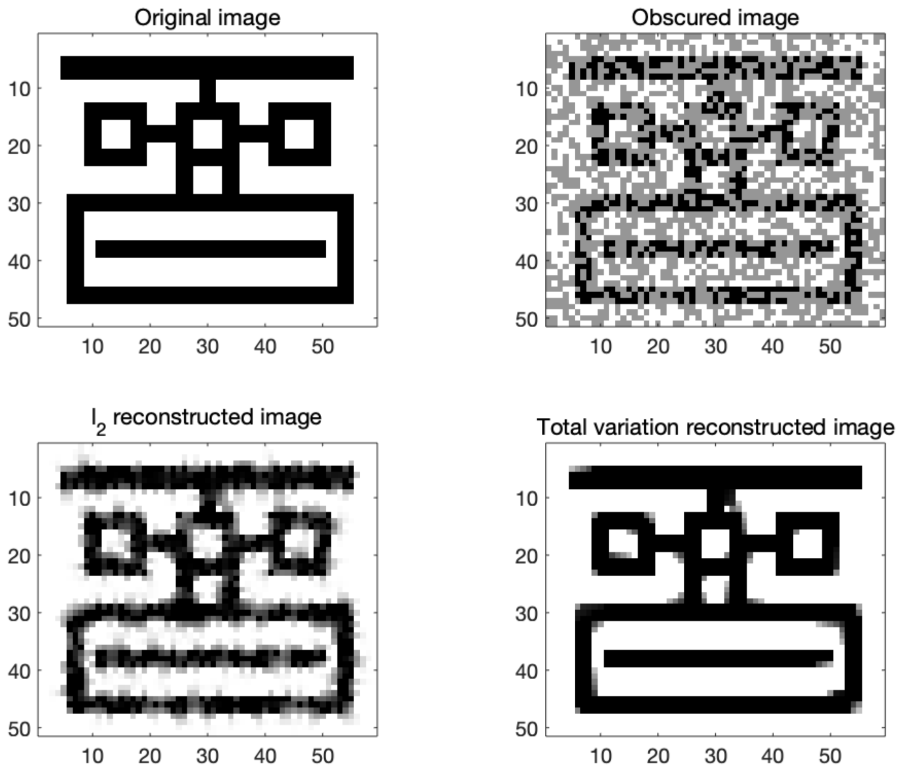
Additional exercise 1:

Tv_img_interp.m

```
% tv_img_interp.m
% Total variation image interpolation.
% EE364a
% Defines m, n, Uorig, Known.

% Load original image.
Uorig = double(imread('tv_img_interp.png'));

[m, n] = size(Uorig);

% Create 50% mask of known pixels.
rand('state', 1029);
Known = rand(m,n) > 0.5;

%%%%% Put your solution code here

% Calculate and define Ul2 and Utv.

% Placeholder:
Ul2 = ones(m, n);
Utv = ones(m, n);

% Calculate Ul2
cvx_begin
   variables Ul2(m,n)
   U1 = Ul2(2:end,2:end) - Ul2(1:(end-1),2:end)
   U2 = Ul2(2:end,2:end) - Ul2(2:end,1:(end-1))
   minimize(norm([U1(:) ; U2(:)],2))
   subject to
      Ul2(Known) == Uorig(Known)
cvx_end

%Calculate Utv
cvx_begin
   variables Utv(m,n)
   U1 = Utv(2:end,2:end) - Utv(1:(end-1),2:end)
   U2 = Utv(2:end,2:end) - Utv(2:end,1:(end-1))
   minimize(norm([U1(:) ; U2(:)],1))
   subject to
```

```
       Utv(Known) == Uorig(Known)
cvx_end

%%%%%

% Graph everything.
figure(1); cla;
colormap gray;

subplot(221);
imagesc(Uorig)
title('Original image');
axis image;

subplot(222);
imagesc(Known.*Uorig + 256-150*Known);
title('Obscured image');
axis image;

subplot(223);
imagesc(Ul2);
title('l_2 reconstructed image');
axis image;

subplot(224);
imagesc(Utv);
title('Total variation reconstructed image');
axis image;
```

The result is:


Original image

Obscured image

l_2 reconstructed image

Total variation reconstructed image

Additional Exercise 2:

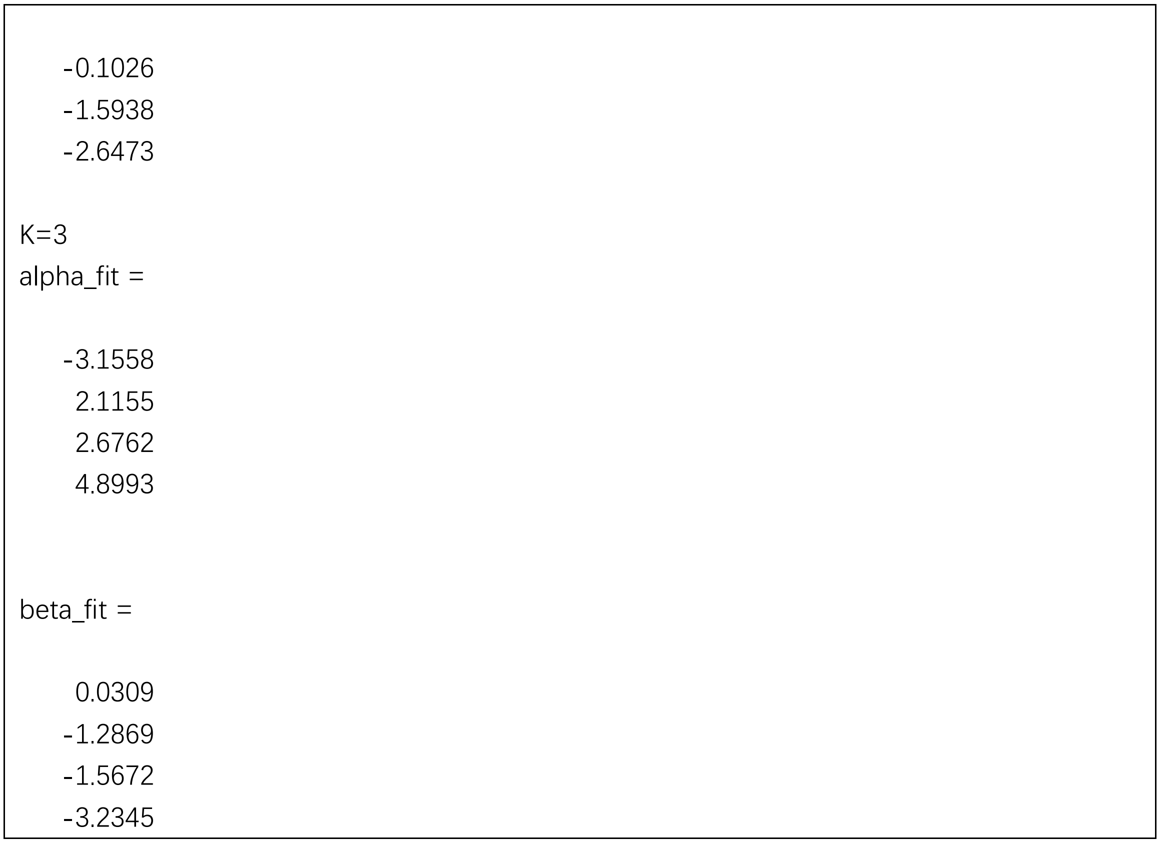
```
plot(x,y,'k:','linewidth',2);
hold on;

for K = 2:5
    a = linspace(0,1,K);
    index = 0;
    % find x index lies in each section
    for i = 2:K
        temp = find(x<=a(i));
        index = [index,temp(end)];
    end
```

```
   % cvx
   cvx_begin
      variables alpha_fit(K-1) beta_fit(K-1)
      F = [];

      % define the vector of constraint 1 (convex)
      C1 = alpha_fit(2:K-1) - alpha_fit(1:K-2);

      % Calculate each piecewise function
      for i = 1:K-1
         f = alpha_fit(i) .* x((index(i)+1):index(i+1)) +
beta_fit(i);
         F = [F;f];
      end

      lhs = [];
      rhs = [];
      for i = 1:K-2
         lhs = [lhs;alpha_fit(i) * a(i+1) + beta_fit(i)];
         rhs = [rhs;alpha_fit(i+1) * a(i+1) + beta_fit(i+1)];
      end


      % define minimize function
      minimize(norm(F-y))
      subject to
         C1 >= 0;
         lhs == rhs;
   cvx_end


   if K==2
      plot(x,F,'y','linewidth',2)
   elseif K==3
      plot(x,F,'r','linewidth',2)
   elseif K==4
      plot(x,F,'g','linewidth',2)
   else
      plot(x,F,'b','linewidth',2)
   end
```

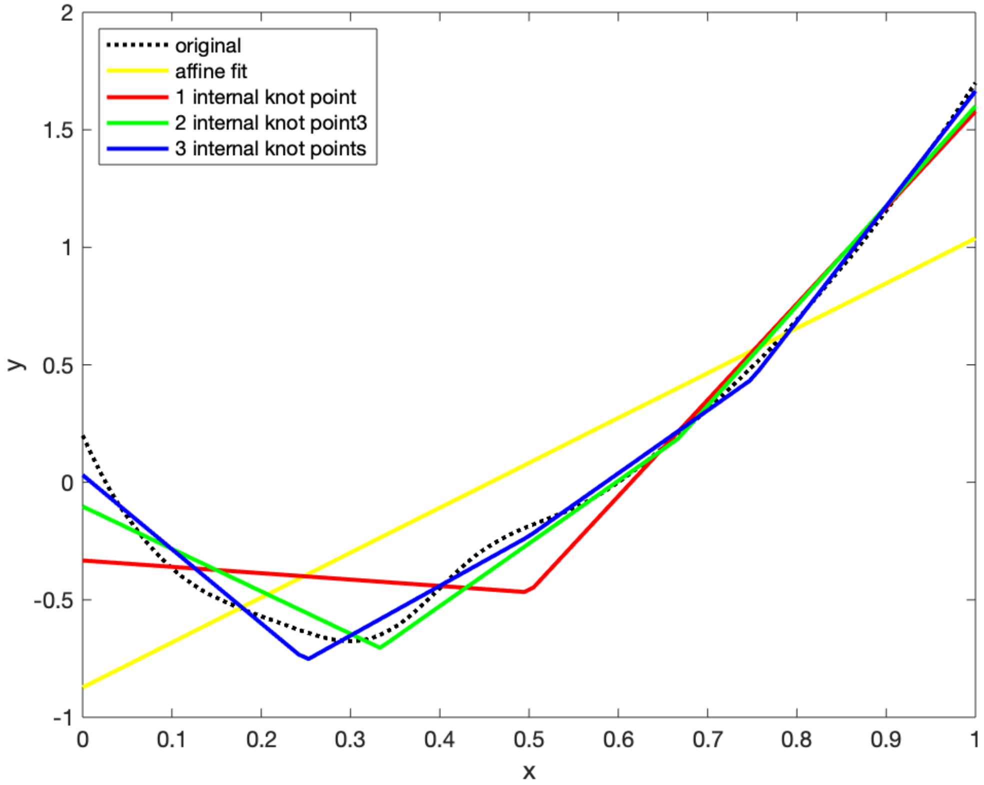
```
end
xlabel('x');
ylabel('y');
legend({'original' , 'affine fit','1 internal knot point' , '2
internal knot point3', ...
'3 internal knot points'}
, 'Location' , 'NorthWest');
```

The result is:

```
K = 0
alpha_fit =

    1.9110



beta_fit =

   -0.8725



K=1
alpha_fit =

   -0.2708
    4.0928



beta_fit =

   -0.3325
   -2.5143



K=2
alpha_fit =

   -1.8061
    2.6675
    4.2477



beta_fit =
```



```
   -0.1026
   -1.5938
   -2.6473

K=3
alpha_fit =

   -3.1558
    2.1155
    2.6762
    4.8993


beta_fit =

    0.0309
   -1.2869
   -1.5672
   -3.2345
```


original
affine fit
1 internal knot point
2 internal knot point3
3 internal knot points
y
x