CIE6010/MDS6118 (F2018) **Assignment 1**

Name (Chinese and English): ____________________

Course Number (taken by you): ____________________

A general optimization problem is

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X} \end{aligned}$$

Exercise 1.1.8 We set the coordinate of p and q on the plane to be:

$$p = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \quad q = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$$

Let x denote the horizontal coordinate of the intersection between the ray of light and the horizontal axis.

- The objective function is the travel time of the light, i.e.,

$$f(x) = \frac{1}{v}\sqrt{(p_1 - x)^2 + p_2^2} + \frac{1}{w}\sqrt{(q_1 - x)^2 + q_2^2}$$

- The feasibility set is

$$\mathcal{X} = [\min\{p_1, q_1\}, \max\{p_1, q_1\}]$$

- When $v = w$, the solution is

$$x = \frac{|q_2|}{|p_2| + |q_2|}p_1 + \frac{|p_2|}{|p_2| + |q_2|}q_1$$

  This is because by settign $f'(x) = 0$, we derive:

$$\frac{1}{v}\frac{x - p_1}{2\sqrt{(x - p_1)^2 + p_2^2}} + \frac{1}{w}\frac{x - q_1}{2\sqrt{(x - q_1)^2 + q_2^2}} = 0$$

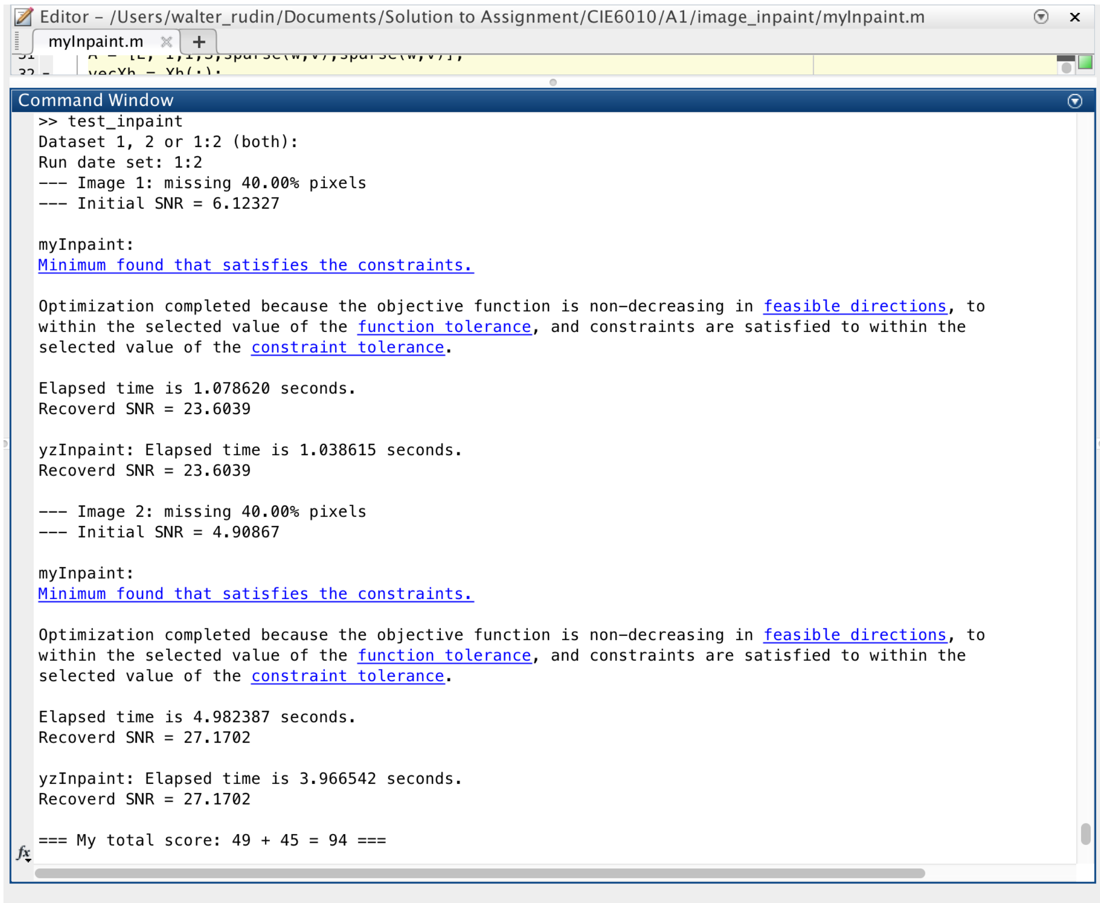
  Or equivalently,

$$(x - p_1)^2[(x - p_1)^2 + p_2^2] = (x - q_1)^2[(x - q_1)^2 + q_2^2] \implies (x - p_1)|q_2| = (q_1 - x)|p_2|$$

  After simplification, the optimal solution is obtained:

$$x = \frac{|q_2|}{|p_2| + |q_2|}p_1 + \frac{|p_2|}{|p_2| + |q_2|}q_1$$

The screen printout from my run



Editor – /Users/walter_rudin/Documents/Solution to Assignment/CIE6010/A1/image_inpaint/myInpaint.m

myInpaint.m

Command Window

```
>> test_inpaint
Dataset 1, 2 or 1:2 (both):
Run date set: 1:2
--- Image 1: missing 40.00% pixels
--- Initial SNR = 6.12327

myInpaint:
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to
within the selected value of the function tolerance, and constraints are satisfied to within the
selected value of the constraint tolerance.

Elapsed time is 1.078620 seconds.
Recoverd SNR = 23.6039

yzInpaint: Elapsed time is 1.038615 seconds.
Recoverd SNR = 23.6039

--- Image 2: missing 40.00% pixels
--- Initial SNR = 4.90867

myInpaint:
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in feasible directions, to
within the selected value of the function tolerance, and constraints are satisfied to within the
selected value of the constraint tolerance.

Elapsed time is 4.982387 seconds.
Recoverd SNR = 27.1702

yzInpaint: Elapsed time is 3.966542 seconds.
Recoverd SNR = 27.1702

=== My total score: 49 + 45 = 94 ===
```

Figure 1: Printout from Run

Generated Figures from my code

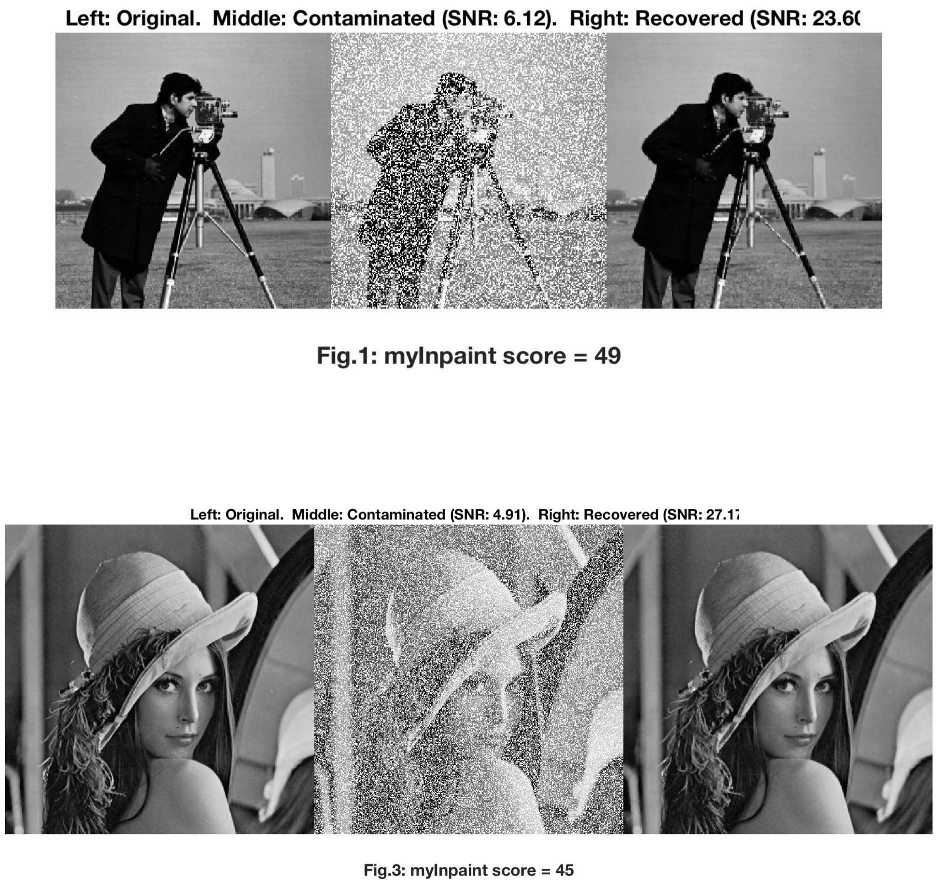


Figure 2: Output of my code

Summary of Matlab Project

- This project is to de-noise an image with a large number of missing pixels at random locations by solving a linear programming problem.
- The idea is to fill in missing pixels such that the output image has a small total variation. The details of the procedure are given as follows:
- Given the available matrix $\hat{X} \in \mathbb{R}^{n\times n}$ and $\Omega \in \mathbb{R}^{m\times 1}$, use command length to find m and n.
- To find $E \in \mathbb{R}$, we first create D by using speye and adding -1s on the first above diagonal; then E is computed as E = [kron(I,D);[kron[D,I]]; with I to be identity matrix with size n.
- S is a sub-matrix of identity matrix of size n^2, consisting of the rows whose indices are in Ω. We use the command given below to compute S:

  ```
  i = 1:m; j = Omega; v = 1 * ones(m,1);  S = sparse(i,j,v,m,n^2)
  ```

- Then we compute $\boldsymbol{A}$ by using the command [E,-I,I;S,sparse(w,v),sparse(w,v)];, where I are all identity matrices of size $n(n-1)$.
- f is a vector of size $(m+2n^2-n)$, with the first mth entries to be zero, the remaining entries to be one. $\boldsymbol{b}$ is a vector of size n^2-n+m, with the first $n^{-}n$ entries to be zero, the last m entries to be the available pixel values with indices in Ω.
- Call the MATLAB command z = linprog(f,[],[],A,b,LB,UB,options) to get optimal solution $\boldsymbol{z}$, where the **LB** is a vector consisting zero entries; UB is a vector consisting Inf entries; options are the *interior-point* algorithm.
- $\boldsymbol{x}$ is the first n^2 entries of $\boldsymbol{z}$, and then we reshape it into $n\times n$ matrix to get recovered image X for output.

A Copy of my Code myInpaint.m.

```
function [ X ] = myInpaint( Xh, Omega )
%      Usage:
%      Input:
%         Xh: the available matrix, with size n*n
%      Omega: the available pixel set Omega, with size m
%     Output:
%          X: the recovered graph

%% Figure Out relevant sizes
n = length(Xh);
m = length(Omega);

%% Construct the E matrix and the S matrix
I = speye(n,n);
% Generate vectors of subscripts and corresponding values of D
i = 1:n-1;
%j = 2:n;
v = -1 * ones(n-1,1);
D = sparse(i,i+1,v,n-1,n) + speye(n-1,n);
E = [kron(I,D);kron(D,I)];
% Generate vectors of subscripts and corresponding values of S
i = 1:m;
j = Omega;
v = 1 * ones(m,1);
S = sparse(i,j,v,m,n^2);

%% Construct the A matrix, f and b vectors
[v,~] = size(E);
[w,~] = size(S);
I = speye(v);
A = [E,-I,I;S,sparse(w,v),sparse(w,v)];
vecXh = Xh(:);
xh_Omega = vecXh(Omega);
b = [sparse(v,1);xh_Omega];
f = [sparse(n^2,1);ones(v * 2,1)]';

%% Solving the linear programming
LB = sparse(n^2 + 2*v,1);
UB = Inf * ones(n^2 + 2*v,1);
```

```
options = optimoptions('linprog','Algorithm','interior-point');
z = linprog(f,[],[],A,b,LB,UB,options);
%% Extract vector x and reshape into matrix X
x = z(1:n^2);
X = reshape(x,[n,n]);
end
```