Tp OPtimisation

In [1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [2]:
```
# Definition de la fonction principale

def f(p):
    return 1/2*math.pow(p[0] , 2) + 7/2* math.pow(p[1],2)
```

In [ ]:

In [3]:
```
# gradiant de f

def grad_f(p):
    return np.array([p[0] ,7*p[1]])
```

In [ ]:

In [45]:
```
def norme(v):
    return math.sqrt(pow(v[0] , 2) + math.pow(v[1],2))
```

In [ ]:

In [58]:
```
def produit_scalaire(vect1 , vect2):
    s = 0
    if(len(vect1) != len(vect2)):
        return "imposible"
    else:
        for i in range(len(vect1)):
            s = s + vect1[i]*vect2[i]
        return s
```

In [ ]:

In [62]:
```
# pour calculer la norme d'un vecteur
def normeg(vect):
    s = 0
    for i in range(len(vect)):
        s = s+ math.pow(vect[i] , 2)
    return math.sqrt(s)
```

In [ ]:

Algorthime de Descente 1

In [ ]:

In [4]:
```
# second critere d'optimalité Stagnation de la solution courante

def stagnation_valeur_courante(xk_1 , xk):
    return norme(xk_1 - xk)
```

Descente à pas fixe respectant le critère d'optimalité

In [46]:
```
# x0 point de ou commence l'algorithme
# s represente le pas
# e represente la precision
def descente_a_pas_fixe(e , x0, s):
    k=0
    xk = x0
    data =[]
    try:
        while( norme(grad_f(xk)) >  e):
            dk = -grad_f(xk)
            xk = xk + s* dk
            k = k+1
            data.append([k , xk])
    except:
        return "Dv"

    return data
```

In [47]:
```
descente_a_pas_fixe(10e-5 , [7,1.5] , 0.125)
```

Out[47]:
```
[[1, array([6.125 , 0.1875])],
 [2, array([5.359375 , 0.0234375])],
 [3, array([4.68945312e+00, 2.92968750e-03])],
 [4, array([4.10327148e+00, 3.66210938e-04])],
 [5, array([3.59036255e+00, 4.57763672e-05])],
 [6, array([3.14156723e+00, 5.72204590e-06])],
 [7, array([2.74887133e+00, 7.15255737e-07])],
 [8, array([2.40526241e+00, 8.94069672e-08])],
 [9, array([2.10460461e+00, 1.11758709e-08])],
 [10, array([1.84152903e+00, 1.39698386e-09])],
 [11, array([1.61133790e+00, 1.74622983e-10])],
 [12, array([1.40992067e+00, 2.18278728e-11])],
 [13, array([1.23368058e+00, 2.72848411e-12])],
 [14, array([1.07947051e+00, 3.41060513e-13])],
 [15, array([9.44536696e-01, 4.26325641e-14])],
 [16, array([8.26469609e-01, 5.32907052e-15])],
 [17, array([7.23160908e-01, 6.66133815e-16])],
 [18, array([6.32765795e-01, 8.32667268e-17])],
 [19, array([5.53670070e-01, 1.04083409e-17])],
 [20, array([4.84461311e-01, 1.30104261e-18])],
 [21, array([4.23903647e-01, 1.62630326e-19])],
 [22, array([3.70915692e-01, 2.03287907e-20])],
 [23, array([3.24551230e-01, 2.54109884e-21])],
 [24, array([2.83982326e-01, 3.17637355e-22])],
 [25, array([2.48484536e-01, 3.97046694e-23])],
 [26, array([2.17423969e-01, 4.96308368e-24])],
 [27, array([1.90245973e-01, 6.20385459e-25])],
 [28, array([1.66465226e-01, 7.75481824e-26])],
 [29, array([1.45657073e-01, 9.69352280e-27])],
 [30, array([1.27449939e-01, 1.21169035e-27])],
 [31, array([1.11518696e-01, 1.51461294e-28])],
 [32, array([9.75788593e-02, 1.89326617e-29])],
 [33, array([8.53815019e-02, 2.36658272e-30])],
 [34, array([7.47088141e-02, 2.95822839e-31])],
 [35, array([6.53702124e-02, 3.69778549e-32])],
 [36, array([5.71989358e-02, 4.62223187e-33])],
 [37, array([5.00490688e-02, 5.77778983e-34])],
 [38, array([4.37929352e-02, 7.22223729e-35])],
 [39, array([3.83188183e-02, 9.02779661e-36])],
 [40, array([3.35289660e-02, 1.12847458e-36])],
```

```
[41, array([2.93378453e-02, 1.41059322e-37])],
[42, array([2.56706146e-02, 1.76324153e-38])],
[43, array([2.24617878e-02, 2.20405191e-39])],
[44, array([1.96540643e-02, 2.75506488e-40])],
[45, array([1.71973063e-02, 3.44383111e-41])],
[46, array([1.50476430e-02, 4.30478888e-42])],
[47, array([1.31666876e-02, 5.38098610e-43])],
[48, array([1.15208517e-02, 6.72623263e-44])],
[49, array([1.00807452e-02, 8.40779079e-45])],
[50, array([8.82065206e-03, 1.05097385e-45])],
[51, array([7.71807055e-03, 1.31371731e-46])],
[52, array([6.75331173e-03, 1.64214664e-47])],
[53, array([5.90914777e-03, 2.05268330e-48])],
[54, array([5.17050429e-03, 2.56585412e-49])],
[55, array([4.52419126e-03, 3.20731765e-50])],
[56, array([3.95866735e-03, 4.00914707e-51])],
[57, array([3.46383393e-03, 5.01143383e-52])],
[58, array([3.03085469e-03, 6.26429229e-53])],
[59, array([2.65199785e-03, 7.83036536e-54])],
[60, array([2.32049812e-03, 9.78795670e-55])],
[61, array([2.03043586e-03, 1.22349459e-55])],
[62, array([1.77663137e-03, 1.52936823e-56])],
[63, array([1.55455245e-03, 1.91171029e-57])],
[64, array([1.36023340e-03, 2.38963787e-58])],
[65, array([1.19020422e-03, 2.98704733e-59])],
[66, array([1.04142869e-03, 3.73380917e-60])],
[67, array([9.11250107e-04, 4.66726146e-61])],
[68, array([7.97343844e-04, 5.83407682e-62])],
[69, array([6.97675863e-04, 7.29259603e-63])],
[70, array([6.10466381e-04, 9.11574504e-64])],
[71, array([5.34158083e-04, 1.13946813e-64])],
[72, array([4.67388323e-04, 1.42433516e-65])],
[73, array([4.08964782e-04, 1.78041895e-66])],
[74, array([3.57844184e-04, 2.22552369e-67])],
[75, array([3.13113661e-04, 2.78190461e-68])],
[76, array([2.73974454e-04, 3.47738077e-69])],
[77, array([2.39727647e-04, 4.34672596e-70])],
[78, array([2.09761691e-04, 5.43340745e-71])],
[79, array([1.83541480e-04, 6.79175931e-72])],
[80, array([1.60598795e-04, 8.48969914e-73])],
[81, array([1.40523945e-04, 1.06121239e-73])],
[82, array([1.22958452e-04, 1.32651549e-74])],
[83, array([1.07588646e-04, 1.65814436e-75])],
```

In [ ]:

In [ ]:

Tableau de test pas le pas fixe

pasTest = [0.25,0.125 , 0.05 , 0.01]

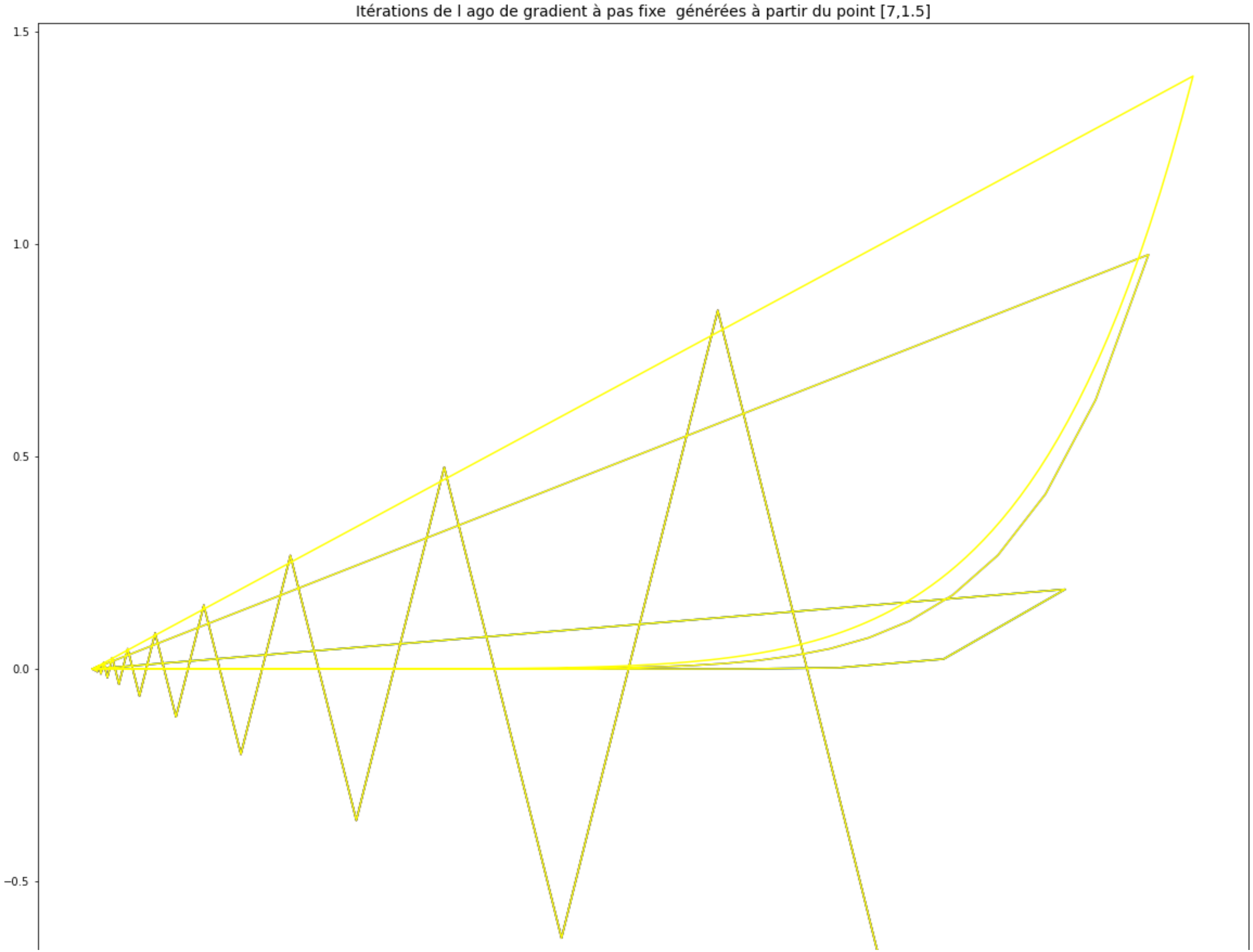
In [15]:
```
pasTest = [0.25,0.125 , 0.05 , 0.01]
```

In [48]:
```
resultat= []
for i in range(len(pasTest)):
    resultat.append(descente_a_pas_fixe(10e-5 , np.array([7,1.5] ),pasTest[i]))
```

In [52]:
```
def dataplot(datap):
    x=[]
    y=[]
    fig ,ax = plt.subplots()
    ax.set_prop_cycle(color=['red','green','blue','yellow'])
    for j in range(len(datap)):
        for i in range(len(datap[j]) ):
            x.append(datap[j][i][1][0])
            y.append(datap[j][i][1][1])
        plt.rcParams["figure.figsize"] = (20,20)
        ax.set_title('Itérations de l ago de gradient à pas fixe  générées à partir du point [7,1.5]',
        fontsize = 14)
        plt.plot(x,y ,label = "itermanfbjkfbgfkjbg")
    plt.show()
```

In [ ]:

In [53]:
```
dataplot(resultat)
```


Itérations de l ago de gradient à pas fixe générées à partir du point [7,1.5]
1.5
1.0
0.5
0.0
−0.5

In [ ]:

In [ ]:

```
#
```

In [ ]:

In [54]:

```
datap= descente_a_pas_fixe(10e-5 , np.array([7,1.5] ),0.01)
x=[]
y=[]
for i in range(len(datap) ):
    x.append(datap[i][1][0])
    y.append(datap[i][1][1])

#datap[1][1][0]
plt.rcParams["figure.figsize"] = (20,20)

#plt.plot(x,y , color ="blue")
#plt.show()
print("ok")
```

```
ok
```

Descente à pas Optimal respectant le critère d'optimalité

In [ ]:

In [55]:
```
#pas Optimal solution du probleme Minf(xk+s*dk) avec s positif

def s(x):
    return (pow(x[0],2) + pow(7,2)*pow(x[1] , 2)) /(pow(x[0],2) + pow(7,3)*pow(x[1] , 2))
```

In [ ]:
```
# e : precision
#x0 : point de départ

def descentePasOptimal(e , x0):
    xk = x0
    k=0
    data = []
    #norme(grad_f(x0))<=e
    while(norme(grad_f(xk)) > e):
        dk = -grad_f(xk)
        xk = xk + s(xk)*dk
        k = k+1
        data.append([k,xk])
    return data
```

In [ ]:
```
descentePasOptimal(10e-10 ,[7,1.5])
```

In [ ]:
```
```

Algroiithme 2 Amidjo et wolfe

f(x+s*d) <= f(x) +ep1*d(grad(f(x))T*d)

soit fi(s) <= fi(0) + ep1*d*fi'(O)

In [ ]:
```
```

In [ ]:
```
```

In [56]:
```
f(np.array([7,1.5]) + 0.5* np.array([-7, -10.5]) )
```

Out[56]:
```
55.34375
```

In [ ]:

In [ ]:

In [59]:
```
f(np.array([7,1.5]) )+0.125*10e-4* produit_scalaire(grad_f(np.array([7,1.5])) , np.array([-7, -10.5]))
```

Out[59]: 32.35509375

In [60]:
```
def premiere_condition_de_wolfe(x ,s, dk , ep1 ):
    if f(x +s*dk )  <= f(x) + ep1*s* produit_scalaire(grad_f(x) ,dk) :
        return True
    else:
        return False
```

In [61]:
```
premiere_condition_de_wolfe(np.array([8,0]) , 1 ,np.array([1,0]),0.0)
```

Out[61]: False

In [66]:
```
def deuxieme_condition_de_wolfe(x,s,dk , ep2):
    if produit_scalaire( grad_f(x + s*dk) , dk)    < ep2 * produit_scalaire(grad_f(x), dk):
        return False
    else:
        True
```

In [ ]:
```
deuxieme_condition_de_wolfe(np.array([8,19]) , 0.1 ,np.array([1,1]),0.8)
```

In [ ]:

In [ ]:

In [ ]:
```
produit_Scalaire(np.array([1,2]) , np.array([2,3,3]))
```

In [63]:
```
 norme([2,1,4])
```

Out[63]: 2.23606797749979

In [ ]:

In [ ]:

In [64]:

```
def recherche_lineaire_de_wolfe(x,ep1 ,ep2 ,s, d ):
    s_=0
    s_plus = float('inf')
    sk = s
    infini = float('inf')
    k = 0
    condition_wolfe = False
    while condition_wolfe == False:
        if premiere_condition_de_wolfe(x ,sk, d , ep1 ) == False :#pas trop grad
            s_plus = sk
            sk = (s_plus + s_ )/2
            condition_wolfe = False
        elif deuxieme_condition_de_wolfe(x,sk,d , ep2)  == False :# pas trop petit
            s_ = sk
            if s_plus < infini :
                sk = (s_plus + s_ )/2
            else:
                sk = 2*sk
        else :
            condition_wolfe = True
        k = k+1
    return sk
```

In [67]:

```
recherche_lineaire_de_wolfe(np.array([7,1.5] ), 10e-4 , 0.99 ,8 , np.array([-7,-10.5]))
```

Out[67]:

```
0.25
```

In [76]:

```
def descente_gradiant_wolfe(e , x0 , ep1 = 10e-4 , ep2  = 0.99 ,s = 1000000) :
    xk = x0
    k=0
    data=[]
    #norme(grad_f(x0))<=e
    while(norme(grad_f(xk)) > e):
        dk = -(grad_f(xk))
        sk = recherche_lineaire_de_wolfe(xk,ep1 ,ep2 ,s, dk )
        xk = xk + sk* dk
        k = k+1
        print(k)
        data.append([k,xk])
    return xk
```

In [78]:

```
descente_gradiant_wolfe(10e-5 , np.array([7, 1.5]) )
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
```

```
27
28
29
30
31
32
33
34
35
36
37
38
```

Out[78]:
```
array([ 5.53091490e-05, -9.91407779e-06])
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: