

PROGRAMMER EN PYTHON FICHE N°5 : ANALYSE D'UNE CHUTE 2D

L'étude du mouvement d'une balle de golf, à partir d'une chronophotographie, permet ici de tracer les vecteurs vitesse au cours du temps. Les outils présentés dans les activités précédentes sont exploités à nouveau. Les positions relevées toutes les 0,066 s sont rassemblées dans un tableau.

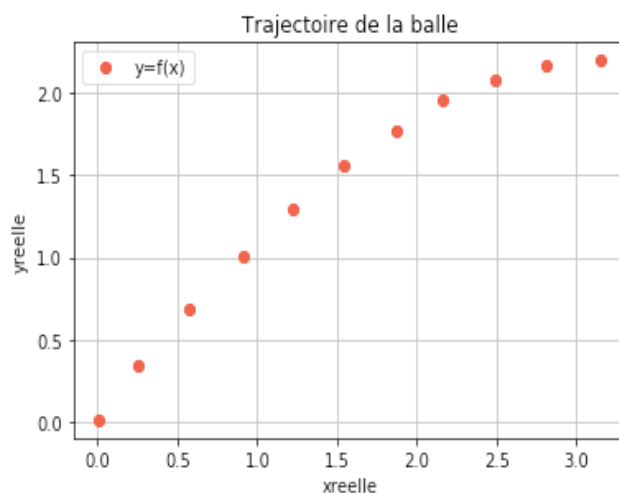
Capacité numérique mise en œuvre : représenter des vecteurs vitesse d'un système modélisé par un point lors d'un mouvement à l'aide d'un langage de programmation.

```
import numpy as np  
import matplotlib.pyplot as plt
```

Représentation de la trajectoire

Trois tableaux rassemblant les temps successifs (intervalle de temps entre deux photos de 0,066 s), les abscisses et les ordonnées sont d'abord créés. Ensuite, la trajectoire est obtenue en traçant l'ordonnée en fonction de l'abscisse.

```
t=0.066*np.arange(11)
x=np.array([0.01,0.25,0.57,0.91,1.22,1.54,1.87,2.16,2.49,2.81,3.15])
y=np.array([0.01549,0.3404,0.6812,1.010,1.297,1.559,1.768,1.95,2.08,2.158,2.193])
plt.plot(x,y,'ro',label="y=f(x)")
plt.xlabel("xreelle")
plt.ylabel("yreelle")
plt.grid()
plt.legend()
plt.title("Trajectoire de la balle")
plt.show()
```



Retrouvez éducol sur :

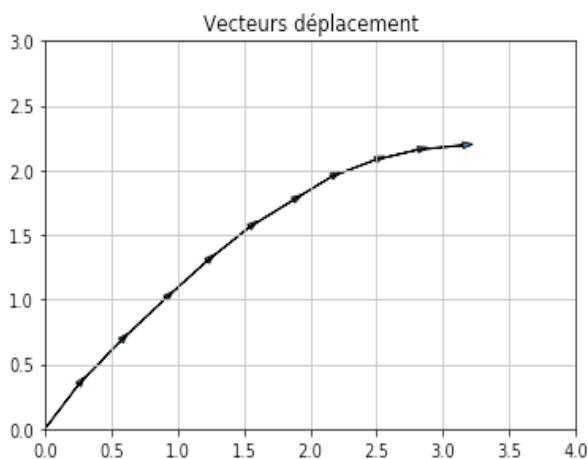


Représentation des vecteurs « déplacement »

Comme précédemment, la boucle for permet de faire défiler les positions successives (termes de yreelle). Le tracé des vecteurs « déplacement » s'effectue avec la fonction plt.arrow. Comme précédemment, il faut prendre garde à redimensionner la liste, pour qu'elle contienne un élément de moins que les listes des positions (on ne peut pas calculer le terme $x[i+1]$ et $y[i+1]$ pour la dernière valeur de i). La commande len(t) indique le nombre de valeurs dans la liste t.

```
n=np.arange(len(t)-1)
for i in n :
    plt.arrow(x[i],y[i],x[i+1]-x[i],y[i+1]-y[i],head_width=0.05)
plt.xlim(0,4)
plt.ylim(0,3)
plt.grid()

plt.title("Vecteurs déplacement")
plt.show()
```



Conclusion : le code précédemment utilisé pour l'étude de la chute libre verticale conduit à un résultat plus lisible.

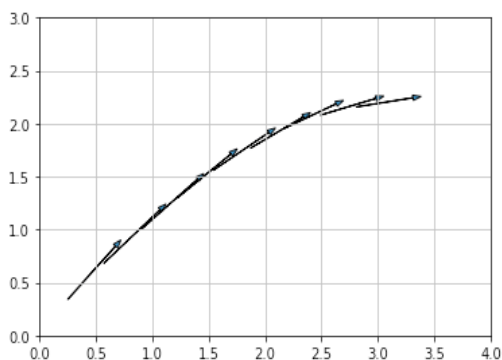
Retrouvez éduscol sur :



Représentation des vecteurs « vitesse »

À la différence du script précédent, le vecteur vitesse à la position « $i+1$ » est ici obtenu en calculant le déplacement entre les positions i et $i+2$ et en le divisant par l'intervalle de temps associé. Pour cela, la liste des positions « i » doit à nouveau être redimensionnée :

```
m=np.arange(len(t)-2)
for i in m :
    plt.arrow(x[i+1],y[i+1],0.1*(x[i+2]-x[i])/(t[i+2]-t[i]),0.1*(y[i+2]-y[i])/(t[i+2]-t[i]),head_width=0.05)
plt.xlim(0,4)
plt.ylim(0,3)
plt.grid()
```



Conclusion

On notera que si le résultat est plus lisible, il faut tout de même réfléchir à l'échelle de représentation du vecteur vitesse sur le graphe – c'est la raison du facteur « 0.1 » qui a été placé devant les vecteurs.

Retrouvez éduscol sur :

