orbites

April 21, 2020

1 Calcul et représentation d'orbites de vecteurs par une matrice

[1]:
```
import numpy as np

import numpy.linalg as alg

import matplotlib.pyplot as plt
```

1.1 Etude d'un système proies-prédateurs

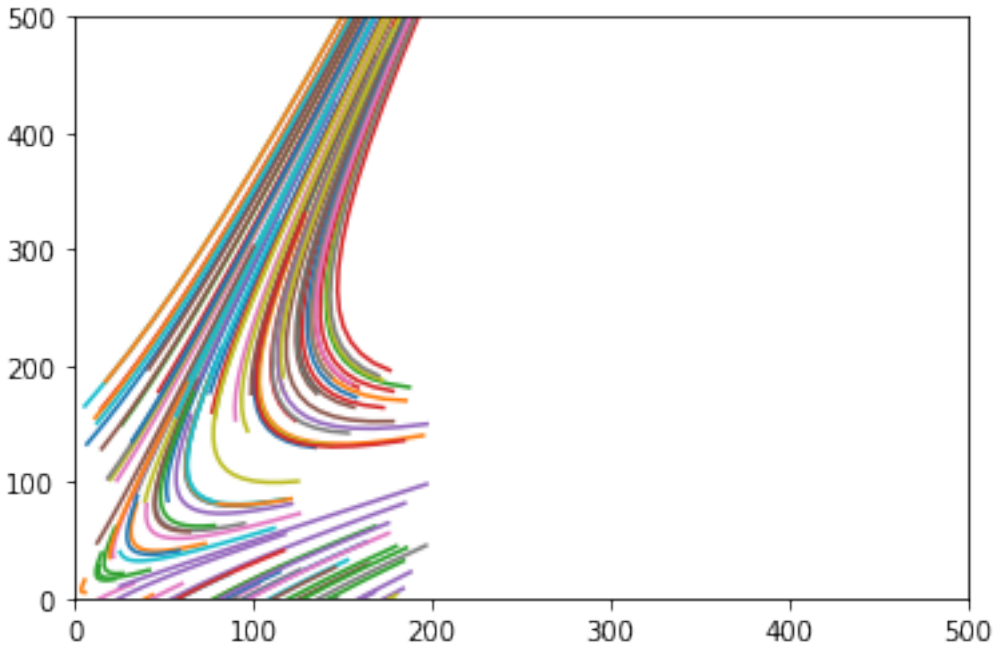
[2]:
```
A= np.array([[0.86,0.08],[-0.12,1.14]]) # matrice de type proies-prédateurs␣
↪géocoucous
#et coyotes
A
```

[2]:
```
array([[ 0.86,  0.08],
       [-0.12,  1.14]])
```

[3]:
```
n=20
```

[4]:
```
def orbite(A,V,n):
    X=np.zeros((n,2))
    X[0]=V
    for i in np.arange(1,n):
        X[i]=A.dot(X[i-1])
    return X
```

[5]:
```
for i in np.arange(100):
    V=200*np.random.rand(2)
    X=orbite(A,V,n)
    plt.plot(X[:,0],X[:,1], '-')
    plt.axis([0,500,0,500])#plt.axis('equal')
    #plt.ylim(-10,10)
```



1.1.1 Valeurs propres et vecteurs propres de A

[6]:
```
alg.eig(A)
```

[6]:
```
(array([0.9, 1.1]),
 array([[-0.89442719, -0.31622777],
        [-0.4472136 , -0.9486833 ]]))
```

[7]:
```
alg.eig(A)[1][1,:]/alg.eig(A)[1][0,:] # calcul de la pente des vecteurs propres
```

[7]:
```
array([0.5, 3. ])
```

1.2 Evolution du couple glucose-insuline dans le sang

[8]:
```
A=np.array([[0.9,-0.4],[0.1,0.9]]) # équilibre glucose-insuline
A
```

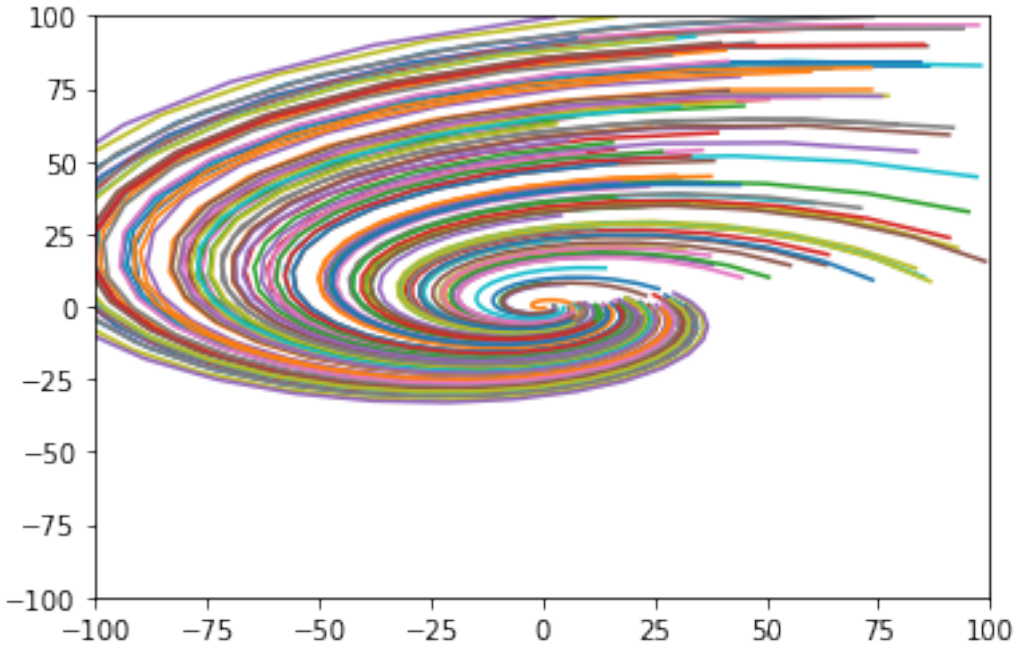
[8]:
```
array([[ 0.9, -0.4],
       [ 0.1,  0.9]])
```

[9]:
```
n=24 # évolution sur 24 heures
```

[10]:
```
def orbite(A,V,n):
    X=np.zeros((n,2))
    X[0]=V
```

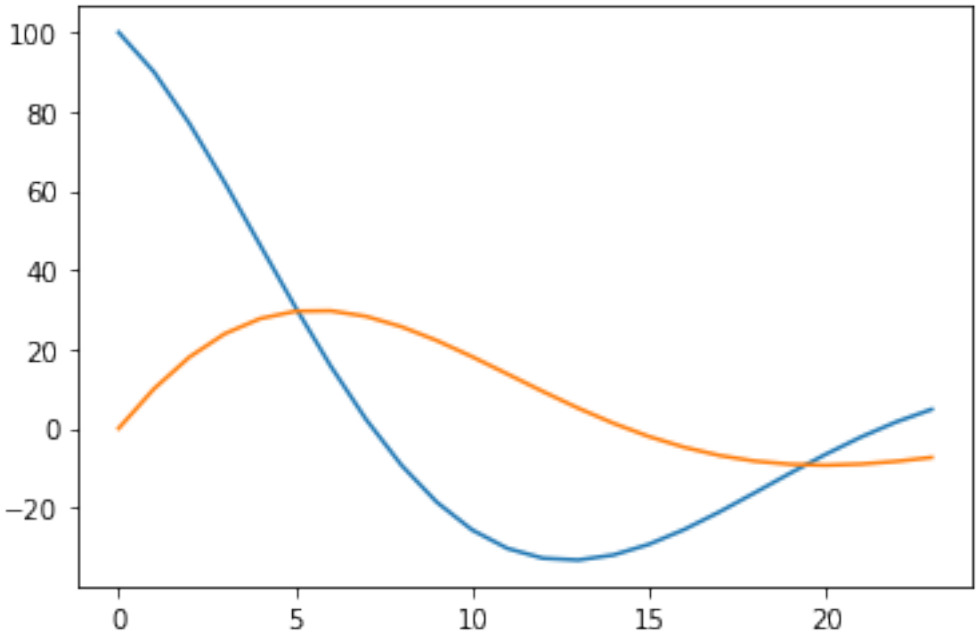
```
    for i in np.arange(1,n):
        X[i]=A.dot(X[i-1])
    return X
```

[11]:
```
for i in np.arange(100):
    V=100*np.random.rand(2)
    X=orbite(A,V,n)
    plt.plot(X[:,0],X[:,1], '-')
    plt.axis([-100,100,-100,100])#plt.axis('equal')
    #plt.ylim(-10,10)
```



1.2.1 Courbes des taux de gluscose (bleu) et d'insuline (rouge) en partant de (100,0).

[12]:
```
X=orbite(A,np.array([100,0]),n)
plt.plot(X, '-')
plt.show()
```



1.2.2 Valeurs propres de A

[13]:
```
alg.eig(A) # valeurs propres complexes et vecteurs propres
```

[13]:
```
(array([0.9+0.2j, 0.9-0.2j]),
 array([[0.89442719+0.j        , 0.89442719-0.j        ],
        [0.        -0.4472136j, 0.        +0.4472136j]]))
```

[14]:
```
abs(alg.eig(A)[0]) # module des valeurs propres
```

[14]:
```
array([0.92195445, 0.92195445])
```