hw3-prob2-b

February 8, 2024

[1]:

```
#
import numpy as np
import scipy.io
import matplotlib.pyplot as plt
```

[2]:

```
# MAT
mat_path = r"../HW3_package/hw3_prob2.mat"
mat_data = scipy.io.loadmat(mat_path)


f = mat_data['f']
x_orig = mat_data['x_orig']
```

[3]:

```
def Dh(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[:, 1:cols] = u[:, 1:cols] - u[:, 0:cols-1]
    d[:, 0] = u[:, 0] - u[:, cols-1]
    return d

def Dht(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[:, 0:cols-1] = u[:, 0:cols-1] - u[:, 1:cols]
    d[:, cols-1] = u[:, cols-1] - u[:, 0]
    return d

def Dv(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[1:rows, :] = u[1:rows, :] - u[0:rows-1, :]
    d[0, :] = u[0, :] - u[rows-1, :]
    return d

def Dvt(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[0:rows-1, :] = u[0:rows-1, :] - u[1:rows, :]
    d[rows-1, :] = u[rows-1, :] - u[0, :]
    return d
```

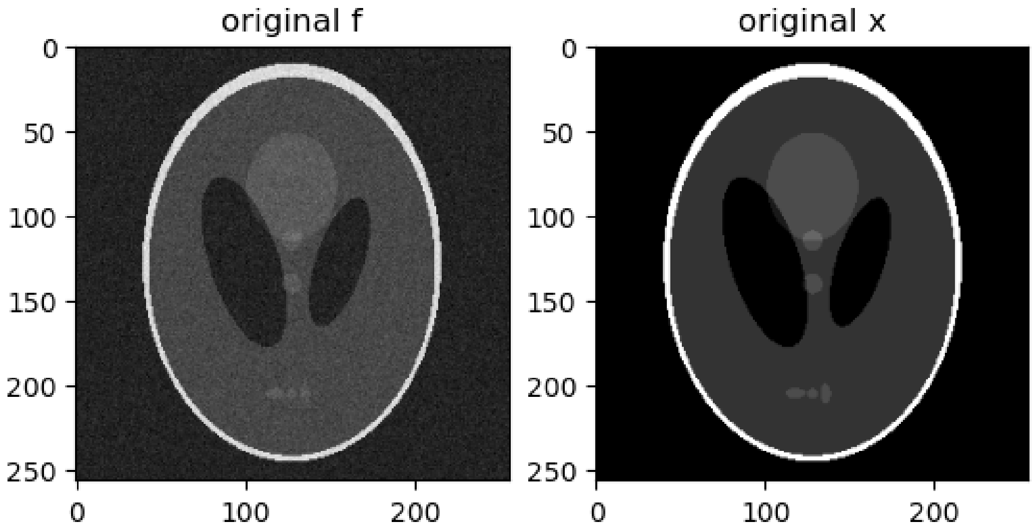
[4]:

```
# 1
wh = np.ones(f.shape)
wv = np.ones(f.shape)
x = np.zeros(f.shape)

mu = 0.02
lmbd = 0.0002

stopping_point = 1e-4
result = []



for idx in range(2):

    # w  1              idx  1     w       .
    if idx > 0:
        # mu, sigma, lambda
        sigma = 300
        mu = 0.0001
        lmbd = 2

        wh = 1 / (np.abs(Dh(x)) + sigma)
        wv = 1 / (np.abs(Dv(x)) + sigma)

    dh = np.zeros(f.shape)
    dv = np.zeros(f.shape)
    qh = np.zeros(f.shape)
    qv = np.zeros(f.shape)

    for iter in range(10000):
        x_minus1 = x

        # minimize x
        x = (mu * f)  + lmbd * (wh ** 2 * (np.roll(x, 1, axis = 1) + np.roll(x,␣
 ↪-1, axis = 1)) + wv ** 2 * (np.roll(x, 1, axis = 0) + np.roll(x, -1, axis =␣
 ↪0)) + wh * Dht(dh - qh) + wv * Dvt(dv - qv))
        x = x / (mu + lmbd * 2 * wh ** 2 + lmbd * 2 * wv ** 2 )

        # minimize dv, dh
        dh =  np.sign(wh *  Dh(x) + qh) * np.maximum(np.abs(wh * Dh(x) + qh) -␣
 ↪(1 / lmbd), 0)
```

```
    dv =  np.sign(wv *  Dv(x) + qv) * np.maximum(np.abs(wv * Dv(x) + qv) -␣
↪(1 / lmbd), 0)

    # minimize qv, qh
    qh = qh + (wh * Dh(x) - dh)
    qv = qv + (wv * Dv(x) - dv)

    # break check
    if  (np.linalg.norm((x - x_minus1), 2) / np.linalg.norm(x, 2)) <␣
↪stopping_point:
      break

  result.append(x)
```

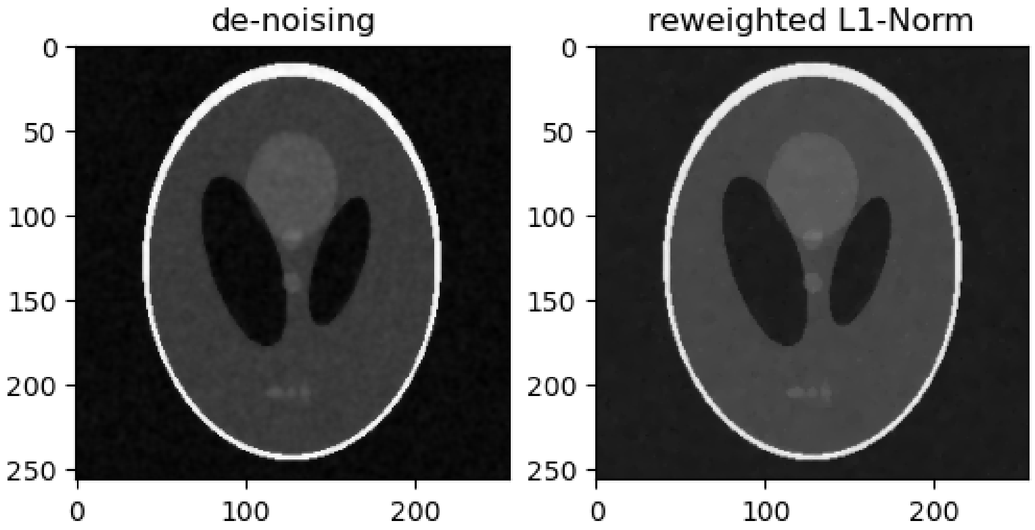
[5]:

```
# original f
plt.subplot(121)
plt.imshow(f, cmap = "gray")
plt.title( label = "original f")

# original x
plt.subplot(122)
plt.imshow(x_orig, cmap = "gray")
plt.title( label = "original x")

plt.show()
```



[6]:

```
plt.subplot(121)
plt.imshow(result[0], cmap = "gray")
plt.title( label = "de-noising")

plt.subplot(122)
plt.imshow(result[1], cmap = "gray")
plt.title( label = "reweighted L1-Norm")

plt.show()
```



hw3-prob3-l0-norm

February 8, 2024

[1]:

```
#
import numpy as np
import scipy.io
import matplotlib.pyplot as plt
```

[2]:

```
# MAT
mat_path = r"../HW3_package/hw3_prob3.mat"
mat_data = scipy.io.loadmat(mat_path)

A = mat_data['A']
b = mat_data['b']
x_orig = mat_data['x_orig']
```

[3]:

```
def normest_numpy(A):
    """
    NumPy       A 2-       .
    """
    # SVD    . full_matrices=False       SVD     .
    U, s, V = scipy.sparse.linalg.svds(A)
    #
    return s[0]

def Dh(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[:, 1:cols] = u[:, 1:cols] - u[:, 0:cols-1]
    d[:, 0] = u[:, 0] - u[:, cols-1]
    return d

def Dht(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[:, 0:cols-1] = u[:, 0:cols-1] - u[:, 1:cols]
    d[:, cols-1] = u[:, cols-1] - u[:, 0]
    return d

def Dv(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[1:rows, :] = u[1:rows, :] - u[0:rows-1, :]
    d[0, :] = u[0, :] - u[rows-1, :]
    return d

def Dvt(u):
    rows, cols = u.shape
    d = np.zeros((rows, cols))
    d[0:rows-1, :] = u[0:rows-1, :] - u[1:rows, :]
    d[rows-1, :] = u[rows-1, :] - u[0, :]
    return d
```

[4]:

```
mu = 1
stopping_point = 2e-4

# # normest    tau  sigma
norm_est = normest_numpy(A)
tau = 1 / norm_est**2
sigma = 1 / (tau * norm_est**2)

theta = 1/np.sqrt(1 + 2 * mu)
```

[9]:

```
y1 = np.zeros(b.shape)
y2 = np.zeros((256,256))
y3 = np.zeros((256,256))
x = np.zeros((256,256))

for idx in range(10000):

    y2 = Dh(x)
    y3 = Dv(x)

    # update y
    y2[np.abs(y2 + sigma * Dh(x)) < sigma] = 0
    y3[np.abs(y3 + sigma * Dv(x)) < sigma] = 0

    x = x.reshape(65536, 1)
    y1 = ((y1 + sigma * A @ x) - (sigma * mu * b)) / (1 + sigma * mu)

    # update x
    x_minus1 = x

    y2 = Dht(y2)
    y3 = Dvt(y3)
    y2 = y2.reshape(65536, 1)
    y3 = y3.reshape(65536, 1)
    x = x - (tau * (A.T @ y1 + y2 + y3))
    x[x < 0] = 0


    x = x + theta * (x - x_minus1)

    x = x.reshape(256,256)
    y2 = y2.reshape(256,256)
    y3 = y3.reshape(256,256)
    x_minus1 = x_minus1.reshape(256,256)

    if (np.linalg.norm((x - x_minus1), 2) / np.linalg.norm(x, 2)) < stopping_point:
        break
```

```
[]
0
[6.52144718 1.37102859 0.73400076 … 2.26296738 1.03017905 1.86745889]
1
[0.84798139 1.51123999 6.81563224 … 0.78266706 1.27299602 1.4217186 ]
2
[2.06626883 0.72762982 1.88489219 … 2.41315535 0.72479602 1.07143709]
3
[5.08517107 1.39348052 1.34634541 … 0.85698757 1.58229985 1.724236  ]
4
[1.45880518 0.61090098 0.51295893 … 2.10545921 0.69316783 0.73494501]
5
[0.60601983 0.93442238 4.39104272 … 0.92525467 1.01151733 1.12444888]
6
[1.1078163  1.24912246 0.53064401 … 4.13615759 5.89867559 1.25689521]
7
[1.05774482 0.78349542 0.55313863 … 0.72175682 0.78897116 0.67698228]
8
[ 0.52755213  1.37517512  0.86997148 … 10.42902613  9.31670566
  1.63204669]
9
[ 1.19926481  0.65790788  0.60995408 … 10.49028755  0.74634357
  0.60398769]
10
[ 0.91687358 17.24401679 12.88963101 … 11.75882913  0.81454467
  0.54401556]
11
[21.67329486 12.53151202 11.14622039 …  5.40966573 16.96385527
 12.59149911]
12
[21.40438472 11.96125659 10.82778487 … 18.48834446 13.33358011
```

```
 40.14843109]
157
[35.92158168 10.91365733  6.35724674 … 49.17760106  2.05558541
 39.80786955]
158
[35.85474935 10.85266539  6.28946152 … 48.90230886  2.12411142
 39.46856467]
159
```

[6]:

```
# original f
x = x.reshape((256, 256))
plt.subplot(121)
plt.imshow(x.T, cmap = "gray")
plt.title( label = "reconstructed image")


x = x.reshape((256, 256))
plt.subplot(122)
plt.imshow(x_orig, cmap = "gray")
plt.title( label = "original image")

plt.show()
```