École **S**upérieure **Pr**ivée d'**I**ngénierie et de **T**echnologies

Correction TP n°2 :
Interpolation et Approximation polynomiale

Module : Analyse numérique Classes: 3A-B

1 Objectif du TP

L'objectif du TP est de :

- Interpoler un nuage de points donné par un polynôme en utilisant la méthode d'interpolation de Lagrange.

- Approximer un nuage de points donné par un polynôme d'approximation au sens des moindres carrées.

- Représenter et interpréter graphiquement les résultats obtenus.

Afin d'implémenter les méthodes d'interpolation et d'approximation polynomiale, on aura besoin des bibliothèques `numpy`, `matplotlib.pyplot` et `pandas` . Les polynômes dans `numpy` peuvent être créés, manipulés et même ajustés à l'aide de la classe `Polynomial` du package `numpy.polynomial`. Il faudra charger ces modules, en tapant les commandes suivantes:

[ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from numpy.polynomial import Polynomial
```

1.1 Déclaration d'un Polynôme

Un polynôme $P \in \mathbb{R}_n[X]$ à une indéterminée X s'écrit sous la forme :

$$P(X) = a_0 + a_1X + a_2X^2 + \cdots a_nX^n = \sum_{i=0}^{n} a_iX^i,$$

avec $a_i \in \mathbb{R}$.

Le polynôme P est caractérisé par ses coefficients a_0, a_1,...,a_n.

Pour déclarer le polynôme P, On utilise la classe `Polynomial` comme suit :

`P=Polynomial([a0,a1,...,an])`

Il faut écrire les coefficients par ordre de degré croissant.

Exemple:

Soient P et Q deux polynômes définies par $P(x) = 1 + 2x + 3x^3$, et $Q(x) = P(x) * P(x)$.

Le script ci-dessous renvoie correctement l'expression des polynômes P, l'expression du polynôme Q ainsi que l'image de 2 par P et Q.

[ ]:
```
P=Polynomial([1,2,0,3])
print(P)
Q=P*P
print(Q)
print("P(2)=",P(2))
print("Q(2)=",Q(2))
```

1.2 Présentation de la bibliothèque pandas

Pandas est une bibliothèque Python pour l'analyse de données. Il propose un certain nombre d'opérations d'exploration, de nettoyage et de transformation des données qui sont essentielles pour travailler avec des données en Python.

Les principales structures de données fournies par pandas sont Series et DataFrames. Les principales caractéristiques de pandas sont les suivantes:

* Génération de statistiques descriptives sur les données.

* Nettoyage des données à l'aide des fonctions pandas intégrées.

* Opérations de données fréquentes pour le sous-ensemble, le filtrage, l'insertion, la suppression et l'agrégation de données.

1.3 Les structures de données en pandas

Pandas nous offre deux structures de données principales à savoir: la classe Series et la classe DataFrame.

1. Une Series est un tableau à une dimension où chaque élément est indexé avec essentiellement un autre array (souvent de chaînes de caractères).

2. Un DataFrame est un tableau à deux dimensions où les lignes et les colonnes sont indexées.

Les options les plus courantes pour charger vos données dans une DataFrame Pandas sont:

* `read_csv`

* `read_excel`

Dans ce qui suit, on s'intéresse à coder la méthode d'interpolation de Lagrange et l'écriture matricielle au sens des moindres carrées.

2 Méthode d'interpolation de Newton

Question 1 :

Écrire une fonction nommée `differences_divisees(X, Y)` prenant en entrée :

- $X = \left(x_0, x_1, \cdots, x_n\right)^T$, un vecteur contenant les abscisses,
- $Y = \left(y_0, y_1, \cdots, y_n\right)^T$, un vecteur contenant les ordonnées.

Cette fonction doit retourner les coefficients de Newton $\beta_0, \beta_1, \cdots, \beta_n$ qui sont les différences divisées utilisées dans le polynôme d'interpolation de Newton. Les différences divisées sont définies récursivement par :

$\beta_0 = f[x_0] = y_0$ et d'une manière générale $f[x_i] = y_i$

$\beta_1 = f[x_0, x_1] = \frac{y_1 - y_0}{x_1 - x_0}$ et d'une manière générale $f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}$

$\beta_i = f[x_0, x_1, \ldots, x_i] = \frac{f[x_1, \ldots, x_i] - f[x_0, x_1, \ldots, x_{i-1}]}{x_i - x_0}$ et d'une manière génrale

$$f[x_i, x_{i+1}, \cdots, x_j] = \frac{f[x_{i+1}, \cdots, x_j] - f[x_i, \cdots, x_{j-1}]}{x_j - x_i}$$

[ ]:
```
def differences_divisees(X, Y):
    n = len(X)
    diff_div = np.zeros((n, n))
    diff_div[:,0] = Y
    for j in range(1, n):
        for i in range(j,n):
            diff_div[i,j] = (diff_div [i,j-1] - diff_div[i-1,j-1]) / (X[i] - X[i-j])
    return np.diag(diff_div)
```

Question 2 : Écrire une fonction nommée `base_Newton(X,i)` prenant en entrée $X = (x_0, x_1, \cdots, x_n)^T$, un vecteur colonne contenant les abscisses x_i des points d'interpolation, i un indice tel que $0 \leq i \leq n$, et retourne le polynôme de la base de Newton $W_i(x)$ donné ci-dessus en utilisant la classe `Polynomial`.

$$W_i(x) = \prod_{j=0}^{i} (x - x_j). \qquad \forall i \in \{0, ..., n\}$$

[ ]:
```
def base_Newton(X,i):
    W=Polynomial([1]) # Initialisation de W
    for j in range(i):
        a0= -X[j]
        a1= 1
        W=W*Polynomial([a0,a1])
    return W
```

Question 3 :

Écrire une fonction nommée `polynome_Newton(X,Y)` prenant en entrée $X = (x_0, x_1, \cdots, x_n)^T$ un vecteur colonne contenant les abscisses, $Y = (y_0, y_1, \cdots, y_n)^T$ un vecteur colonne contenant les ordonnées, et retourne le polynôme

$$P(x) = \sum_{i=0}^{n} \beta_i W_i(x).$$

[ ]:
```
def polynome_Newton(X,Y):
    P=Polynomial([0]) # Initialisation de P
    beta=differences_divisees(X, Y)
    for i in range(len(X)):
        P+=beta[i]* base_Newton(X,i)
    return P
```

Question 3 :

Déterminer l'expression du polynôme d'interpolation $P \in \mathbb{R}_2[X]$ qui relie les points de coordonnées $(-1,2)$, $(0,1)$ et $(1,-1)$ en utilisant la fonction `polynome_Newton`.

[ ]:
```
X=np.array([-1,0,1]).T
Y=np.array([2,1,-1]).T
polynome_Newton(X,Y)
```

Out[ ]: $x \mapsto 1.0 - 1.5x - 0.5x^2$.

3 Approximation polynomiale au sens des moindres carrées

Question 1 :

Écrire une fonction nommée `matrice(X,p)` prenant en entrée $X = \left(x_0, x_1, \cdots, x_n\right)^T$, un vecteur colonne contenant les abscisses, et un ordre du modèle polynomial `p`, et retourne la matrice A définie par:

$$A = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & \cdots & x_0^p \\ 1 & x_1 & x_1^2 & \cdots & \cdots & x_1^p \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & \cdots & x_n^p \end{pmatrix} \in \mathcal{M}_{n,p}(\mathbb{R}).$$

[ ]:
```
def matrice(X,p):
    n = len(X)
    A = np.zeros((n,p+1))
    for j in np.arange(0,p+1):
        A[:,j] = X[:,0] **j
    return A
```

Question 2 :

Écrire une fonction nommée `regression(X,Y,p)`, prenant en entrée $X = \left(x_0, x_1, \cdots, x_n\right)^T$ un vecteur colonne contenant les abscisses, $Y = \left(y_0, y_1, \cdots, y_n\right)^T$ un vecteur colonne contenant les ordonnées, `p` un ordre du modèle polynomial, et retourne le vecteur Λ des coefficients estimés du modèle au sens des moindres carrées vérifiant:

$$\Lambda = ({}^tAA)^{-1}\ {}^tAY.$$

[ ]:
```
def regression(X,Y,p):
    A= matrice(X,p)
    Lambda=  np.linalg.inv(np.transpose(A).dot(A)).dot(np.transpose(A)).dot(Y[:,0])
    return  Lambda
```

Question 3 :

Déterminer l'expression de la droite de régression qui ajuste au mieux les points $(-1,2)$, $(0,1)$ et $(1,-1)$ au sens des moindres carrés, en utilisant la fonction `regression` et la classe `Polynomial`.

[ ]:
```
X=np.array([[-1,0,1]]).T
Y=np.array([[2,1,-1]]).T
Polynomial(regression(X,Y,1))
```

Out[ ]: $x \mapsto \frac{2}{3} - \frac{3}{2}x$.

4 Application

Dans cette application, on s'intéresse à charger une base de données d'une agence immobilière dans un dataframe et effectuer des prédictions sur les prix et les superficies des logements en Tunisie.

1 Charger la base de données du fichier `immobilier_Tunisie.csv` et l'affecter à la variable `df`.

[ ]:
```
df = pd.read_csv('immobilier_Tunisie.csv')
```

2 Afficher les noms de colonnes de `df` en utilisant l'instruction `df.columns`.

[ ]:
```
df.columns
```

Un client de l'agence immobilière souhaite `acheter` un `appartement` situé dans la région de `Carthage`.

3 Extraire les lignes du dataframe `df` associées au catégorie `Appartements` dans la région `Carthage`, de type `Vendre`, et l'affecter à nouveau à la variable `df`.

Indication: Pour extraire des lignes d'un dataframe, vous pouvez utiliser la commande `df = df[df['nom de Series'].str.contains("chaine de caractère à chercher")]`.

[ ]:
```
df = df[df['category'].str.contains("Appartements")]
df = df[df['type'].str.contains("Vendre")]
df = df[df['region'].str.contains("Carthage")]
```

4 Créer une variable `prix` qui contient les valeurs numériques de la colonne `price` en utilisant la commande `.values`.

[ ]:
```
prix = df['price'].values
```

5 Créer une variable `area` qui contient les valeurs numériques de la colonne `size` en utilisant la commande `.values`.

[ ]:
```
area= df['size'].values
```

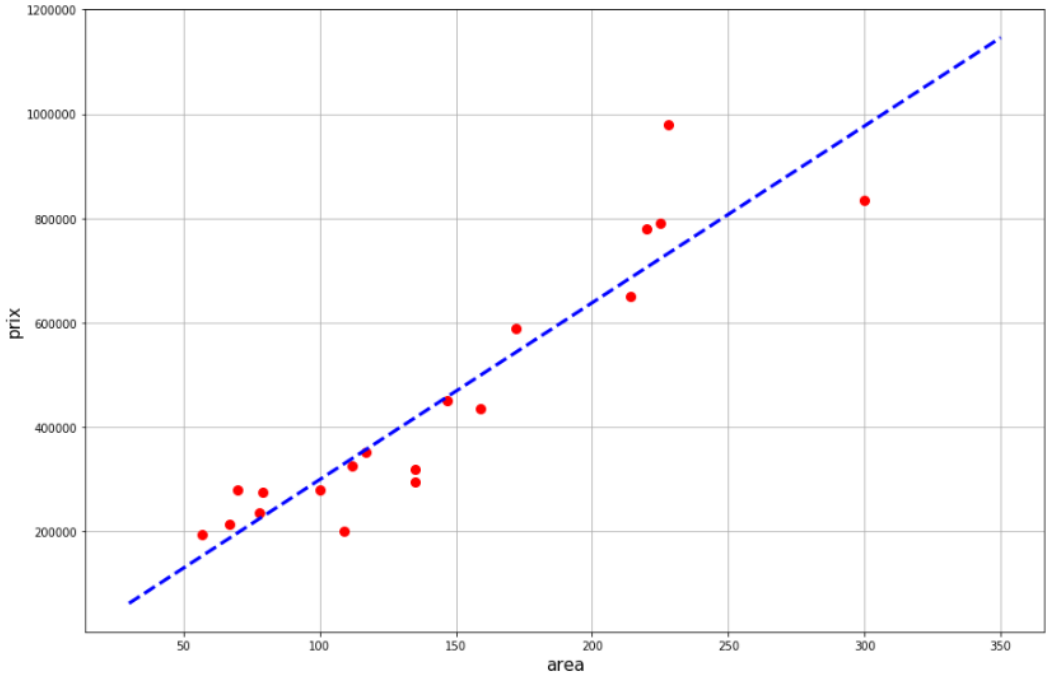
6 Transformer les deux vecteurs obtenus en vecteurs colonnes en utilisant la commande `.reshape(-1,1)`.

[ ]:
```
prix = prix.reshape(-1,1)
area = area.reshape(-1,1)
```

7 Représenter l'évolution des valeurs de la variable `prix` en fonction de la variable `area`.

Représenter sur la même figure la droite de régression au sens des moindres carrés de la variable `prix` en fonction de la variable `area`.

[ ]:
```
p=1
plt.figure(figsize=(15,10))
t=np.linspace(30,350,1000)
P=Polynomial(regression(area,prix,p))
plt.plot(area,prix,'ro',t,P(t),'b-', lw=3,markersize=8)
plt.xlabel('area',fontsize=15)
plt.ylabel('prix',fontsize=15)
plt.grid(True)
```



8 Le client veut avoir une valeur approximative du prix d'achat d'un appartement situé au `Carthage` d'une superficie de $200m^2$. Estimez cette valeur.

[ ]:
```
print(P(200))
```

Out[ ]: Le prix $\simeq$ 638313.59 DT.