Mona Mokart

TP3 IMA203
Méthodes variationnelles

PARTIE 1 : Débruitage par régularisation quadratique

Q1

resoud_quad_fourier minimise $\sum_i ||K_i \star u - V_i||^2$. Pour minimiser E1 il faut l'exprimer sous la forme précédente.
Avec E1 = ||u-v||² + λ||grad(u)||² on peut identifier :
K1 = dirac
K2 = $\sqrt{\lambda}$K , avec Kx le noyau de convolution qui correspond au gradient selon x
K3 = $\sqrt{\lambda}$Ky, avec Ky le noyau de convolution qui correspond au gradient selon y
C'est ce que fait la fonction minimisation quadratique

Q2

bruit d'ecart-type 25

On voit que si lambda est trop petit, on remarque que le bruit est toujours bien présent. Quand lambda est grand, la minimisation renvoie une image floutée.

Quand lambda est petit, on cherche à minimiser ||u-v||² de façon plus importante que ||grad(u)||² donc on retrouve un image proche de l'image initiale v qui est l'image bruitée et inversement si lambda est grand on cherche davantage à minimiser le gradient

Or quand on minimise le gradient on minimise les fortes pentes, les contours, ce qui explique le flou

Q3

bruit d'ecart-type 5

```python
def diff(lamb,u_0,v):
    ''
    u_1 = minimisation_quadratique(v,lamb).copy()
    return norm2(u_1-v)-norm2(u_0-v)

def dicho_mini(u_0,v,debut,fin,eps):
    a = debut
    b= fin
    ecart = b-a
    while ecart>eps:
        m = (a+b)/2

        if diff(m,u_0,v)>0:
            #la solution est inférieure à m
            b = m
        else:
            #la solution est supérieure à m
            a = m
        ecart = abs(b-a)
    return m
```

```python
dicho_mini(im,imb,0.1,10,0.001)
```

```
0.331427001953125
```

Ainsi en connaissant la norme du bruit de l'image bruité et l'image restaurée, on peut trouver un lambda satisfaisant

Q4

bruit d'ecart-type 5

```python
lambdas=np.linspace(0.1,1,200)
normes=[]
for lamb in lambdas :
    u_1 = minimisation_quadratique(imb,lamb).copy()
    normes.append([norm2(u_1-im),lamb])
normes.sort()
print(normes[0][1])
```

```
0.11809045226130654
```

On ne retrouve pas la même valeur de lambda. Surement parce qu'ici on connaît l'image parfaite.

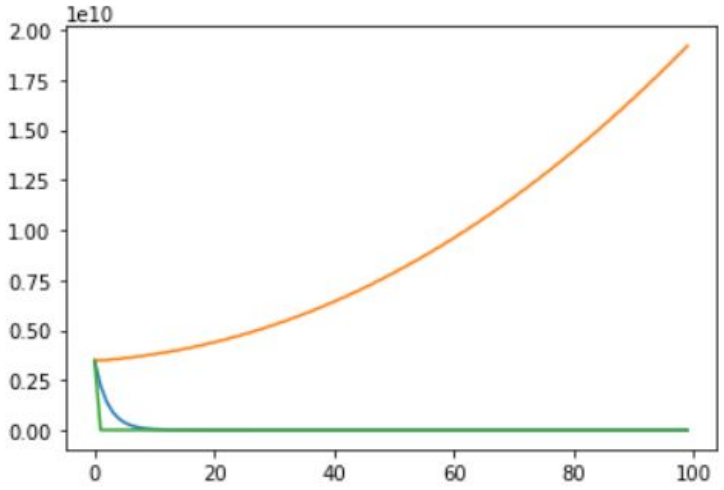
PARTIE 2 : Débruitage par variation totale

Q1

bruit d'ecart-type 25

```
u1,en1 =minimise_TV_gradient(imb,1,0.1,100)
u2,en2=minimise_TV_gradient(imb,1,1,100)
u3,en3=minimise_TV_gradient(imb,1,0.5,100)
plt.plot(en1)
plt.plot(en2)
plt.plot(en3)
```

```
[<matplotlib.lines.Line2D at 0x14ef30597f0>]
```



On voit que selon la valeur du pas l'énergie ne converge pas forcément et si cela converge ce n'est pas à la même vitesse. Quand elle ne converge pas elle augmente ce qui n'est pas acceptable. Ainsi la méthode de descente de gradient pour minimiser la variation totale ne fonctionne pas correctement.

On remarque qu'avec des pas différents on ne converge pas vers la même image. Ici le pas 1 est trop grand pour converger vers un résultat satisfaisant.

Ainsi la descente de gradient avec la variation totale n'est pas satisfaisante, il faut trouver un autre algorithme pour la minimiser

Q2

```
from time import time

t1 = time()
(u001,energ001)=minimise_TV_gradient(imb,40,0.1,100)
d1 = time()-t1
E2grad=E2_nonperiodique(u001,imb,40)
```

```
t2=time()
uchamb=vartotale_Chambolle(imb,40,itmax=30)
d2=time()-t2
E2chamb=E2_nonperiodique(uchamb,imb,40)
```

```
te de gradient : ",d1,"\nDurée de minimisation avec Chambolle :",d2,"\nRapport des ener
```

```
Durée de descente de gradient :  6.436352014541626
Durée de minimisation avec Chambolle : 1.3705432415008545
Rapport des energies (Chambolle/Descente de gradient) :  0.7698203758365081
```

On remarque que c’est plus long de minimiser par descente de gradient avec un pas de 0.1 et 100 itérations qu'avec l’algorithme de Chambolle. De plus, l'énergie est meilleure avec Chambolle car plus petite. Pour obtenir la même énergie avec descente de gradient il faudrait diminuer le pas et augmenter les itérations mais cela sera encore plus long que ça ne l'est déjà (avec pas=0.1 et iteration=100)

On obtient un rapport d’énergie proche de 1 avec 1000 pas d’une valeur de 0.01 ce qui donne une durée de calcul = 62.55186128616333 s soit 47 fois plus long qu’avec Chambolle.

La technique de minimisation de E2 est nettement plus satisfaisante avec Chambolle.

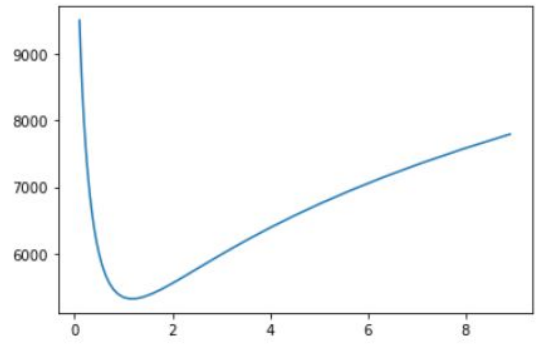
PARTIE 3 : Comparaison

1 Calcul des meilleurs lambdas

Pour la régularisation quadratique :

```
4
5  errq=[]
6  vk_q=np.arange(-1,1,0.05)
7  for k in vk_q:
8      restq=minimisation_quadratique(imb,10**(k))
9      errq.append(norm2(restq-im))
10
11 plt.plot(10**vk_q,errq)
12 i_q=np.argmin(errq)
13 lambdas_q=10**vk_q
14 print("Meilleur lambda pour régularisation quadratique : ",lambdas_q[i_q])
```

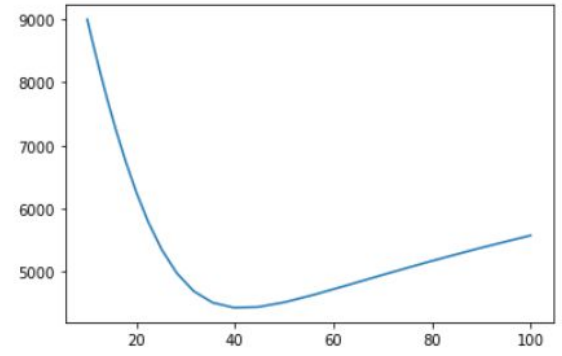
```
Meilleur lambda pour régularisation quadratique :  1.1220184543019658
```



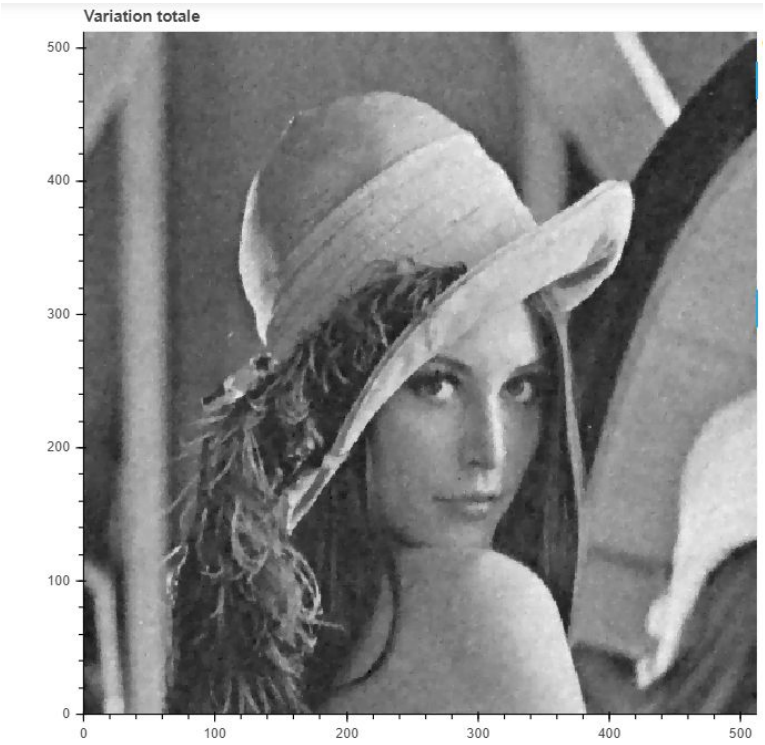
Pour la variation totale :

```
1  errvt=[]
2
3  vk_vt=np.arange(1,2.01,0.05)
4  for k in vk_vt:
5      restva=vartotale_Chambolle(imb,10**(k))
6      errvt.append(norm2(restva-im))
7  plt.plot(10**vk_vt,errvt)
8  i_vt=np.argmin(errvt)
9  lambdas_vt=10**vk_vt
10 print("Meilleur lambda pour la variation totale : ",lambdas_vt[i_vt])
```

```
Meilleur lambda pour la variation totale :  39.81071705534978
```

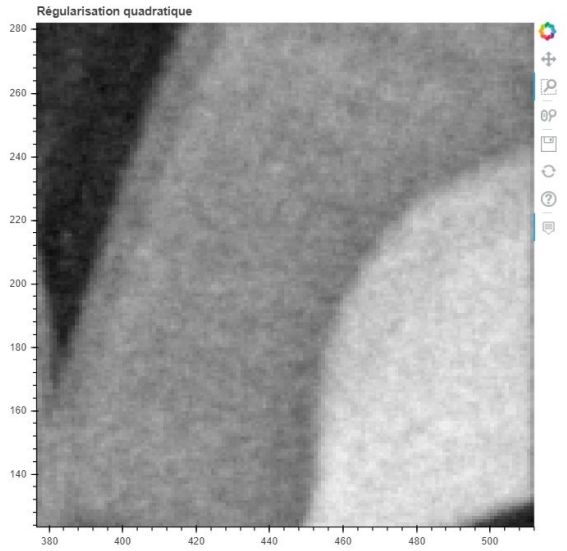


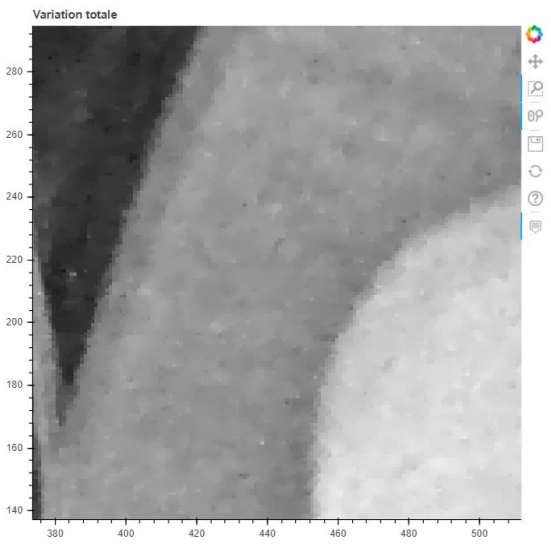
2 Comparaison des méthodes :




```
Erreur :  4480.320025383569
Temps de calcul :  1.4236247539520264
```

Bords :
Exemple zoom du **bord droit**





On obtient une meilleure erreur avec la minimisation de E2 (méthode variation totale) et à l'œil nu l'image semble meilleure qu'avec la régularisation quadratique. De plus, au niveau des bords, on remarque qu'ils sont plus proches de ceux de l'image initiale avec la méthode de variation totale. Cette méthode est légèrement plus longue à calculer mais reste très raisonnable. Ainsi la méthode par variation totale est meilleure.