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

Contexte de l'enseignement

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, ...).

Contenu des cours/TD

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

Travaux Pratiques

- 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.

Travaux Pratiques

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 librairies et simple à étendre,
- free,
- opensource,
- populaire et largement documenté...

Programmation avec Python - Anaconda : exemple

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 $v_0 = 100\ m \cdot s^{-1}$ selon un angle $\alpha = 30^o$ dans un champ de pesanteur d'intensité $g = 9.81\ m \cdot s^{-2}$

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 :

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

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

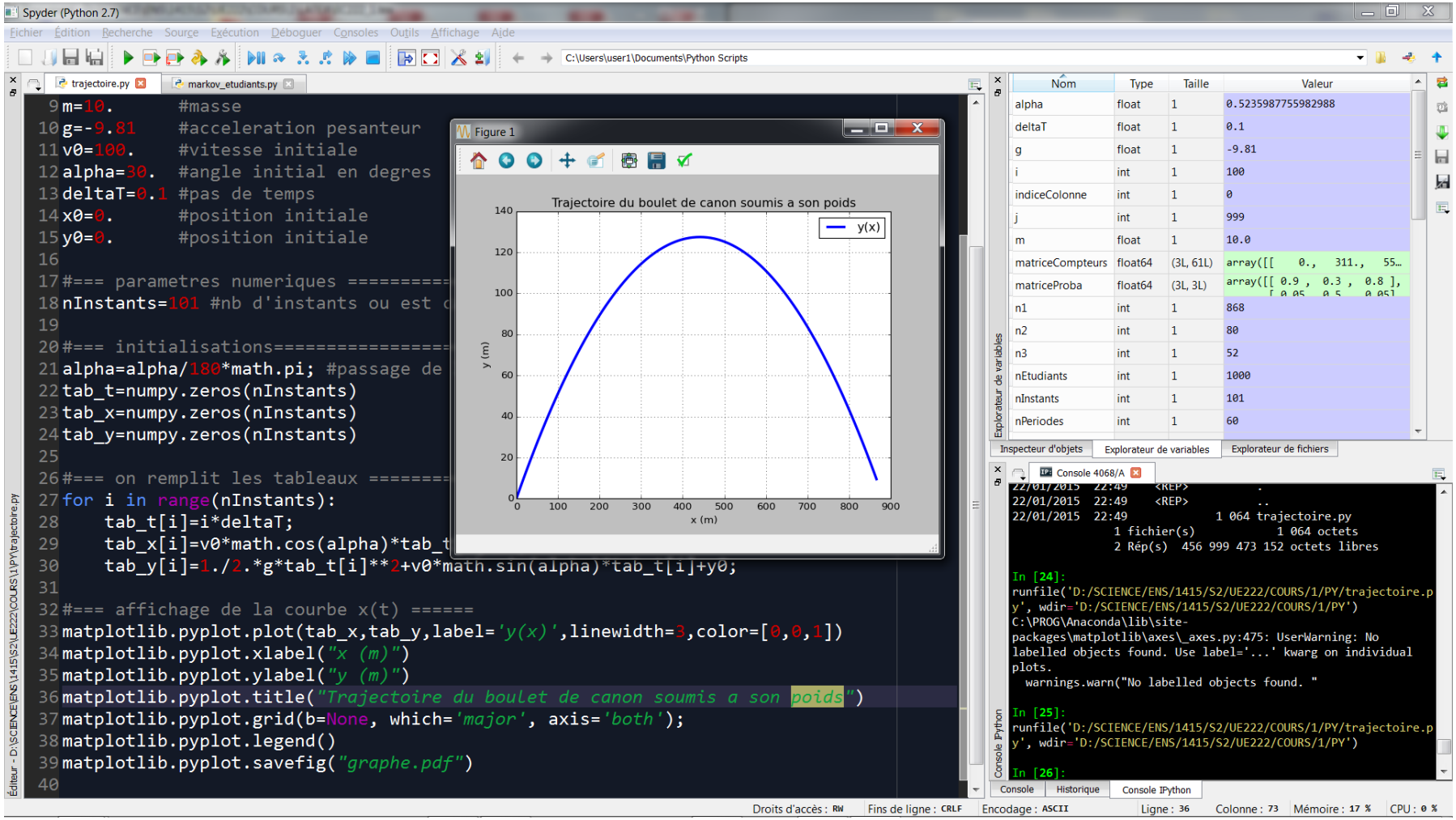
Programmation avec Python - Anaconda : exemple

```
import numpy      #librairie pour les tableaux
import math       #librairie de fct mathematiques
import matplotlib #librairie pour afficher des courbes

#=== parametres physiques =============
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

#=== parametres numeriques ============
nInstants=101 #nb d'instants ou est calculee la trajectoire
```

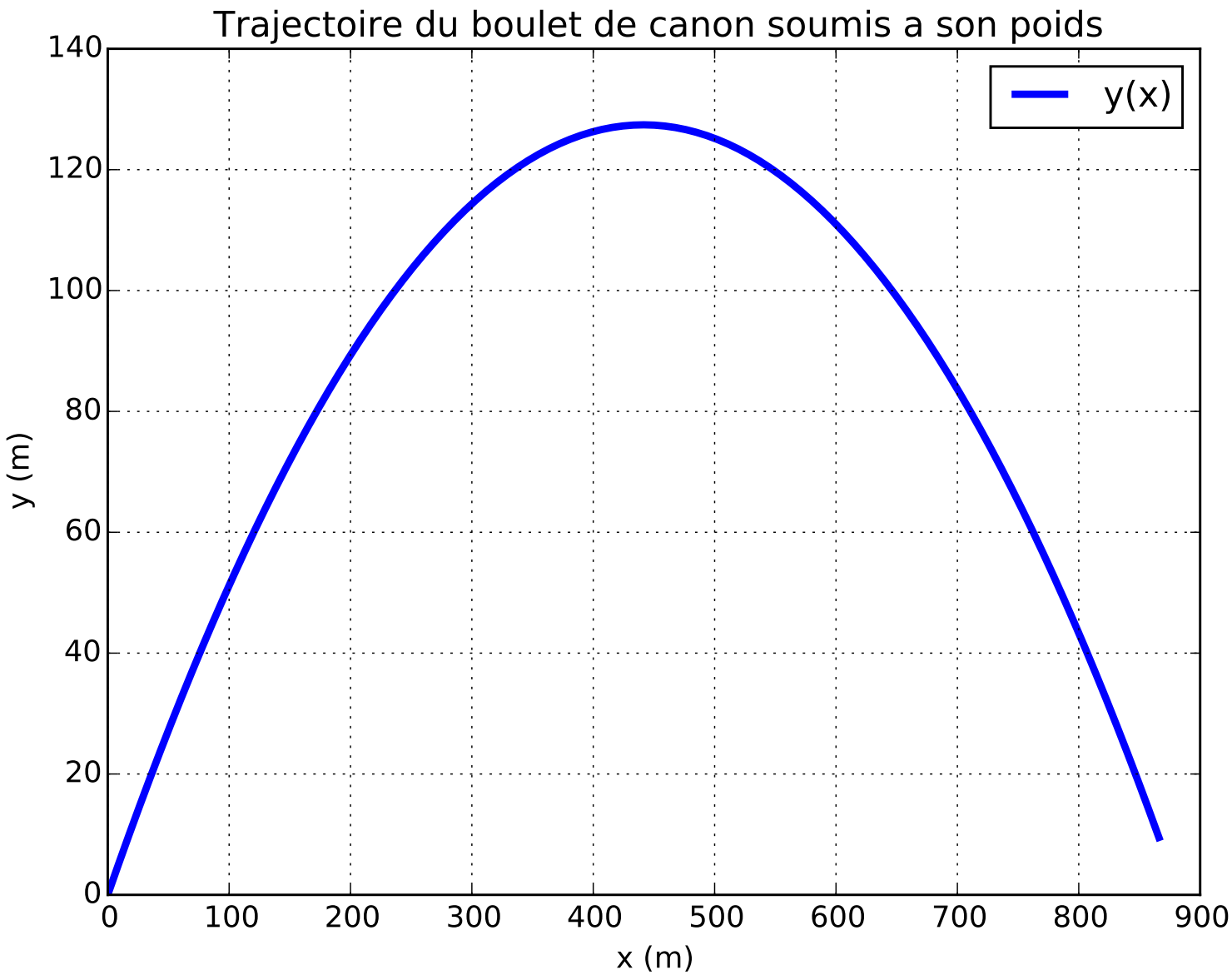
Programmation avec Python - Anaconda : exemple

```
#=== 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



Programmation avec Python - Anaconda : exemple



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)
```

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 altitude max
alti_max=y0
for i in range(nInstants):
   if(tab_y[i]>alti_max):
       alti_max=tab_y[i]
```

Evaluation

Eléments

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 :

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