Séance 06 :

TP Circularité

1. Présentation

C.F. sujet.

2. Importation des données

1. ***En vous inspirant du document*** `Fichiers_csv.pdf` ***importer dans deux listes*** `X` ***et*** `Y` ***les données issues de la MMT. On prendra soin de ne laisser dans ces listes que les données, c'est-à-dire que les titres seront enlevés.***

   ❏ On commence comme toujours :

   ```
   1 import matplotlib.pylab as plt
   2 import numpy as np
   3 import os
   ```

   ❏ On crée un dossier `ressources` dans lequel on place le fichier `fichier_point.csv`

   ❏ On ouvre et lit ce fichier comme demandé :

   ```
   1  def openCSV(path):
   2
   3      dirname = os.path.dirname(__file__)
   4      filename = os.path.join(dirname, path)
   5
   6      f = open(path, 'r')
   7      X = []
   8      Y = []
   9      for line in f.readlines():
   10         try:
   11             a, b = line.split(',')
   12             a = float(a)
   13             b = float(b)
   14             X.append(a)
   15             Y.append(b)
   16         except:
   17             pass
   18     return (X, Y)
   ```

   Notons que tous les lignes contenant des chaînes de caractères vont faire échouer les lignes 8 ou 9, et donc seront passées.

2. ***Dans la mesure où on veut utiliser*** `Numpy` ***pour travailler les tableaux, transformer ces listes en tableau*** `numpy.ndarray`***.***

   ```
   1 def array(doubleTable):
   2     return (np.array(doubleTable[0]), np.array(doubleTable[1]))
   ```

3. Méthode des moindres carrés

3. ***Ecrire une fonction*** `S(X, Y)` ***qui prend en argument les données issues de la MMT et qui retourne un vecteur*** `Som` ***dans lequel seront rangées les valeurs :***

   `Sxi` `Sxiyi2` `Syi` `Sxi2yi` `Sxiyi`

   `Sxi3` `Sxi2` `Syi3` `Syi2` `n`

```python
def S(X, Y):
    Sxi = np.sum(X)
    Syi = np.sum(Y)
    Sxiyi = np.dot(X, Y)
    Sxi2 = np.sum(X**2)
    Syi2 = np.sum(Y**2)
    Sxiyi2 = np.dot(X, Y**2)
    Sxi2yi = np.dot(X**2, Y)
    Sxi3 = np.sum(X**3)
    Syi3 = np.sum(Y**3)
    n = len(X)
    return(Sxi, Syi, Sxiyi, Sxi2, Syi2, Sxiyi2, Sxi2yi, Sxi3, Syi3, n)
```

4. ***Ecrire une fonction*** `cercle(X, Y)` ***qui prend en argument les données issues de la MMT et qui retourne*** `Xc` ***et*** `Yc` ***les coordonnées du centre du cercle et*** `R` ***le rayon du cercle passant au mieux.***

```python
def cercle(X, Y):
    Sxi, Syi, Sxiyi, Sxi2, Syi2, Sxiyi2, Sxi2yi, Sxi3, Syi3, n = S(X, Y)

    a = Sxi2 - ((Sxi)**2) / n
    b = Sxiyi - (Sxi * Syi) / n
    c = Syi2 - (Syi)**2 / n
    d = (1/2)*(Sxi3+Sxiyi2 - (1/n)*(Sxi*(Sxi2 + Syi2)))
    e = (1/2)*(Syi3+Sxi2yi - (1/n)*(Syi*(Sxi2 + Syi2)))

    xc = (c*d-b*e)/(a*c-b**2)
    yc = (a*e-b*d)/(a*c-b**2)

    R = np.sqrt((1/n)*(Sxi2+Syi2-2*(xc*Sxi+yc*Syi))+xc**2+yc**2)

    return (xc, yc, R, n)
```

5. ***Compléter le script de manière à afficher sur une même figure la position des points (cercle rouge) mesurés ainsi que le cercle qui passe au mieux (défini par la méthode des moindres carrés). Le rayon moyen sera affiché sur la figure.***

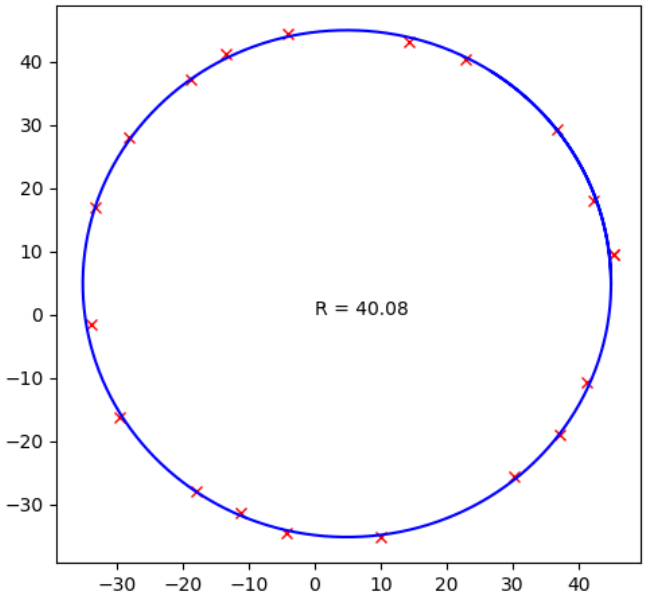
   ❏ On crée d'abord la fonction `traceCercle` permettant de créer une liste de coordonnées cartésiennes correspondant à celles d'un cercle :

```python
def traceCercle(Xc, Yc, R, n):
    X, Y = [], []
    h = (2*np.pi+1)/n
    for i in range(0, n):
        X.append(R * np.cos(i*h) + Xc)
        Y.append(R * np.sin(i*h) + Yc)
    return (X, Y)
```

   ❏ Puis on exécute le script suivant pour afficher ce qui est demandé :

```python
X, Y = array(openCSV('ressources/fichier_point.csv'))
Xc, Yc, R, n = cercle(X, Y)
Xth, Yth = traceCercle(Xc, Yc, R, 1000)
plt.plot(X, Y, 'rx')
plt.plot(Xth, Yth, 'b')
plt.text(0, 0, f'R = {R:.2f}')
plt.show()
```

   ❏ On obtient alors la figure suivante :



6. ***Ecrire une fonction*** `circularite(Xc, Yc, R, X, Y)` ***prenant en argumant les paramètres du cercle 'qui passe au mieux' ainsi que les deux tableaux de coordonnées des points mesurés et qui retourne*** `dc`***, le dédaut de circularité constaté ainsi que les distances des points les plus éloignés (*** `Emax` ***extérieurement, et*** `Emin` ***intérieurement) au cercle*** C***.***

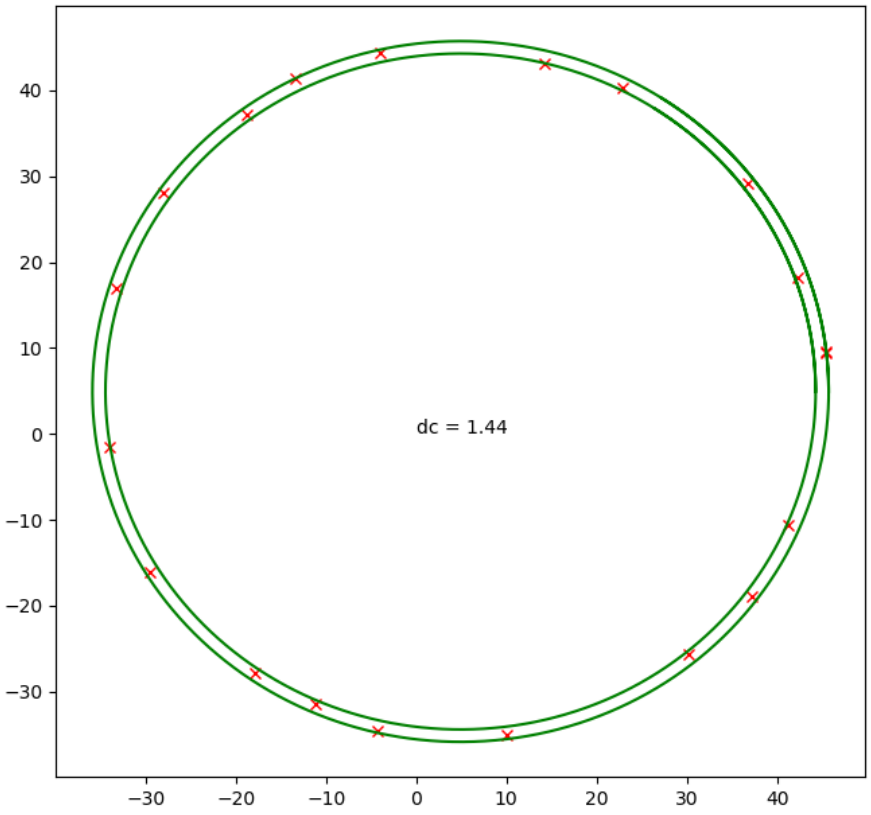
```python
def circularite(Xc, Yc, R, X, Y):
    # On calcule la norme de CM_i
    R_mes = np.sqrt((Xc-X)**2 + (Yc-Y)**2)

    E_min = R - np.min(R_mes)
    E_max = np.max(R_mes) - R

    dc = E_max + E_min
    return(E_max, E_min, dc)
```

7. ***Faire afficher sur le graphe précédent :***
   - C_{max} ***concentrique à*** C ***de diamètre le plus petit qui englobe tous les points*** M_i
   - C_{min} ***concentrique à*** C ***de diamètre le plus grand qui ne contient aucun point*** M_i

❏ On modifie le script précédent pour rajouter les cercles demandés :

```python
X, Y = array(openCSV('ressources/fichier_point.csv'))
Xc, Yc, R, n = cercle(X, Y)
E_max, E_min, dc = circularite(Xc, Yc, R, X, Y)
Xmax, Ymax = traceCercle(Xc, Yc, R + E_max, 1000)
Xmin, Ymin = traceCercle(Xc, Yc, R - E_min, 1000)
plt.plot(X, Y, 'rx')
plt.plot(Xmax, Ymax, 'g')
plt.plot(Xmin, Ymin, 'g')
plt.text(0, 0, f'dc = {dc:.2f}')
plt.show()
```

❏ Et on obtient la figure suivante :


dc = 1.44
40
30
20
10
0
−10
−20
−30
−30
−20
−10
0
10
20
30
40