Département de Mathématiques **Année Universitaire 2020-2021**

Faculté des Sciences

Université de Tlemcen

Module : Langages de programmation

Master E.D.P : M1 (Durée :01h15 mn)

Epreuve Finale

Problème :

On se donne les fonctions **Lambda_1 , Lambda_2 et Lambda_3** à deux valeurs $\widehat{x}$ et $\widehat{y}$ définies par :

Lambda_1 ($\widehat{x}$ **,** $\widehat{y}$ **) = (1 -** $\widehat{x}$ **-** $\widehat{y}$ **)**

Lambda_2($\widehat{x}$ **,** $\widehat{y}$ **) =** $\widehat{x}$

Lambda_3($\widehat{x}$**,** $\widehat{y}$**) =** $\widehat{y}$

Où

Le couple **(**$\widehat{x}$ **,** $\widehat{y}$**)** peut prendre les valeurs suivantes :

$\widehat{A}_0$ **= (**$\widehat{x}_0$ **,** $\widehat{y}_0$ **) = (0. , 0.) ,** $\widehat{A}_1$ **= (**$\widehat{x}_1$**,** $\widehat{y}_1$ **) = (1. ,0.) ,** $\widehat{A}_2$ **=(**$\widehat{x}_2$ **,** $\widehat{y}_2$ **) =(0., 1.)** dans **IR².**

On se donne aussi les deux variables **x** et **y** définies par :

x =x($\widehat{x}$**,** $\widehat{y}$**) = x0 * Lambda_1 (** $\widehat{x}$**,** $\widehat{y}$ **) + x1 * Lambda_2(** $\widehat{x}$ **,** $\widehat{y}$**) + x2* Lambda_3(** $\widehat{x}$ **,** $\widehat{y}$ **)**

y =y($\widehat{x}$**,** $\widehat{y}$**) = y0 * Lambda_1 (** $\widehat{x}$ **,** $\widehat{y}$ **) + y1 * Lambda_2(** $\widehat{x}$ **,** $\widehat{y}$ **) + y2* Lambda_3(** $\widehat{x}$ **,** $\widehat{y}$ **)**

Où

B_0 = (x_0 ,y_0) , B_1 = (x_1 ,y_1) et B_2 = (x_2 ,y_2) des points quelconques de IR^2 .

Partie 1 : (06.00 pts)

1) Calculer le **Jacobien J1** de la fonction **GG.**

Où

$$GG(\widehat{x}, \widehat{y}) = \begin{cases} GG1(\widehat{x}, \widehat{y}) = x(\widehat{x}, \widehat{y}) \\ GG2(\widehat{x}, \widehat{y}) = y(\widehat{x}, \widehat{y}) \end{cases}$$

2) Exprimer $\widehat{x}(x, y)$ et $\widehat{y}(x, y)$ en fonction de x et y **.**

Programmation : (14.00 pts)

- Ecrire un script en **python** en utilisant les listes ainsi que la function

get (des Dictionnaires) (pour faire un choix) , permettant de calculer :

Le **Jacobien J1 .**

On aura besoin des functions :

- **Lambda_1, Lambda_2, Lambda_3**
- **Function_x :** pour définir la fonction **x.**
- **Function_y :** pour définir la fonction **y.**
- **Deriv_part_x :** pour calculer les deux dérivées partielles de x.
- **Deriv_part_y :** pour calculer les deux dérivées partielles de y.
- **Choix_lambda** : pour choisir entre les fonctions Lambda_1 , Lambda_2 et Lambda_3.
- **Choix_points_A** : pour choisir entre les trois points $\widehat{A}_0$ **,** $\widehat{A}_1$ **et** $\widehat{A}_2$.
- **Det_Jacob** : pour calculer le déterminant de la matrice Jacobienne **J1**.

N.B :

1) Toutes les **functions** ainsi que le programme principal doivent être écrits dans le même script.

2) Pour faire un choix, il faut utiliser la function **get** (dictionnaire)

3) On sait que pour une fonction f définie par :

f : IR^2 ----------> IR

(x , y) ---------> f(x , y)

On a

$$\frac{\partial f(x,y)}{\partial x} = \lim_{h \to 0} (f(x+h,y) - f(x,y))/h$$

et

$$\frac{\partial f(x,y)}{\partial y} = \lim_{k \to 0} (f(x,y+k) - f(x,y))/k$$

4) Il faut respecter la nomination des functions citée ci-dessus.

5) Vérification : **B_0 = (4.,6.) , B_1 = (7. ,8.) et B_2 = (9., 11.)**

6) pour introduire les données , utiliser l'instruction :

x = list(map(float, input("introduire vos données séparées par un espace : ").split( ))).

Ou

x= [float(x) for x in input("Introduire plusieurs valeurs réelles séparées par un espace : ").split()]

Bon Courage

Département de Mathématiques **Année Universitaire 2020-2021**

Faculté des Sciences

Université de Tlemcen

Module : Langages de programmation

Master E.D.P : M1 (Durée :01h15 mn)

Corrigé de l'Epreuve Finale

""
""""""""""""""""""""""""""""""

""
""""""""""""""""""""""""""""""

Partie 01: Partie Théorique:

""
""""""""""""""""""""""""""""""

""
""""""""""""""""""""""""""""""""

1) Calcul du Jacobien : (02.00 pts)

Le Jacobien (ou Matrice Jacobienne) notée par J ,ça sera une matrice 2x2 donnée par:

J[0][0]=x1-x0 ; J[0][1]=x2-x0

J[1][0]=y1-y0 ; J[1][1]=y1-y0

2) Expression de x_chap et y_chap en fonction de x et y : (04.00 pts)

On doit résoudre le système **(S)** suivant :

(x1-x0)*x_chap + (x2-x0)*y_chap = x-x0

(S) (02.00 pts)

(y1-y0)*x_chap + (y2-y0)*y_chap = y-y0

On remarque que le déterminant du système (S) c'est le det(J).

On aura :

x_chap = 1./det(J) * [(x-x0)*(y2-y0) -(y-y0)*(x2-x0)]

(02.00 pts)

y_chap = 1./det(J) * [(x1-x0)*(y-y0) -(y1-y0)*(x-x0)]

""
""

****
****

Partie 02: Programmation

****
****

```
""""
```

00.50 pts : Pour la fonction Lambda_1

```
""""
def Lambda_1 (A):
    yy=1-A[0]-A[1]
    return yy
```

"""

00.50 pts : Pour la fonction Lambda_2

"""

def Lambda_2 (A):

yy=A[0]

return yy

"""

00.50 pts : Pour la fonction Lambda_3

"""

def Lambda_3 (A):

yy=A[1]

return yy

"""

On définit la fonction choix_lambda faire un choix entre les fonctions Lambda_1, Lambda_2 et Lambda_3 en utilisant la fonction **get** des dictionnaires:

"""

"""

00.50 pts : Pour la fonction Choix_lambda

"""

def Choix_lambda(argument_0):

switcher_1 = {

0: lambda_1,

1: lambda_2,

2: lambda_3

}

```
    func = switcher_1.get(argument_0)
    return func
"""
```

01.00 pts : Pour la fonction Choix_points_A

```
"""
def Choix_points_A(argument_1):
    A0=[0.,0.];A1=[1.,0.];A2=[0.,1.]
    switcher_A = {
        1: A0,
        2: A1,
        3: A2
        }
    points_A= switcher_A.get(argument_1)
    return points_A
"""
```

01.50 pts : Pour la fonction Function_x

```
"""
def Function_x (AA0,AA1,AA2,AA3,Func) :
          x=AA0[0]*Func[0](AA3)+AA1[0]*Func[1](AA3)+AA2[0]*Func[2](AA3)
           return x
"""
```

01.50 pts : Pour la fonction Function_y

```
"""
def Function_y(AA0,AA1,AA2,AA3,Func):
     y=AA0[1]*Func[0](AA3)+AA1[1]*Func[1](AA3)+AA2[1]*Func[2](AA3)
     return y
```

"""

02.00 pts : Pour la fonction Deriv_part_x

"""

def Deriv_part_x(AA0,AA1,AA2,AA,h,k,Func):

```
    AAA=[];AAA1=[] # list()
    xx=AA[0]+h; AAA.append(xx);xx=AA[1];AAA.append(xx)
    xx=AA[0]; AAA1.append(xx);xx=AA[1]+k;AAA1.append(xx)
    deriv_1=Function_x(AA0,AA1,AA2,AAA,Func)
    deriv_2=Function_x(AA0,AA1,AA2,AA,Func)
    xxx1=(deriv_1-deriv_2)
    deriv_x_x=xxx1/h
    deriv_3=Function_x(AA0,AA1,AA2,AAA1,Func)
    deriv_4=Function_x(AA0,AA1,AA2,AA,Func)
    xxx2=deriv_3-deriv_4
    deriv_x_y=xxx2/k
    return deriv_x_x,deriv_x_y
```

"""

02.00 pts : Pour la fonction Deriv_part_y

"""

def Deriv_part_y(AA0,AA1,AA2,AA,h,k,Func):

```
   AAA=[];AAA1=[] # list()
    xx=AA[0]+h;AAA.append(xx);xx=AA[1];AAA.append(xx)
    xx=AA[0];AAA1.append(xx);xx=AA[1]+k;AAA1.append(xx)
    deriv_1=Function_y(AA0,AA1,AA2,AAA,Func)
    deriv_2=Function_y(AA0,AA1,AA2,AA,Func)
```

```
    xxx3=(deriv_1-deriv_2)
    deriv_y_x=xxx3/h
    deriv_3=Function_y(AA0,AA1,AA2,AAA1,Func)
    deriv_4=Function_y(AA0,AA1,AA2,AA,Func)
    xxx4=(deriv_3-deriv_4)
    deriv_y_y=xxx4/k
    return deriv_y_x,deriv_y_y

"""
 00.50 pts :  Pour la fonction Det_Jacob
"""
def Det_Jacob(J_x,J_y):
    y=(J_x[0]*J_y[1])-(J_x[1]*J_y[0])
    return y

# 00.50 pts : pour l'utilisation de : if __name__=="__main__"
if __name__=="__main__":
     i=0;Func=[]
     while i < 3:
       xx=Choix_Lambda(i)
       Func.append(xx)
       i+=1
     print("Calcul des éléménts de La Jacobienne :\n")
     # B0=(2,2.); B1=[3.,3.];B2=[2.,3.]
     print("introduire B0:\n")
```

```
    """
      00.25 pts + 00.25 pts  + 00.25 pts =00.75 pts :
      Pour la lecture des points B0, B1 et B2.
    """
    #B0=list(map(float, input("Introduire vos valeurs séparées par un espace: ").split()))
    B0 = [float(x) for x in input("Entrer plusieurs valeurs réelles séparées par un espace :\n ").split()]
    print("introduire B1:\n")
    #B1=list(map(float, input("Introduire vos valeurs séparées par un espace: ").split()))
    B1 = [float(x) for x in input("Entrer plusieurs valeurs réelles séparées par un espace :\n ").split()]
    print("introduire B2:\n")
    #B2=list(map(float, input("Introduire vos valeurs séparées par un espace: ").split()))
    B2 = [float(x) for x in input("Entrer plusieurs valeurs réelles séparées par un espace :\n ").split()]

    """
      00.50 pts : Pour Faire un choix sur les points A0 , A1 et A2
    """

    print(" faire un choix suivant la valeur de A:\n")
    print("argument_1 == 1 : A0, ou argument_1 == 2 : A1,ou argument_1==3 :A2\n ")
    argument_1=int(input("lire argument_1  pour choisir A:\n"))
    points_A=Choix_points_A(argument_1)
```

```
#print("points_A:\n",points_A)


"""
   00.25 pts : Pour la lecture des réels h et k
"""
 print("lire h un réel positif assez petit :\n")
 h=float(input("introduire h:\n"))
 print("lire k un réel positif assez petit :\n")
 k=float(input("introduire k:\n"))
 """
   00.50 pts  :  Pour le calcul de Jacob_x
 """
Jacob_x=Deriv_part_x(B0,B1,B2,points_A,h,k,Func)
 """
   00.50 pts :  Pour le calcul de Jacob_x
 """
Jacob_y=Deriv_part_y(B0,B1,B2,points_A,h,k,Func)


"""
   00.50 pts : Pour le calcul du Déterminant
"""
det=Det_Jacob(Jacob_x,Jacob_y)
 """
```

Résultats :

On trouvera avec les points B0,B1 et B2 choisis:

Notons que la Jacobienne est une Matrice de type (2,2)

1ère ligne de la matrice Jacobienne:

3. 5.

1ème ligne de la matrice Jacobienne:

2. 5.

D'où le Déterminant sera est égal à :

det=5.

"""

```
print("Jacob_x:\n",Jacob_x)
print("Jacob_y:\n",Jacob_y)
print("det:\n",det)
```

M. A Benchaib