Cristian Alejandro Chávez Becerra

TP de Méthodes Variationelles

Télécom Paris – IMA203

1 Débruitage par régularisation quadratic

1.1

Le programme resoud_quad_fourier minimise $E_1(u)$

$$\sum_i \|K_i \star u - V_i\|^2$$

en utilisant la méthode des moindres carrés dans le domaine de Fourier. Dont la solution est :

$$\hat{u} = \frac{\sum_i \overline{\hat{K}_i(\omega)}\,\hat{v}_i}{\sum_i |\overline{\hat{K}_i(\omega)}|^2}$$

Voici une explication étape par étape de la fonction et de son utilisation :

a. Initialisation des filtres K_x, K_y, et delta :

```
(sy, sx) = v.shape
Kx = np.zeros((sy, sx))
Ky = np.zeros((sy, sx))
Kx[0, 0] = 1
Kx[0, 1] = -1
Ky[0, 0] = 1
Ky[1, 0] = -1
delta = np.zeros((sy, sx))
delta[0, 0] = 1.0
```

Ici, les filtres K_x, K_y, et delta sont initialisés et seront utilisés dans l'énergie $E_1(u)$.

b. Définition de K et V:

s = lamb**0.5

K = (s * Kx, s * Ky, delta)

V = (np.zeros((sy, sx)), np.zeros((sy, sx)), v)

Les opérateurs K et les images V utilisés dans le processus de minimisation sont définis ici. K contient les filtres et V contient des zéros pour les deux premières composantes, et l'image observée v pour la troisième composante.

c. Appel à `resoud_quad_fourier`:

return resoud_quad_fourier(K, V)

Appelle la fonction `resoud_quad_fourier` avec les opérateurs et les images définis précédemment.

d. Fonction `resoud_quad_fourier`:

def resoud_quad_fourier(K, V):

...

Cette fonction résout le problème des moindres carrés dans le domaine de Fourier en utilisant les opérateurs K et les images V.

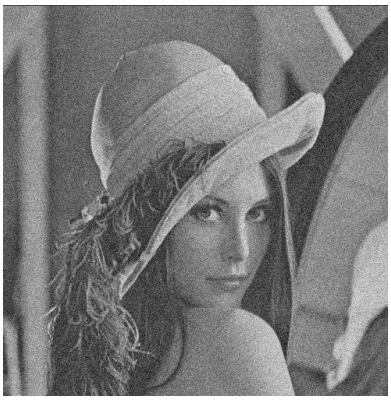
En résumé, la fonction `minimisation_quadratique` utilise des filtres et des opérateurs dans le domaine de Fourier pour minimiser l'énergie $E_1(u)$ et trouver l'image u qui minimise cette énergie. La résolution se fait à l'aide de la fonction `resoud_quad_fourier`.

1.2

Lambda = 0.1

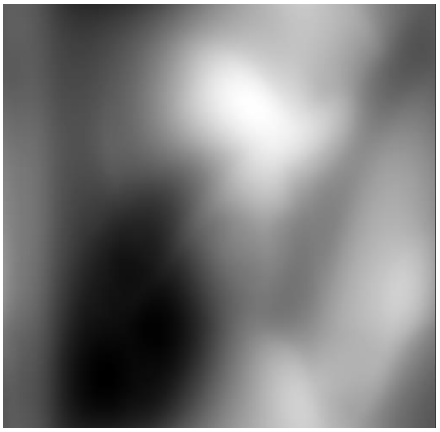
Cristian Alejandro Chávez Becerra

Lambda = 0.001

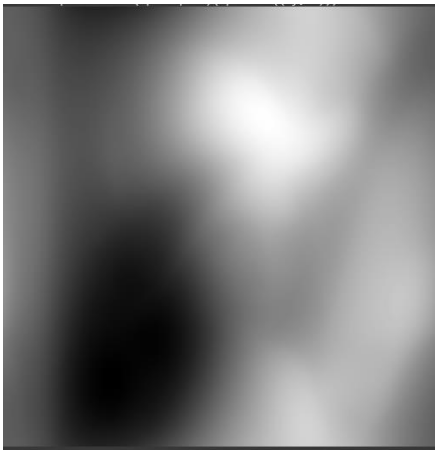


Lambda = 100

Lambda = 10000



Lambda = 1000000



Lorsque lambda prend des valeurs très élevées, l'algorithme accorde de plus en plus d'importance à la norme du gradient, et tente donc de réduire la norme du gradient de l'image, ce qui permet de rendre l'image plus régulière, mais ce faisant, l'image perd de nombreux détails.

En revanche, si lambda est très petit, il n'accordera pas autant d'importance à la norme du gradient, mais seulement au terme attaché aux donnés, de sorte qu'il rendra l'image aussi semblable que possible à l'image originale, et comme l'image originale est celle qui contient du bruit, elle en contiendra.

1.3

```
norm=norm2(imb-im)
lambda_min=0.001
lambda_max=1
resmin=minimisation_quadratique(imb,lambda_min)
resmax=minimisation_quadratique(imb,lambda_max)
errmin=norm2(imb-resmin)
errmax=norm2(imb-resmax)
for j in range(10):
    lambda_half=(lambda_min+lambda_max)/2
    resmil=minimisation_quadratique(imb,lambda_half)
    errmil=norm2(resmil-imb)
    if errmil>norm:
        lambda_max=lambda_half
        errmax=errmil
        resmax=resmil
    else:
        lambda_min=lambda_half
        errmin=errmil
        resmin=resmil
print(lmmil)
```

La méthode de dichotomie vise à trouver la valeur qui rend la fonction égale à zéro.

La fonction pour laquelle nous cherchons à trouver zéro est la fonction de la différence entre les deux quantités

$$f_1(\lambda) = ||\tilde{u}(\lambda) - v||^2 - ||u - v||^2$$

Zéro dans cette fonction signifie que les deux quantités sont égales, (il n'y a pas de différence entre elles).

Valeur pour lambda trouvée : 0.331

1.4

```
range_for_lambda=np.arange(-1,0,0.05)
best_Error=norm2(imb)+10
bestlambda = 0
for i in range_for_lambda:
    lamb=10**i
    res=minimisation_quadratique(imb,lamb)
    err=norm2(im-res)
    if err<best_Error:
        bestlambda=lamb
        best_Error=err

print(bestlambda)
```

Cet extrait de code est une exploration de différentes valeurs de lambda (λ) pour trouver la valeur qui minimise la norme L^2 entre l'image parfaite (u) et l'image reconstruite (ũ). Voici l'explication étape par étape :

`range_for_lambda = np.arange(-1, 0, 0.05)` : Un tableau ` lambda ` est créé allant de -1 à 0 avec des incréments de 0.05. Cet intervalle représente les valeurs possibles de log10(λ) explorées.

`best_Error = norm2(imb) + 10` : ` best_Error ` est initialisé à une valeur supérieure à la norme L^2 de l'image dégradée (`imb`). Cette valeur est utilisée pour effectuer des comparaisons et trouver le meilleur résultat.

Une boucle `for i in range_for_lambda:` est lancée pour itérer sur les valeurs possibles de log10(λ).

a. `lamb = 10**k` : Calcule la valeur actuelle de λ à partir de la valeur de `k`.

b. `res = minimisation_quadratique(imb, lamb)` : Effectue une minimisation quadratique pour la valeur actuelle de λ et obtient l'image reconstruite.

c. `err = norm2(im - res)` : Calcule la norme L^2 entre l'image parfaite (`im`) et l'image reconstruite (`res`).

d. `print(lamb, err)` : Affiche la valeur de λ et l'erreur calculée sur la console.

e. `if err < best_Error:` : Compare l'erreur actuelle avec la meilleure erreur jusqu'à présent. Si l'erreur actuelle est inférieure, met à jour `bestLambda` (la meilleure valeur de λ) et ` best_Error ` (la meilleure erreur).

A la fin de la boucle, ` bestLambda ` contiendra la valeur de λ qui résulte en la plus petite distance quadratique (L^2) entre l'image reconstruite et l'image parfaite.

Cette approche utilise une recherche exhaustive sur une échelle logarithmique pour trouver la meilleure valeur de λ. La boucle itère sur plusieurs valeurs de log10(λ), et la valeur réelle de λ est

calculée en utilisant 10^k. Le but est de trouver la valeur de λ qui minimise la différence entre l'image parfaite et l'image reconstruite.

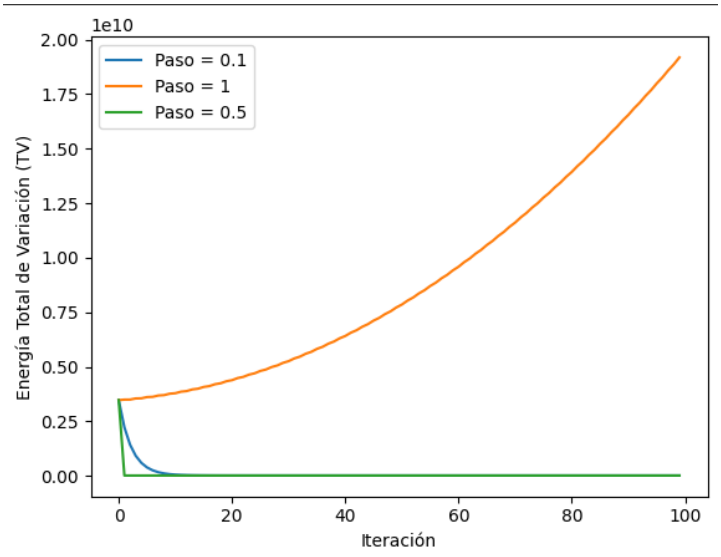
Valeur pour lambda trouvée : 0.11

Est différente de la valeur lambda trouvée au point précédent, peut-être en raison du fait que dans ce cas, nous comparons directement avec l'image originale.

2 Débruitage par variation totale

2.1

```
imb = degrade_image(im, 25)

(u, energ) = minimise_TV_gradient(imb, 1, 0.1, 100)   # paso = 0.1
(u, energ2) = minimise_TV_gradient(imb, 1, 1, 100)     # paso = 1
(u, energ3) = minimise_TV_gradient(imb, 1, 0.5, 100)    # paso = 0.5

plt.plot(energ, label='Paso = 0.1')
plt.plot(energ2, label='Paso = 1')
plt.plot(energ3, label='Paso = 0.5')

plt.legend()  # Agrega la leyenda

plt.xlabel('Iteración')
plt.ylabel('Energía Total de Variación (TV)')

plt.show()
```



Nous n'obtenons pas la même valeur d'énergie minimale, cela dépend du lambda, si lambda >= 1 l'énergie augmente.

2.2

```
from time import time

t1 = time()
(u001,energ001)=minimise_TV_gradient(imb,40,0.1,100)
d1 = time()-t1
F2grad=E2_nonperiodique(u001,imb,40)

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

print('Temps gradient: ' + str(d1))
print('Temps Chambolle: ' + str(d2))
print('Energie gradient: ' + str(F2grad))
print('Energie Chambolle: ' + str(E2chamb))
print('Proportion entre energies Chamb/Grad: ' + str(E2chamb/F2grad))
```

```
Temps gradient: 1.678403377532959
Temps Chambolle: 0.33785295486450195
Energie gradient: 264356717.9822526
Energie Chambolle: 202955902.95771545
Proportion entre energies Chamb/Grad: 0.7677349927280485
```

Nous pouvons observer que la méthode de Chambole prend moins de temps et que l'énergie résultante est également inférieure à celle de la méthode de descente de gradient.

3 Comparaison

Image obtenue avec la méthode minimisation_quadratic

Meilleur lambda : 1.12

Image obtenue avec la méthode vartotale_Chambolle

Cristian Alejandro Chávez Becerra

Meilleur lambda : 40.73

Dans une évaluation qualitative, les résultats de la méthodevartotale_Chambolle me semblent meilleurs, car l'image est plus régulière, a moins de bruit et préserve encore bien les bords et le contraste.