HW2, Dept: 수리과학과, NAME: 국윤범

Problem 1 - Exercise 4.3

Problem 1 main script

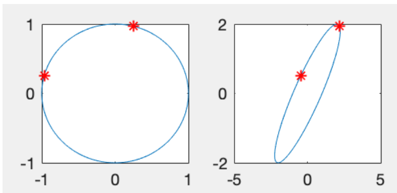
```
%%%% Problem 1 %%%%
% matrix in (3.7)
m3 = [1, 2; 0, 2];

% matrices in Exercise 4.1
m41 = [3, 0; 0, -2];
m42 = [2, 0; 0, 3];
m43 = [1,1; 0,0];
m44 = [1,1; 1,1];

figure;

% plot a unit circle with the right singular vectors
% and an ellipse with the left singular vectors.
transformation(m3, 1);
transformation(m41, 2);
transformation(m42, 3);
transformation(m43, 4);
transformation(m44, 5);
```

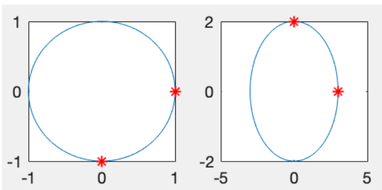
Problem 1 function script

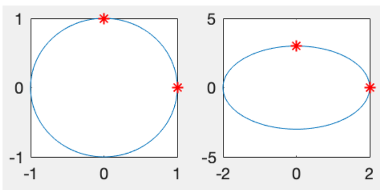
```
% function which draws two subplots; circle on left & ellipse on right
function transformation(mat, count)
t=0 : pi/100 : 2*pi;
xcircle = cos(t);
ycircle = sin(t);

circle_mat = [xcircle; ycircle];
% Each column of 'ellipse' corresponds to a point in image of circle
ellipse = mat * circle_mat;
xellipse = ellipse(1, :);
yellipse = ellipse(2, :);
```

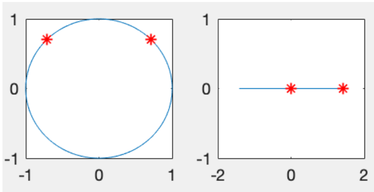
```

[u, s, v] = svd(mat);
s = diag(s);

% Draw circle on the left side
subplot(5, 2, 2*count-1); plot(xcircle, ycircle);
% Mark the right singular vectors
hold on
plot(v(1,1), v(2,1), 'r*');
plot(v(1,2), v(2,2), 'r*');
hold off

% Draw ellipse on the right side
subplot(5, 2, 2*count); plot(xellipse, yellipse);
% Mark the left singular vectors
hold on
plot(s(1)*u(1,1), s(1)*u(2,1), 'r*');
plot(s(2)*u(1,2), s(2)*u(2,2), 'r*');
hold off
end
```

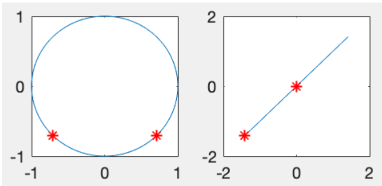
The following is the result of the codes above. (m3, m41, m42, m43, m44 in order)











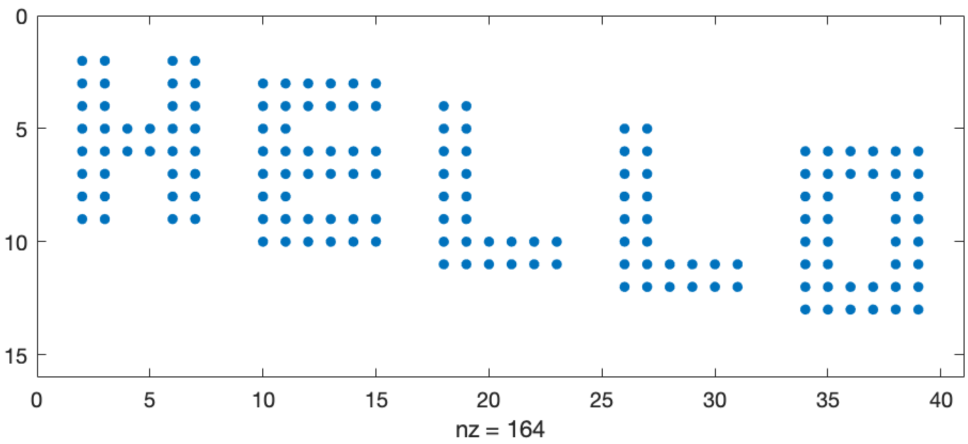
Problem 2 - Exercise 9.3

```matlab
%%%%% Problem 2 %%%%%
%%%%% 2-(1)

mat = zeros(15,40);
width = 5; height = 7;

% Filling the positions of each letter "H", "E", "L", "L", "O" by 1
for i=0:4
    % (x, y) coordinate of upper-left position of current letter
    x = 2+i; y = 2+i*8;
    mat(x:x+height, y:y+width) = 1;
end

% Change some portion of each letter block into 0
for i=0:4
    % (x, y) coordinate of upperleft position of current letter
    x = 2+i; y = 2+i*8;

    % Letter "H"
    if i==0
        mat(x:x+2, y+2:y+3)=0; mat(x+height-2:x+height, y+2:y+3)=0;
    % Letter "E"
    elseif i==1
        mat(x+2, y+2:y+width)=0; mat(x+height-2, y+2:y+width)=0;
    % Letter "L"
    elseif i==2
        mat(x:x+height-2, y+2:y+width)=0;
```

```
    % Letter "L"
    elseif i==3
        mat(x:x+height-2, y+2:y+width)=0;
    % Letter "O"
    else
        mat(x+2:x+height-2, y+2:y+width-2)=0;
    end
end

spy(mat)

%%%% 2-(2)
% Obtain singular values of given matrix
sigmas = svd(mat)'
% Remove zero singular values (used only for neat graph of semilogy)
sigmas_refined = sigmas; sigmas_refined(rank(mat)+1 :end)=0;

figure;
subplot(1,3,1); plot(sigmas); title("Raw Singular Values");
subplot(1,3,2); semilogy(sigmas); title("Logarithms of singular values");
subplot(1,3,3); semilogy(sigmas_refined);
title("Logarithms of only nonzero singular values");

% Compute the rank of matrix
sprintf("The rank of 'HELLO' matrix is %d.", rank(mat))

%%%% 2-(3)
[u, s, v] = svd(mat);

for i=1:rank(mat)
    approx_sigmas = diag(s); approx_sigmas(i+1:end)=0;
    ns = length(approx_sigmas);
    approx_s = s; approx_s(1:ns, 1:ns) = diag(approx_sigmas);

    low_matrix = u * approx_s * v';
    if i==1
        figure;
    elseif i==6
        figure;
    end

    if i<6
        subplot(rank(mat)/2, 1, i); pcolor(low_matrix); colormap(gray);
```

```
        elseif 5<i
            subplot(rank(mat)/2, 1, i-5); pcolor(low_matrix); colormap(gray);
        end
end
```

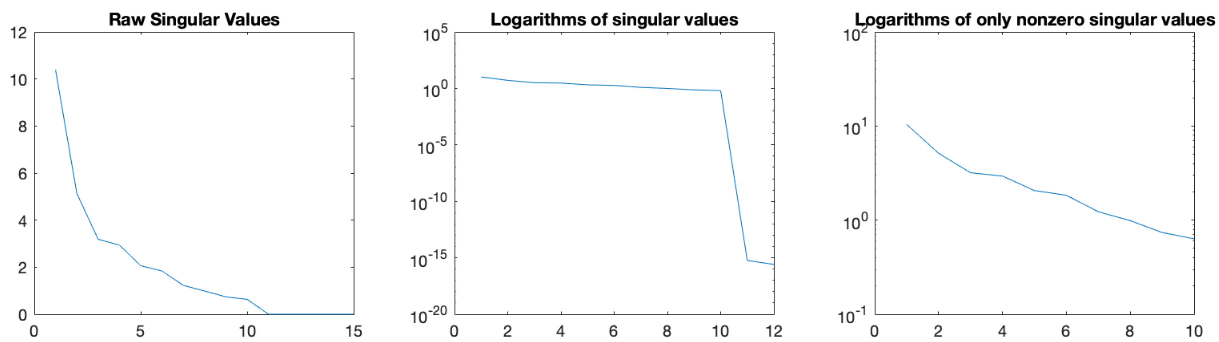
(a) The result of *spy("HELLO" matrix)*



(b) (The list of singular values of the matrix)

```
sigmas =

  Columns 1 through 11

   10.3833    5.1318    3.1878    2.9383    2.0625    1.8376    1.2270    0.9879    0.7380    0.6329    0.0000

  Columns 12 through 15

    0.0000         0         0         0


ans =

    "The rank of 'HELLO' matrix is 10."
```

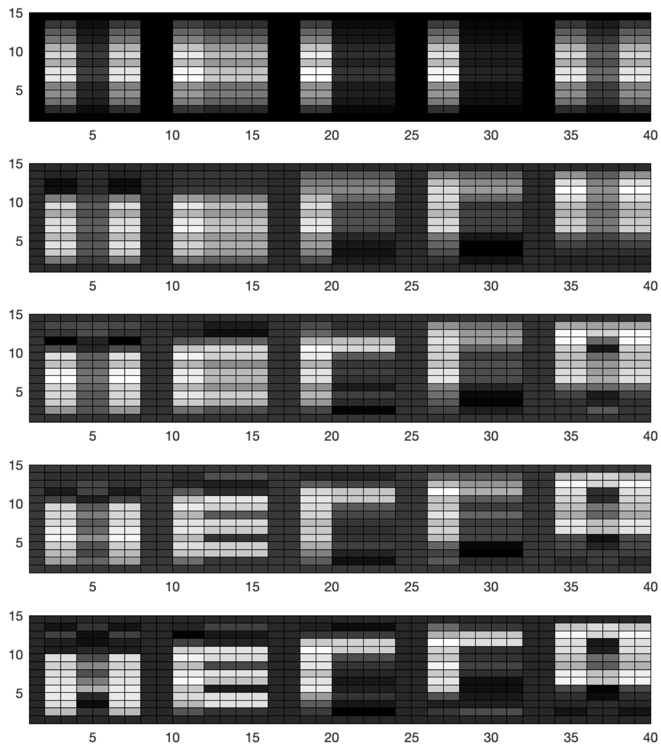
(plot and semilogy)

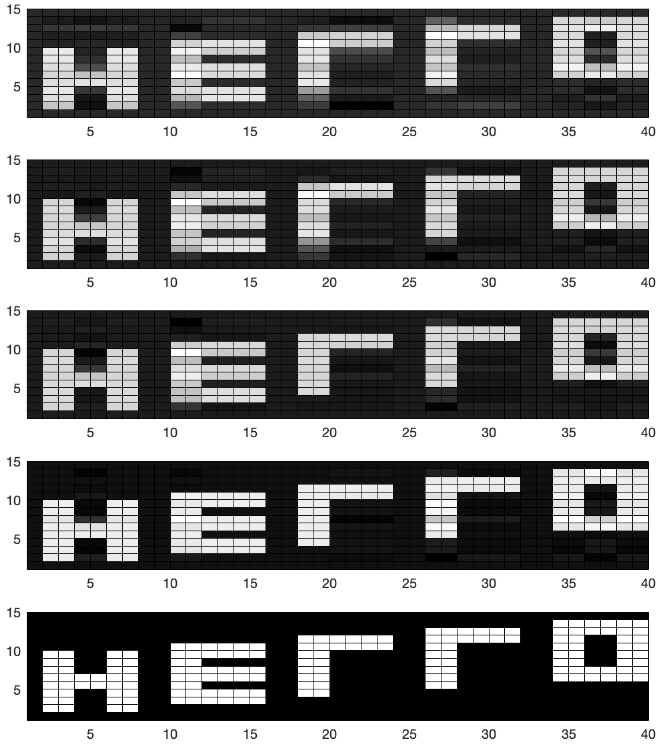


Mathematically, the exact rank of matrix is 10. Note that each letter has 2 independent column vectors and the last position of ‘1’ in each column vectors in each letter differs by 1. Hence, $2 \times 5 = 10$ independent column vectors span the column space.

In fact, we can check out in the first picture under (b) that $rank(mat)$ yields 10. Also, when counting nonzero singular values in the picture, 0.6329 is the last nonzero singular values, so that 10 nonzero singular values exist in total.

(c) The following is low rank approximation from 1 to 10.



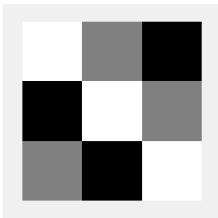


Problem 3

```
%%%% Problem 3 %%%%
%%%% 3-(1)

% Change working directory
cd '/Users/Kook/Desktop/mat_workspace/hw2';
load('Check.mat');
figure; imshow(Check);

%%%% 3-(2)

% Apply SVD to Check matrix
[u,s,v] = svd(Check);
s = diag(s);

% Save u, s, v as mat files
```

```
save('Check_u.mat', 'u');
save('Check_s.mat', 's');
save('Check_v.mat', 'v');

% Calculate smalles amount of data needed in this method
comp_entry = rank(Check) * (1 + 2 * length(s));
sprintf("In total, %d entries are required.", comp_entry)
sprintf("Compression rate is %f.", comp_entry/3600)

%%%% 3-(3)
load('Check_u.mat');
load('Check_s.mat');
load('Check_v.mat');

frame = zeros(60);
for n=1:60
    frame = frame + s(n)*u(:,n)*v(:,n)';
end
figure; imshow(frame);
```

(a)

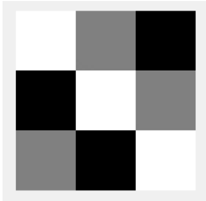


(b)

```
ans =

    "In total, 363 entries are required."


ans =

    "Compression rate is 0.100833."
```

Since the rank of the matrix is 3, only 3 left and right singular vectors with 3 singular values are enough to recover the original matrix. Hence, the smallest number of entries to store as data is computed like $rank(matrix) \cdot (1 + 60 + 60)$, where 60 is the length of singular vectors and 1 corresponds to singular values.

(c)



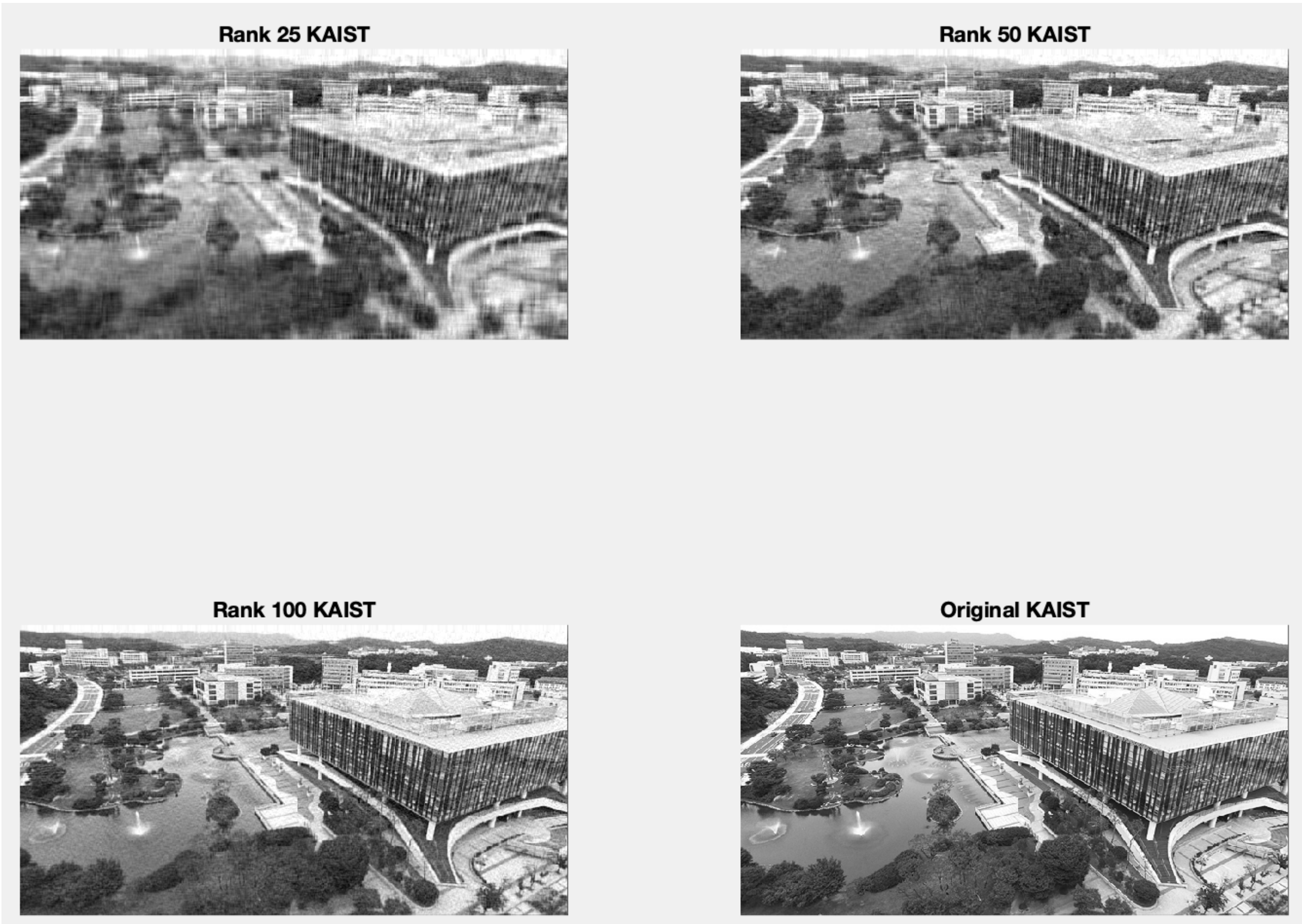
Comparing (c) with (a), we can conclude that the method in (b) recover the original data without loss of data.

Problem 4

```
%%%% Problem 4 %%%%
%%%% 4-(1)

% Convert kaist.jpg from rgb2 to gray
kaist = rgb2gray(imread('kaist.jpg'));
kaist = im2double(kaist);
tmp = size(kaist); row = tmp(1); col = tmp(2);

% Apply svd to kaist
[u,s,v] = svd(kaist);
diag_s = diag(s);

% Rank approximation
low_rank_array = zeros(row, col, 3);
ranks = [25, 50, 100];
figure;

for n=1:3
    low_diag = diag_s; low_diag(ranks(n)+1:end)=0;
    low_s = s; low_s(1:length(diag_s), 1:length(diag_s)) = diag(low_diag);

    low_kaist = u * low_s * v';
    low_rank_array(:,:,n) = low_kaist;

    subplot(2,2,n), imshow(low_kaist);
    title(sprintf('Rank %d KAIST', ranks(n)));
end
subplot(2,2,4), imshow(kaist), title('Original KAIST');

%%%% 4-(2)
```

```
original_norm = norm(kaist, 'fro');

for n=1:3
    rel_error = norm(kaist-low_rank_array(:,:,n), 'fro')/original_norm;
    sprintf("Relative Error when Rank-%d approximation : %f", ...
    ranks(n), rel_error)
end

%%%% 4-(3)
for n=1:3
    comp_rate = ranks(n)*(1+row+col)/(row*col);
    sprintf("Compression Rate when Rank-%d approximation : %f", ...
    ranks(n), comp_rate)
end
```

(a)



(b)

```
ans =

    "Relative Error when Rank-25 approximation : 0.183235"


ans =

    "Relative Error when Rank-50 approximation : 0.139969"


ans =

    "Relative Error when Rank-100 approximation : 0.094135"
```

(c)

```
ans =

    "Compression Rate when Rank-25 approximation : 0.110650"


ans =

    "Compression Rate when Rank-50 approximation : 0.221300"


ans =

    "Compression Rate when Rank-100 approximation : 0.442599"
```