement
- 2 Contenu des cours/TD
- 3 Travaux Pratiques
- 4 Evaluation**

Éléments

Trois éléments distincts sont considérés pour l'évaluation du module :

- ▶ un devoir surveillé (DS) : programme des premières séances de CM/TD,
- ▶ 3 séances de travaux pratiques et un examen sur table associé (TP)
- ▶ un examen final (F) : programme de l'ensemble des CM, TD et TP.

Pondération

A titre indicatif la note finale N l'an passé était donnée par :

$$N = 0.75 \times \text{Max}(F, 0.4 \times DS + 0.6 \times F) + 0.25 \times TP$$