effectifsL1L2

April 2, 2020

1 Evolution des effectifs en L1-L2

[1]:
```
import numpy as np # bibliothèques chargées : numpy

import numpy.linalg as alg # utile en algèbre linéaire

import matplotlib.pyplot as plt # pour les graphiques
```

1.1 Le problème à étudier

Effectifs l'année n : $x_1(n)$ en L1 et $x_2(n)$ en L2.

On a $x_1(n+1) = \frac{1}{4}x_1(n) + 90$ et $x_2(n) = \frac{1}{2}x_1(n) + \frac{1}{4}x_2(n)$.

Vecteur effectifs l'année n : $X(n) = (x_1(n), x_2(n)) \in \mathbb{R}^2$.

Relation de récurrence linéaire $X(n+1) = MX(n) + E$ avec

$$M = \begin{pmatrix} 1/4 & 0 \\ 1/2 & 1/4 \end{pmatrix} \quad \text{et } E = \begin{pmatrix} 90 \\ 0 \end{pmatrix}.$$

1.1.1 Initialisation des matrices en Python

[2]:
```
M= np.array([[1/4,0],[1/2,1/4]])
M
```

[2]:
```
array([[0.25, 0.  ],
       [0.5 , 0.25]])
```

[3]:
```
E=np.array([90,0]) # vecteur des entrées dans le cycle d'étude
E
```

[3]:
```
array([90,  0])
```

Exemple avec $X(0) = (50, 50)$.

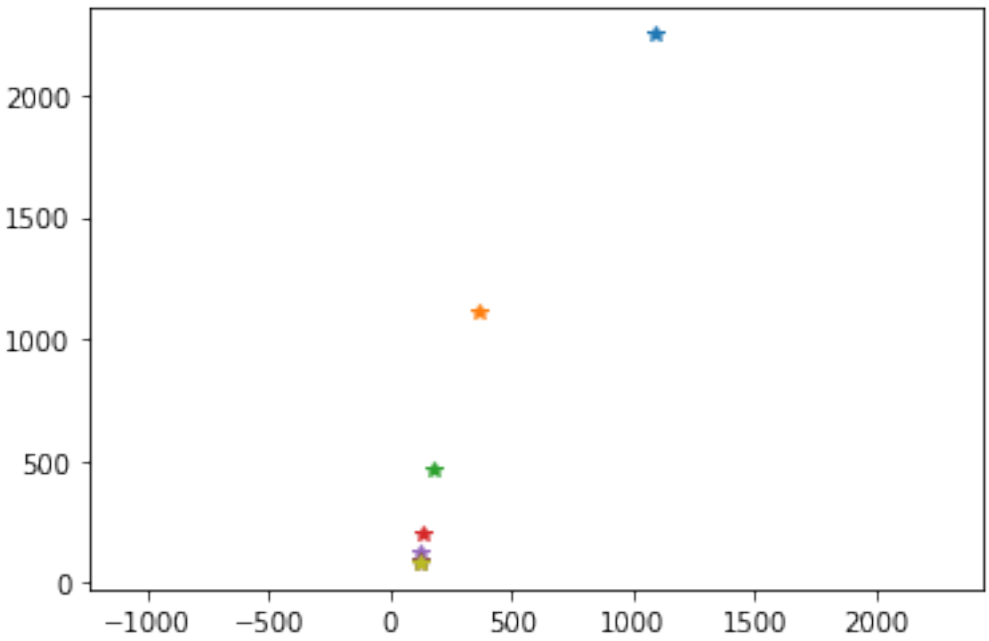
[4]:
```
X0=np.array([50,50])
X1= M.dot(X0)+ E
X1
```

[4]: array([102.5, 37.5])

1.1.2 On effectue une boucle pour calculer les effectifs par récurrence.

[5]:
```
X=np.array([4000,1000]) # condition initiale
print(X)
for i in np.arange(9):
    X= M.dot(X) + E
    print(X)
    plt.plot(X[0],X[1],'*')
    plt.axis('equal')
```

```
[4000 1000]
[1090. 2250.]
[ 362.5 1107.5]
[180.625 458.125]
[135.15625 204.84375]
[123.7890625 118.7890625]
[120.94726562  91.59179688]
[120.23681641  83.37158203]
[120.0592041   80.96130371]
[120.01480103  80.26992798]
```



1.1.3 Départ à partir d'un effectif aléatoire de 0 à 200 en L1 L2

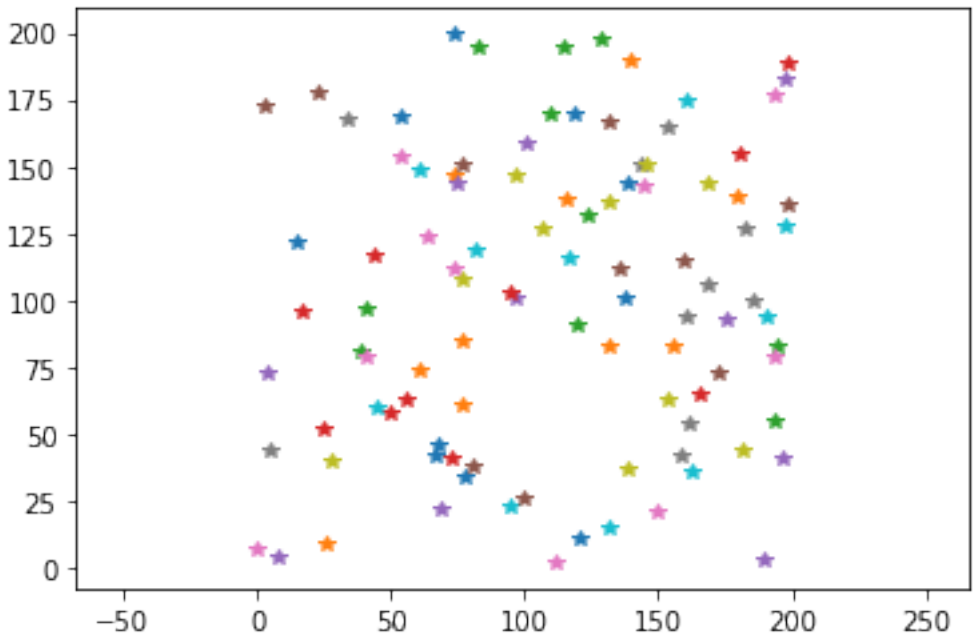
[6]:
```
V=200*np.random.rand(2) # on multiplie par 200 le vecteur aléatoire de
↪[0,1]x[0,1]
```

[7]:
```
V
```

[7]:
```
array([138.07117588,  13.19229419])
```

Et hop, une centaine de vecteurs aléatoires!

[8]:
```
for i in np.arange(100):
    X=200*np.random.rand(2)
    plt.plot(X[0],X[1], '*')
    plt.axis('equal')
```



On définit une fonction qui renvoie la suite des $X(n)$ (on dit l'orbite ou la trajectoire de $X(0)$).

[9]:
```
def orbite(V,n): # V est la donnée initiale et n le nombre de vecteurs dessinés.
    X=np.zeros((n,2)) # on va remplir une matrice nx2 ligne à ligne avec les
↪X(k)
    X[0]=V # on initialise la première ligne de X avec la donnée initiale
    for i in np.arange(1,n):
        X[i]=M.dot(X[i-1]) + E # la récurrence
    return X
```

Autre méthode em emplilant les résultats.

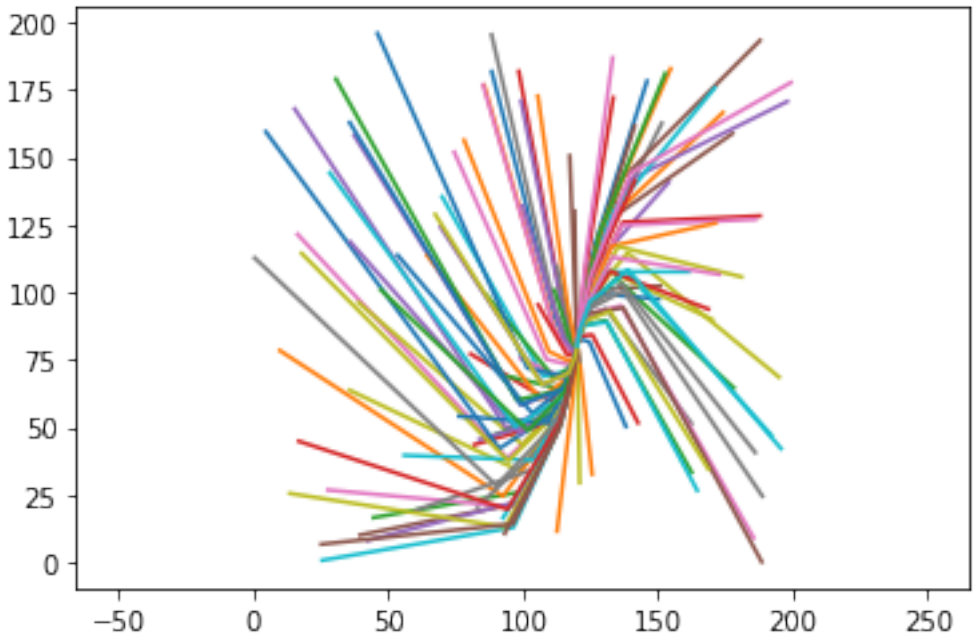
[10]:
```
def orbite(V,n):  # V est la donnée initiale et n le nombre de vecteurs␣
↪dessinés.
    X=V # on va empliler les résultats dans X
    for i in np.arange(1,n):
        V =M.dot(V) + E  # la récurrence
        X=np.vstack((X,V))# on empile verticalement
    return X
```

Avec cela, on dessine des tas d'orbites aléatoires à la fois!

[11]:
```
orbite(V,10)
```

[11]:
```
array([[138.07117588,  13.19229419],
       [124.51779397,  72.33366149],
       [121.12944849,  80.34231236],
       [120.28236212,  80.65030234],
       [120.07059053,  80.30375665],
       [120.01764763,  80.11123443],
       [120.00441191,  80.03663242],
       [120.00110298,  80.01136406],
       [120.00027574,  80.0033925 ],
       [120.00006894,  80.000986  ]])
```

[12]:
```
for i in np.arange(100):  # 100 orbites ici!
    V=200*np.random.rand(2)
    plt.plot(orbite(V,20)[:,0],orbite(V,20)[:,1], '-')
    plt.axis('equal')
```



1.1.4 Tada !

1.2 Calcul des effectifs limites

Les effectifs limites doivent satisfaire $X = MX + E$ c'est-à-dire $(Id - M)X = E$.

[13]:
```
N= np.eye(2) - M # np.eye(2) fabrique la matrice identité de taille 2x2
N
```

[13]:
```
array([[ 0.75,  0.  ],
       [-0.5 ,  0.75]])
```

[14]:
```
alg.matrix_rank(N) # on teste si N est inversible
```

[14]:
```
2
```

[15]:
```
X = alg.solve(N,E) # calcul des effectifs stables.
X
```

[15]:
```
array([120.,  80.])
```

1.2.1 Retour sur la première fonction orbite utilisée

[16]:
```
n=10
```

[17]:
```
X=np.zeros((n,2))
X
```

[17]:
```
array([[0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.],
       [0., 0.]])
```

[18]:
```
X[0]=V
X
```

[18]:
```
array([[195.57103656,  42.10045498],
       [  0.        ,   0.        ],
       [  0.        ,   0.        ],
       [  0.        ,   0.        ],
```

```
       [  0.        ,   0.        ],
       [  0.        ,   0.        ],
       [  0.        ,   0.        ],
       [  0.        ,   0.        ],
       [  0.        ,   0.        ],
       [  0.        ,   0.        ]])
```

[19]:
```
for i in np.arange(1,10):
    X[i]=M.dot(X[i-1]) + E
```

[20]:
```
X
```

[20]:
```
array([[195.57103656,  42.10045498],
       [138.89275914, 108.31063203],
       [124.72318979,  96.52403758],
       [121.18079745,  86.49260429],
       [120.29519936,  82.21354979],
       [120.07379984,  80.70098713],
       [120.01844996,  80.2121467 ],
       [120.00461249,  80.06226166],
       [120.00115312,  80.01787166],
       [120.00028828,  80.00504448]])
```