Langage Python A3 TD5

Exercice 1 : numpy et matplotlib

C'est une mini introduction de numpy et matplotlib,
Essayez à votre manière de découvrire ces deux modules (ne pas oublier dir, pour le module et pour les classes et méthodes)
Importer numpy et matplotlib de la façon suivante:

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

Tester le code suivant :

```
In [18]: a=np.arange(10)

In [19]: a
Out[19]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [20]: print(a)
[0 1 2 3 4 5 6 7 8 9]

In [21]: type(a)
Out[21]: numpy.ndarray

In [22]: L=np.arange(2,15,3)

In [23]: print(L)
[ 2  5  8 11 14]

In [24]: p1=np.array([1,2,3])

In [25]: p1
Out[25]: array([1, 2, 3])

In [26]: print(p1)
[1 2 3]

In [27]: type(p1)
Out[27]: numpy.ndarray

In [28]: l2=[10,12,13]

In [29]: l1=[1,2,3]

In [30]: l1+l2
Out[30]: [1, 2, 3, 10, 12, 13]

In [31]: p2=np.array(l2)

In [32]: p1+p2
Out[32]: array([11, 14, 16])
```

Accordez une importance à p1+p2, pour les listes on aurait dû faire une boucle pour additionner les éléments 1 à 1. Il s'agit d'une vectorisation en numpy.

la fonction arange :

Syntaxe : **numpy.arange(***start*, *stop*, *step*, *dtype=None***)**

La fonction arange(a,b,h) du module numpy renvoie le tableau des réels a+k*h compris dans [a; b[(où k est ici un entier et où le h représente le pas)

Accédez à la documentation de arange (help(np.arange) ou tout simplement (np.arange?)

La fonction linspace :

Syntaxe :

numpy.linspace(*start*, *stop*, *num=50*, *endpoint=True*, *retstep=False*, *dtype=None*, *axis=0***)**

la fonction linspace(a, b, n) retourne une liste de n (*n=50 par défaut)* éléments régulièrement espacés, calculés sur l'intervalle [a, b] (*endpoint = True* par défaut donc b est inclus).

reststep est le pas *h* entre chaque point de la liste.

Exemple :

```
In [45]: x1 = np.linspace(0, 10, N, endpoint=True)

In [46]: print(x1)
[ 0.          1.42857143  2.85714286  4.28571429  5.71428571  7.14285714
  8.57142857 10.        ]

In [47]: x2 = np.linspace(0, 10, N, endpoint=False)

In [48]: print(x2)
[0.   1.25 2.5  3.75 5.   6.25 7.5  8.75]
```

Exemple de tracé:

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

plt.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Matplotlib plot.
plt.show()
```

Astuce: *Pour afficher le plot dans une fenêtre séparée sous spyder :*
Tapez ***%matplotlib auto*** *dans l'interpréteur python, pour rechanger taper* ***%matplotlib inline***

Exercice 2 :

Sans utiliser numpy, **Tracez sin(x) x entre 0 et 2*pi**

Exercice 3 :

Tracez sin(x) x entre 0 et 2*pi :

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

x=np.linspace(0,2*np.pi,100)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```

Retracez sin(x) x entre 0 et 2*pi avec un pas de 0.05

Exercice 4 : Masse-Ressort

Soit A et B les extrémités d'un ressort mis en parallèle avec un amortisseur. A est fixe de position p0 (0,0,0) . B est libre de position initiale (0,-1,0). A l'extrémité B est associé une masse m. nous ne prenons pas en compte la force de la pesanteur. La raideur du ressort est raideur, l'amortisseur a un coefficient amort.

Nous imposons un déplacement 0.1 selon l'axe y du point B, p1 sera égal à p1(0,-1.1,0), puis nous le relâchons. Écrivez un programme qui permet de tracer l'allongement en fonction du temps, pour un temps de simulation time_sim et un deltat d'intégration. Utilisez les valeurs d'initialisation suivantes

```
#p0 est la position initiale de l'extrémité fixe du ressort
p0=np.array([0.,0.,0.])
#p1 est la position au repos de l'extrémité libre du ressort
p1=np.array([0.,-1.,0.])
#m est la masse de l'extrémité libre
m=1.0
#k est la raideur du ressort
k=0.1
#amort est l'amortissement du ressort
amort=0.1
#deltat est le pas d'integration
deltat=1.0
```

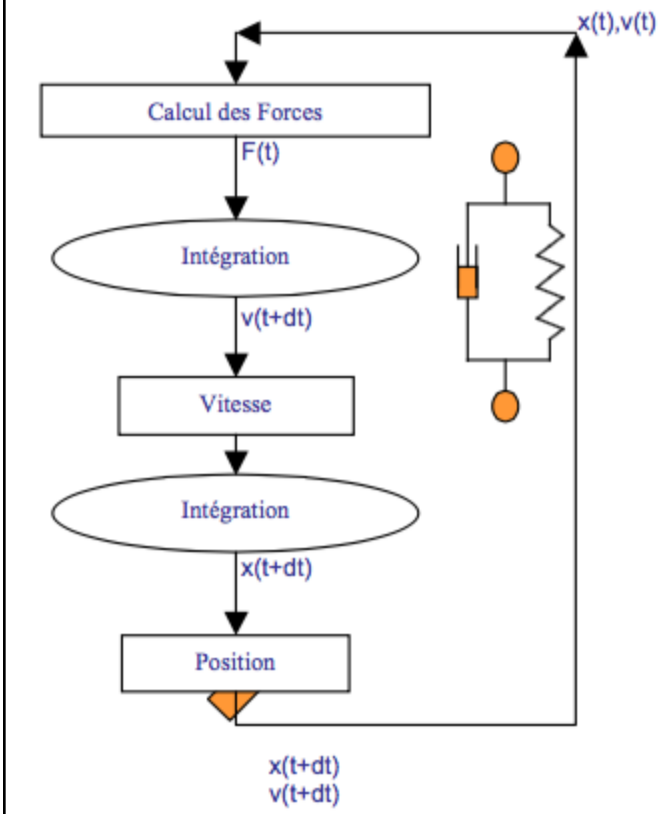
#time_sim est le temps de la simulation
time_sim=50

Amusez vous en changeant certaines valeurs initiales.

Rappel :loi fondamentale de la dynamique

m.gamma=somme des force

m.gamma=-amort*v-k*deltaLong*BA/norm(BA) où BA est un vecteur

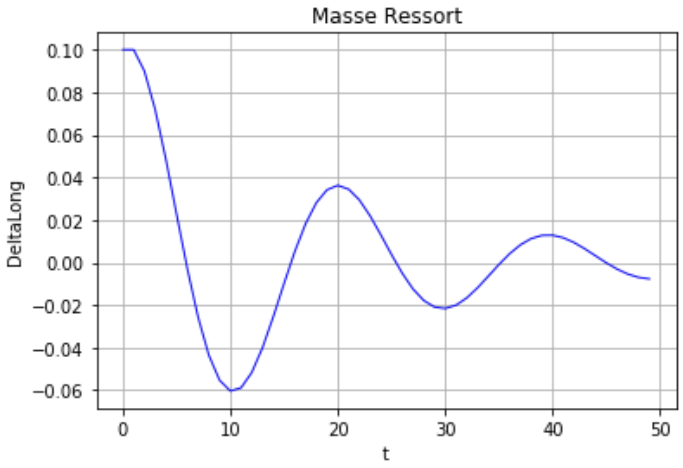


Intégration Euler semi implicite :

$$v_{t+dt} = v_t + \frac{F_t}{m}dt$$

$$x_{t+dt} = x_t + v_{t+dt}dt$$

Exemple resultat:



Exercice 5 :

Refaites l'exercice 4 en paramétrant la masse, la raideur, l'amortisseur, deltat et time_sim en utilisant des sliders. Inspirer vous de la démonstration Slider demo publiée sur :

https://matplotlib.org/stable/gallery/index.html