Projet 2

Zouiche Omar

6 Février 2021

Rapport 2

Exercice 1 : Espaces de fonctions, linéarité, continuité, norme : Sur $E = C([0,1], \mathbb{R})$ muni de $\|.\|_\infty$. On a la fonction \$ T : E $\rightarrow E\$telleque\forall f \in E, \forall x \in [0,1], (T(f))(x) = \int_0^x f(t)dt$

Exerice 2 : Espaces de fonctions, linéarité, continuité, norme : Sur $E = C([0,1], \mathbb{R})$ muni de $\|.\|_\infty$. Soit la fonction :

$$T : \begin{cases} E \rightarrow \mathbb{R} \\ f \mapsto T(f) = f(1) - 2f(0) \end{cases}$$

- T est définie en fonction de $f(1)$ et $f(0)$, f étant définie comme fonction continue sur $[0,1]$, qui est un intervalle de $\mathbb{R}$ donc un compact, une fonction continue définie sur un compact est ***bornée*** et atteint ses ***bornes***, donc : $f(1) \in \mathbb{R}$ et $f(0) \in \mathbb{R}$ et donc $T(f) \in \mathbb{R}, \forall f \in E$. - T est bien linéaire :

$$\begin{aligned} T(f + \lambda g) &= (f + \lambda g)(1) - 2(f + \lambda g)(0) \\ &= f(1) + \lambda g(1) - 2f(0) - 2\lambda g(0) \\ &= T(f) + \lambda T(g) \end{aligned} \tag{1}$$

- T est bien continue, $\forall f, g \in E$:

$$\begin{aligned} |T(f) - T(g)| &= |f(1) - g(1) - 2f(0) + 2g(0)| \\ &\leq |f(1) - g(1)| + 2|f(0) - g(0)| \\ &\leq ||f - g||_\infty + 2||f - g||_\infty \\ &= 3||f - g||_\infty \end{aligned} \tag{2}$$

- Quand on prend $g = 0$ on a :

$$|||T||| = \sup_{f \in E} |T(f)| \leq 3$$

Avec $f = 3x - 2$ on a bien $T(f) = 3$ donc \$ |||T||| = 3\$.

Exercice : Matrice de Hilbert, conditionnement et algorithmes de minimisation : Soit \$A = (a_{i,j})*{i,j}* , *a*{i,j} = $\frac{1}{i+j-1, \quad \forall i,j \in [\![1,n]\!]!]} \quad \$ lamatriced'Hilbert.Nousallonsutiliserleprogrammegradproj.pypourcomparerlesrsultatssurunematricecommecelled'Hilbert.Ona:J$:

$$\nabla J(x) = Ax - b$$

On remarque que le conditionnement de A est :

```
[11]: import math
import matplotlib.pyplot as plt
from scipy.optimize import minimize_scalar
import numpy as np
import scipy as sp
from scipy.linalg import hilbert
import random

A = hilbert(100)
cA = np.linalg.cond(A)

print(cA)
```

```
4.073996146476839e+19
```

Nous allons utiliser gradproj.py pour résoudre le système $Ax = B$ avec une matrice A dont le conditionnement est relativement grand.

```
[15]: def scxy(x,y): #produit scalaire
    scxy=0
    for i in range(0,len(x)):
        scxy = scxy + x[i]*y[i]
    return scxy

def func(x,y): #fonction J
    f = (1/2)*scxy(A.dot(x),x) - scxy(A.dot(np.ones(len(x))),x)
    return f

def funcp (x,y): #Grad J
    fp = A.dot(x) - A.dot(np.ones(len(x)))
    #for i in range(0, len(x)):
        #fp[i] = A.dot(x)[i] - A.dot(np.ones(len(x)))[i]
    return fp



#debut de l'initialisation des variables
nbgrad=145
epsdf=0.001
ndim=100
idf=0    #1 si calcul du gradient par differences finies ou analytique
```

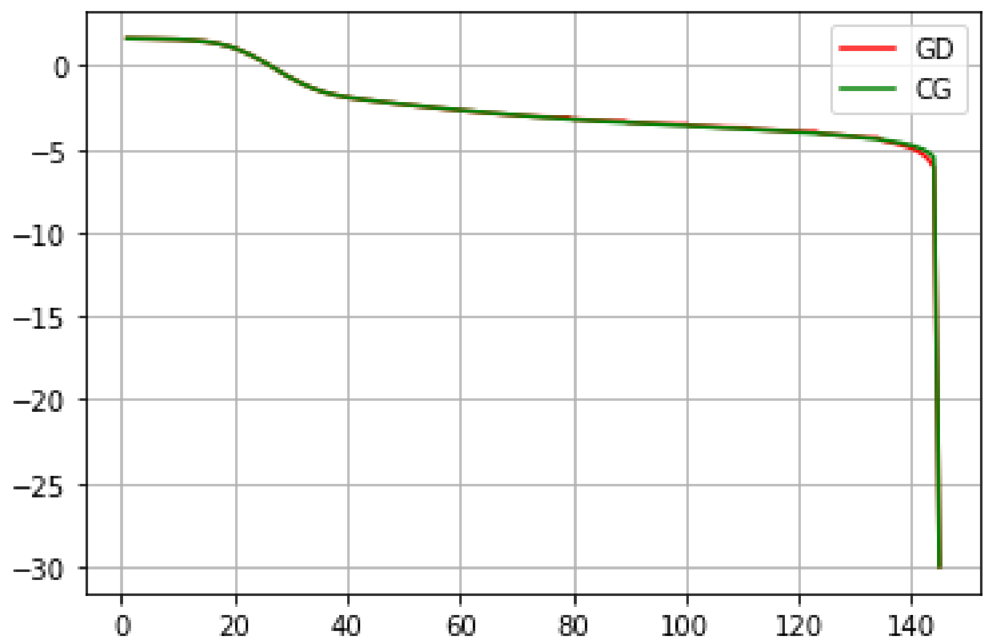
```python
#fin de l'initialisation des variables
erreur=np.zeros(nbgrad)

y= np.zeros(ndim)

for i in range(len(y)):
    y[i]= 1

for igc in [0,1]: #boucle generale
    ro0=0.01
    ro=ro0

    it=[]
    history=[]
    historyg=[]

    for ii in range(0, nbgrad):
        it=it+[ii+1]
        history=history+[0]
        historyg=historyg+[0]

    xmax=[]
    xmin=[]
    x=[]
    for i in range(0, ndim):
        xmax=xmax+[5]
        xmin=xmin+[-5]

        x=x+ [0.20]

    dfdx=np.zeros(ndim)
    d=dfdx

    for itera in range(0, nbgrad):
        dfdx0=dfdx

        if idf==1:
            for i in range(0, ndim):
                x[i]=x[i]+epsdf
                fp=func(x, y)
                x[i]=x[i]-2*epsdf
                fm=func(x, y)
                x[i]=x[i]+epsdf
                dfdx[i]=(fp-fm)/(2*epsdf)
        elif idf==0:
            dfdx=funcp(x, y)
```

```
        #il faut cree gg comme la some des carres de dfdx
        gg=0
        for j in range(0, ndim):
            gg=gg+dfdx[j]**2

#steepest descent
        if igc==0:
            for j in range(0, ndim):
                d[j]=dfdx[j]
        if igc==1:
#Polack-Ribiere  Conjugate Gradient
            xnum=0
            for j in range(0, ndim):
                xnum=xnum+dfdx[j]*(dfdx[j]-dfdx0[j])
            xden=0
            for j in range(0, ndim):
                xden=xden+dfdx[j]**2
            beta=0
            if(xden>1.e-30):
                beta=max(0,xnum/xden)

            for j in range(0, ndim):
                d[j]=dfdx[j]+beta*d[j]


#New xn+1= xn+rho*d  with d either -Gradf ou from CG
        for i in range(0, ndim):
            x[i]=x[i]-ro*d[i]
            x[i]=max(min(x[i], xmax[i]), xmin[i])

        erreur[itera] = np.linalg.norm(x - np.ones(len(x)))
        f=func(x, y)
        history[itera]=f
        historyg[itera]=gg


        if (itera >2 and history[itera-1] > f):
             ro=min(ro*1.25, 100*ro0)
        else:
             ro=max(ro*0.6, 0.01*ro0)


    print('Le minimum calculé est :',x)

    h1=history[nbgrad-1]
    hg1=historyg[1]
```

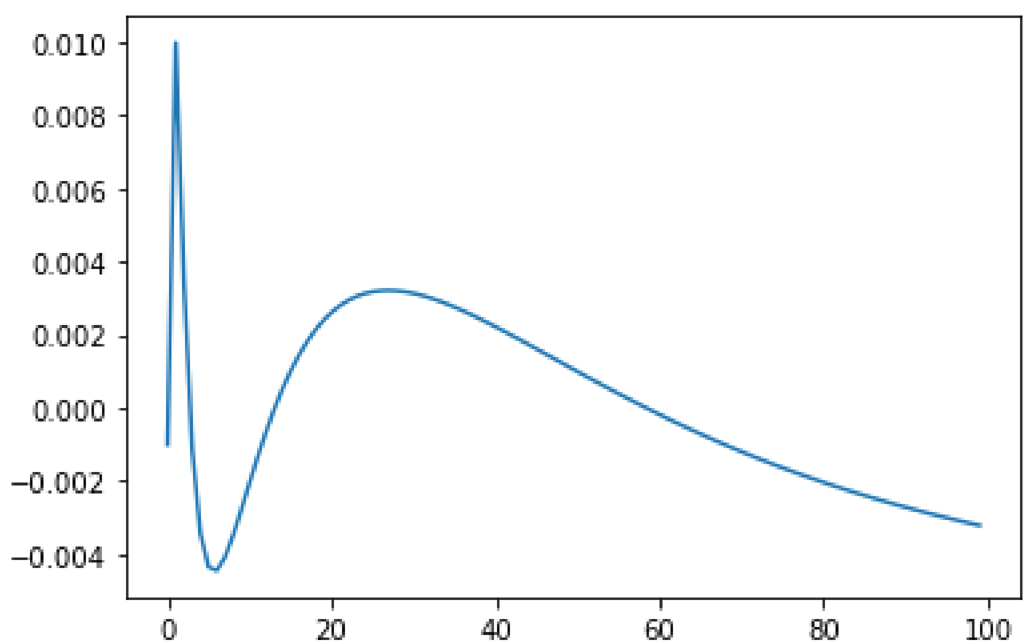
```
    for itera in range(0, nbgrad):
        history[itera]=max(abs(history[itera] - h1),1.0e-30)
        historyg[itera]=historyg[itera] /hg1
    #plt.plot(it,history)
    print("igc=",igc)
    if igc==0:
        plt.plot(it, np.log10(history), color='red', label='GD')
        xd = x
        erreur_d = erreur
        erreur = np.zeros(nbgrad)
    if igc==1:
        plt.plot(it, np.log10(history), color='green',label='CG')
        xc = x
        erreur_c = erreur
        erreur = np.zeros(nbgrad)


plt.legend()
plt.grid(True)
plt.show()

plt.plot(np.array(xd)-(xc))

plt.figure()

plt.plot(np.array(it),np.array(erreur_d), label = '$||x-x_{cible}||$')
plt.title('Convergence en fonction des itérations (GD)')
plt.legend()

plt.figure()
plt.plot(np.array(it),np.array(erreur_c), label = '$||x-x_{cible}||$')
plt.title('Convergence en fonction des itérations (GC)')
plt.legend()
```

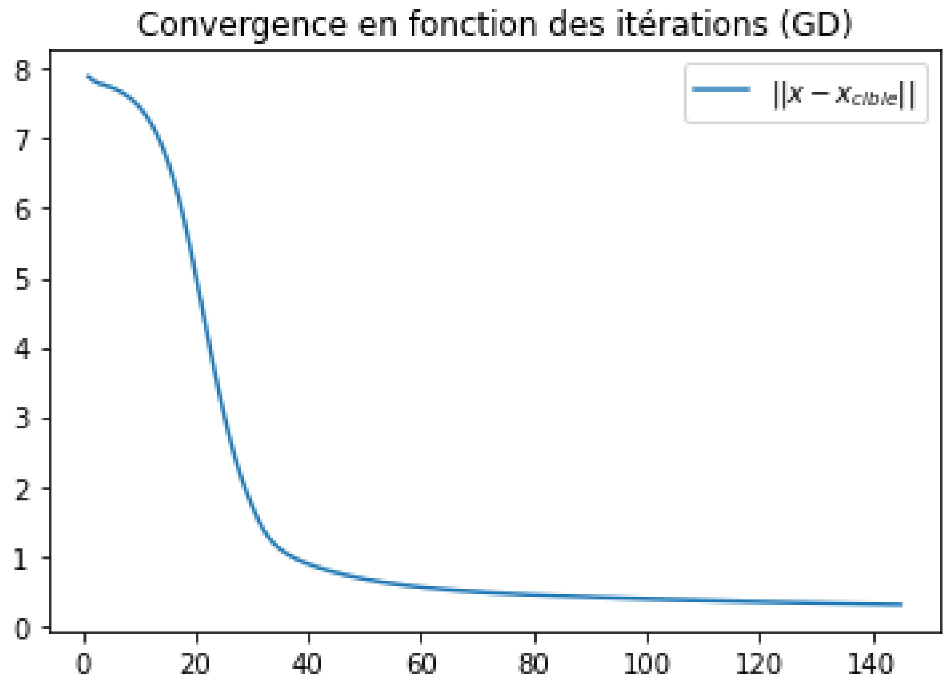
```
Le minimum calculé est : [0.9876549636239575, 1.0291357827293754,
1.0320399951796635, 1.019138584177282, 1.0038133155176137, 0.9905023743498873,
0.9803337512647073, 0.9732701923617787, 0.9689006321662967, 0.9667357346675987,
0.9663150769619163, 0.9672407290139985, 0.9691821175467759, 0.971870328060239,
0.9750892722615371, 0.9786666936296978, 0.982466100187983, 0.9863799200321758,
0.9903238538652553, 0.994232283256324, 0.9980545660446944, 1.0017520587803534,
1.0052957270192195, 1.0086642275744966, 1.0118423684011773, 1.0148198703089024,
1.017590369987486, 1.020150616189913, 1.0224998207909612, 1.0246391342758578,
1.026571221415633, 1.0282999177862822, 1.0298299516658023, 1.0311667189120652,
1.0323161008596913, 1.0332843172087702, 1.0340778074233836, 1.0347031353892886,
1.0351669130700694, 1.0354757396947525, 1.0356361536499772, 1.0356545947678848,
1.0355373751188925, 1.0352906567597424, 1.0349204351637278, 1.0344325272867636,
1.0338325634069383, 1.0331259820280965, 1.0323180272613968, 1.03141374820149,
```

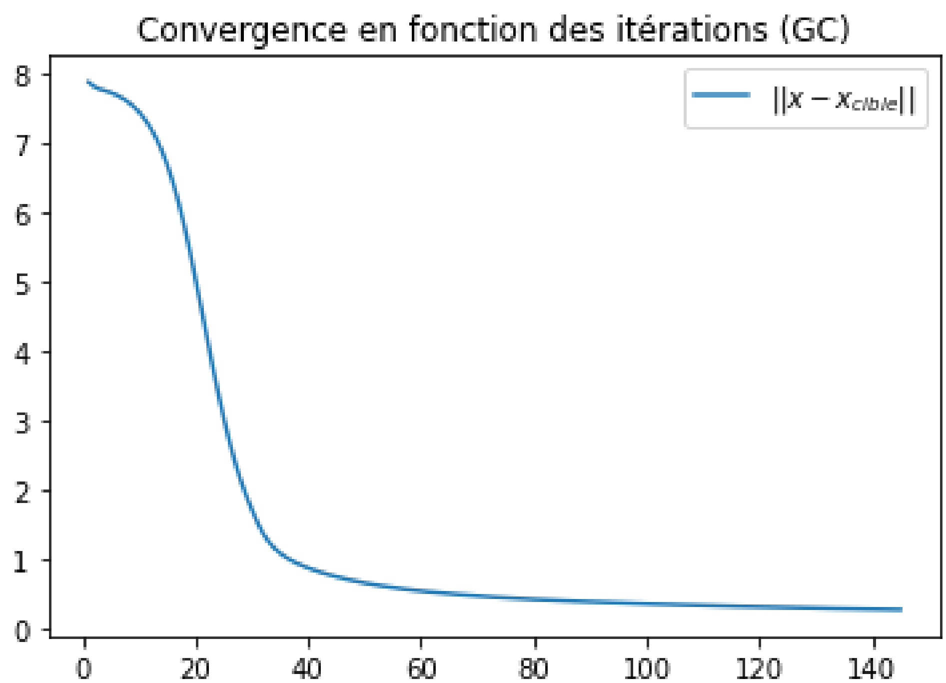
```
1.030417999898474, 1.0293354455970751, 1.0281705599701976, 1.0269276331240322,
1.0256107751897623, 1.024223921350209, 1.0227708371769983, 1.021255124176274,
1.019680225459986, 1.0180494314749136, 1.0163658857353075, 1.0146325905149034,
1.0128524124638894, 1.0110280881232143, 1.0091622293150087, 1.00725732839319,
1.0053157633422387, 1.0033398027155562, 1.0013316104081622, 0.9992932502602458,
0.9972266904904145, 0.9951338079586856, 0.993016392260892, 0.9908761496567194,
0.9887147068348368, 0.9865336145187584, 0.9843343509178142, 0.982118325027824,
0.9798868797864126, 0.9776412950877184, 0.9753827906620632, 0.9731125288251619,
0.9708316171023846, 0.968541110732986, 0.9662420150593056, 0.9639352878057437,
0.9616218412524465, 0.9593025443081976, 0.9569782244870216, 0.9546496697929,
0.952317630516754, 0.9499828209496671, 0.947645921016373, 0.9453075778325696,
0.9429684071896773, 0.9406289949706059, 0.9382898984994595, 0.9359516478287142,
0.9336147469665422, 0.9312796750472311]
igc= 0
Le minimum calculé est : [0.9886590404762988, 1.0191621167534521,
1.0286027240609898, 1.020247272695651, 1.0072522821516254, 0.9948572868074493,
0.9847812630297268, 0.9773538558440805, 0.9723843157978506, 0.9695145069744177,
0.9683615492205546, 0.9685725734308362, 0.9698422554167107, 0.9719147859377493,
0.9745797403594013, 0.9776660023730965, 0.9810355248119256, 0.984577637210945,
0.9882041256400463, 0.9918450984670486, 0.9954455664253977, 0.9989626383448618,
1.002363232777329, 1.005622215133108, 1.0087208826458802, 1.011645732275902,
1.0143874582160413, 1.0169401355723835, 1.019300555040263, 1.0214676801525457,
1.0234422041585605, 1.025226188010186, 1.026822764486895, 1.0282358963462217,
1.0294701786800011, 1.0305306774989866, 1.031422798055158, 1.0321521776055649,
1.0327245982907691, 1.033145916583314, 1.033422006397308, 1.0335587134684447,
1.033561819034041, 1.0334370111883566, 1.0331898625699132, 1.0328258132698722,
1.0323501580398813, 1.031768037036211, 1.031084429465555, 1.0303041496051466,
1.0294318447589879, 1.02847199478633, 1.0274289128976604, 1.0263067474671381,
1.0251094846510984, 1.023840951638253, 1.0225048203868428, 1.0211046117287432,
1.0196436997412477, 1.018125316304196, 1.0165525557755164, 1.0149283797291557,
1.0132556217107735, 1.0115369919742299, 1.009775082169341, 1.00797236995753,
1.0061312235367306, 1.0042539060608466, 1.0023425799432943, 1.000399311036229,
0.9984260726799611, 0.9964247496187429, 0.9943971417810781, 0.9923449679235791,
0.9902698691390355, 0.98817341222975, 0.9860570929481755, 0.9839223391074173,
0.981770513564606, 0.9796029170802057, 0.9774207910573164, 0.9752253201640976,
0.9730176348436183, 0.9707988137148942, 0.9685698858690773, 0.9663318330646533,
0.9640855918257717, 0.9618320554474069, 0.9595720759111595, 0.9573064657154725,
0.9550359996238437, 0.952761416334475, 0.950483420074915, 0.948202682124823,
0.9459198422700807, 0.9436355101914519, 0.9413502667904069, 0.9390646654553363,
0.9367792332705718, 0.9344944721709009]
igc= 1
```



[15]: <matplotlib.legend.Legend at 0x21703b04c10>







Le calcul de x pas la méthode de la plus forte pente comparé au calcul par le gradient conjugué, on remarque un manque de précision due au mal conditionnement de la matrice. Néamoins les méthodes convergent vers 0.