Project Two Template

MAT-350: Applied Linear Algebra

Student Name:

Matthew Dunfee

Date:

2/21/2025

Problem 1

Use the svd() function in MATLAB to compute A_1, the **rank-1 approximation of A.** Clearly state what A_1 is, rounded to 4 decimal places. Also, **compute** the root-mean square error (RMSE) between A and A_1.

Solution:

```
%code
A = [
     1 2 3;
     3 3 4;
     5 6 7;
     ];

[U, S, V] = svd(A)
```

```
U = 3×3
   -0.2904    0.9504   -0.1114
   -0.4644   -0.2418   -0.8520
   -0.8367   -0.1957    0.5115
S = 3×3
   12.5318         0         0
         0    0.9122         0
         0         0    0.3499
V = 3×3
   -0.4682   -0.8261   -0.3136
   -0.5581    0.0012    0.8298
   -0.6851    0.5635   -0.4616
```

```
A_rank_1 = S(1,1)*U(:,1)*V(:,1).'
```

```
A_rank_1 = 3×3
    1.7039    2.0313    2.4935
    2.7243    3.2477    3.9867
    4.9087    5.8517    7.1832
```

```
error = A - A_rank_1
```

```
error = 3×3
```

```
   -0.7039   -0.0313    0.5065
    0.2757   -0.2477    0.0133
    0.0913    0.1483   -0.1832
```

```
rmse_value1 = sqrt(mean(error(:).^2))
```

```
rmse_value1 =
0.3257
```

Problem 2

Use the svd() function in MATLAB to compute A_2, the **rank-2 approximation of** A. Clearly state what A_2 is, rounded to 4 decimal places. Also, **compute** the root-mean square error (RMSE) between A and A_2. Which approximation is better, A_1 or A_2? Explain.

Solution:

```
%code
A_rank_2 = A_rank_1 + S(2,2)*U(:,2)*V(:,2).'
```

```
A_rank_2 = 3×3
    0.9878    2.0324    2.9820
    2.9065    3.2474    3.8624
    5.0561    5.8515    7.0826
```

```
error = A - A_rank_2
```

```
error = 3×3
    0.0122   -0.0324    0.0180
    0.0935   -0.2474    0.1376
   -0.0561    0.1485   -0.0826
```

```
rmse_value2 = sqrt(mean(error(:).^2))
```

```
rmse_value2 =
0.1166
```

Explain: rmse_value2 is better becuase is will result in greater compression. remse_value1 would be better if you wanted to peserve losslessness.

Problem 3

For the 3×3 matrix A, the singular value decomposition is $A = USV'$ where $U = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3]$. Use MATLAB to **compute** the dot product $d_1 = dot(\mathbf{u}_1, \mathbf{u}_2)$.

Also, use MATLAB to **compute** the cross product $\mathbf{c} = cross(\mathbf{u}_1, \mathbf{u}_2)$ and dot product $d_2 = dot(\mathbf{c}, \mathbf{u}_3)$. Clearly state the values for each of these computations. Do these values make sense? **Explain**.

Solution:

```
%code
d1 = dot(U(:,1), U(:,2))
```

```
d1 =
1.6653e-16
```

```
c = cross(U(:,1), U(:,2))
```

```
c = 3×1
   -0.1114
   -0.8520
    0.5115
```

```
d2 = dot(c, U(:,3))
```

```
d2 =
1.0000
```

Explain:

d1: The dot product of two orthogonal vectors should be zero.

c: The cross product of two vectors in 3d results in a vector that is perpendicular to both input vectors.

d2: For a properly oriented orthonormal basis in 3d, the cross product of the first two basis vectors should either be equal to or the negative of the third basis vector.

Problem 4

Using the matrix $U = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3]$, d**etermine whether or not the columns of** U **span** $\mathbb{R}^3$. **Explain your approach.**

Solution:

```
%code
rref(U)
```

```
ans = 3×3
     1     0     0
     0     1     0
     0     0     1
```

Explain: The rank is equal to the demintions of the space. Therefore its spans the space.

Problem 5

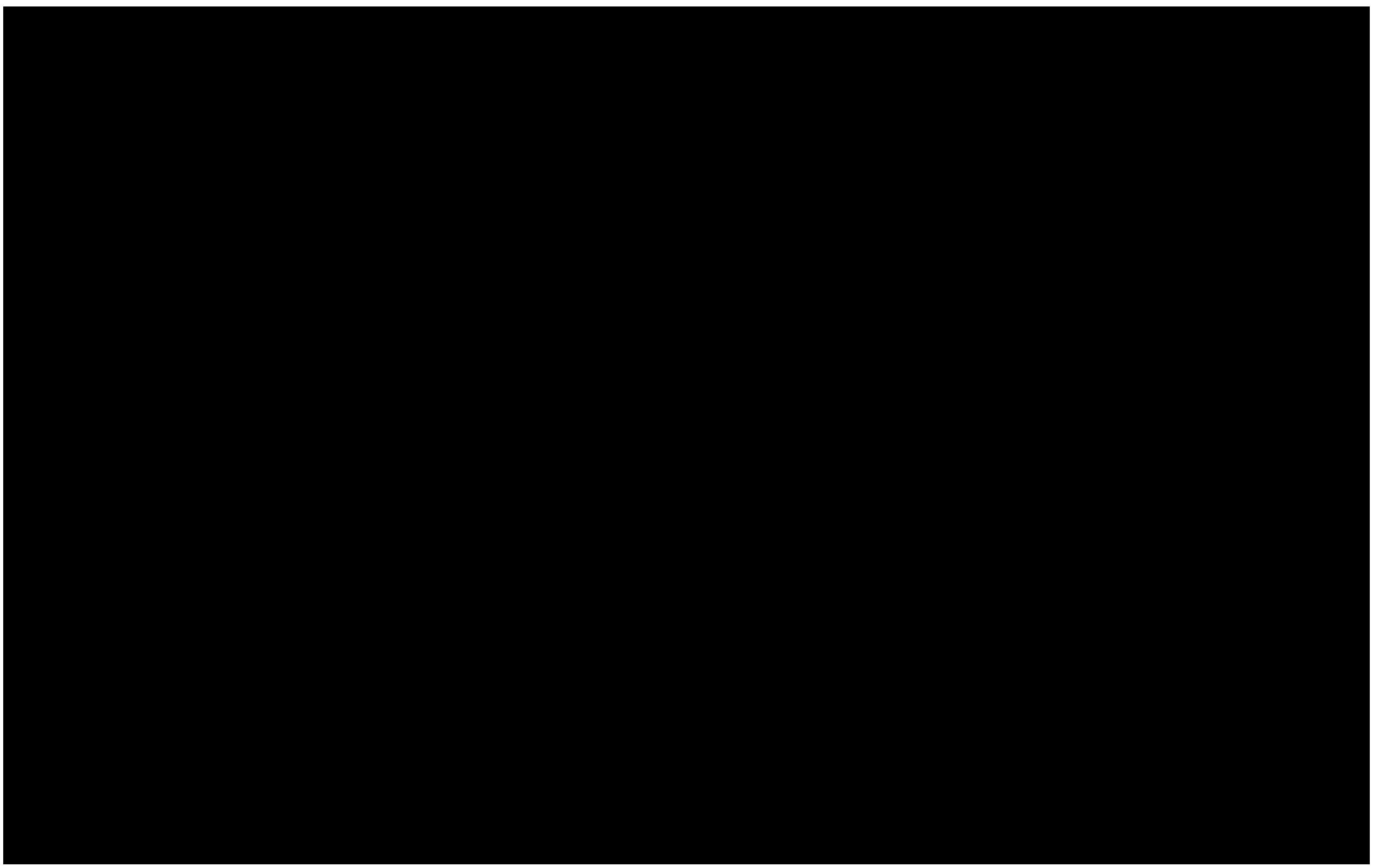
Use the MATLAB imshow() function to load and display the image A stored in the image.mat file, available in the Project Two Supported Materials area in Brightspace. For the loaded image, **derive the value of** k that will result in a compression ratio of $CR \approx 2$. For this value of k, **construct the rank-*k* approximation of the image**.

Solution:

```
%code
% Load variables
MAT350ProjectTwoMATLABImage = load("C:\Users\Matthew\Documents\MATLAB\MAT 350 Project Two MATLAB Image.mat");
myImage = MAT350ProjectTwoMATLABImage.A;
clear MAT350ProjectTwoMATLABImage;

% Display results
myImage
```

```
myImage = 2583×4220 uint8 matrix
   23   23   31   34   22   22   35   31   30   29   32   34   31   28   30   31 ···
   30   30   31   36   30   25   31   31   29   25   24   30   38   41   38   36
   38   33   23   21   19   16   25   35   31   26   18   23   37   41   35   32
   29   30   29   29   24   20   26   28   31   30   21   21   31   32   27   29
   20   24   32   37   32   32   33   17   26   30   26   26   34   33   28   31
   26   23   24   26   24   34   38   17   22   27   28   30   37   36   30   28
   30   29   25   25   24   31   36   23   25   28   29   30   32   33   30   27
   32   35   29   23   21   22   30   31   29   30   32   29   25   29   34   34
   27   25   22   27   28   24   28   30   23   35   34   26   21   25   33   32
   23   20   16   23   29   29   32   31   29   32   25   20   16   18   25   25
     ⋮
```

```
imshow(myImage)
```



```
function [k, U, S, V] = problem5(c, myImage)
    [m, n] = size(myImage);

    % this is the area that I needed the extension for.
    % originally I had c in the numerator instead of the denominator.
    % Simple mistake that I just needed some time to catch.
    computed_k = floor((m*n)/(c*(m+n+1)));
    k = min(computed_k, min(m,n));

    [U, S, V] = svd(double(myImage));
end

% This function computes the rank-k approximation recursively.
function approx = recursiveApproximation(k, U, S, V)
    % Base case: if k is 0, return a zero matrix with appropriate dimensions.
    if k == 0
        approx = zeros(size(U,1), size(V,1));
    else
        % Recursively compute the rank-(k-1) approximation and add the kth term.
        approx = recursiveApproximation(k-1, U, S, V) + S(k,k) * U(:,k) * V(:,k)';
    end
end
% moved cr 2 into problem 7 for readability
```

Explain:

being in the image

compute k for desired compress ratio

create a rank k approximation

Problem 6

Display the image and compute the root mean square error (RMSE) between the approximation and the original image. Make sure to include a copy of the approximate image in your report.

Solution:

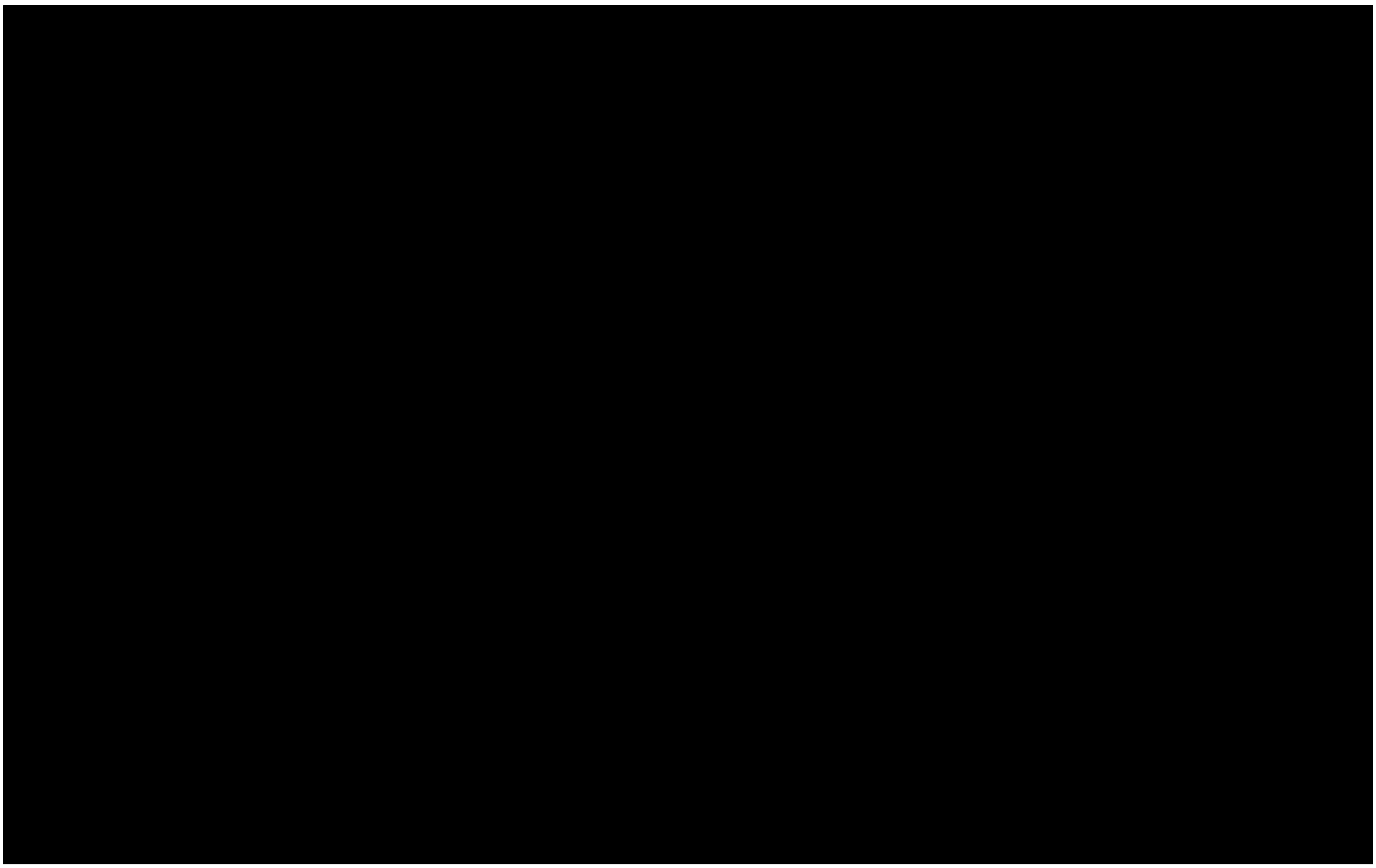
```
%code
function rmse_val = problem6(approx, myImage)
    imshow(approx, []);
    title('Approximate Image');

    error = double(myImage) - approx;
    rmse_val = sqrt(mean(error(:).^2));
end
```

Problem 7

Repeat Problems 5 and 6 for $CR \approx 10$, $CR \approx 25$, and $CR \approx 75$. **Explain** what trends you observe in the image approximation as CR increases and provide your recommendation for the best CR based on your observations. Make sure to include a copy of the approximate images in your report.

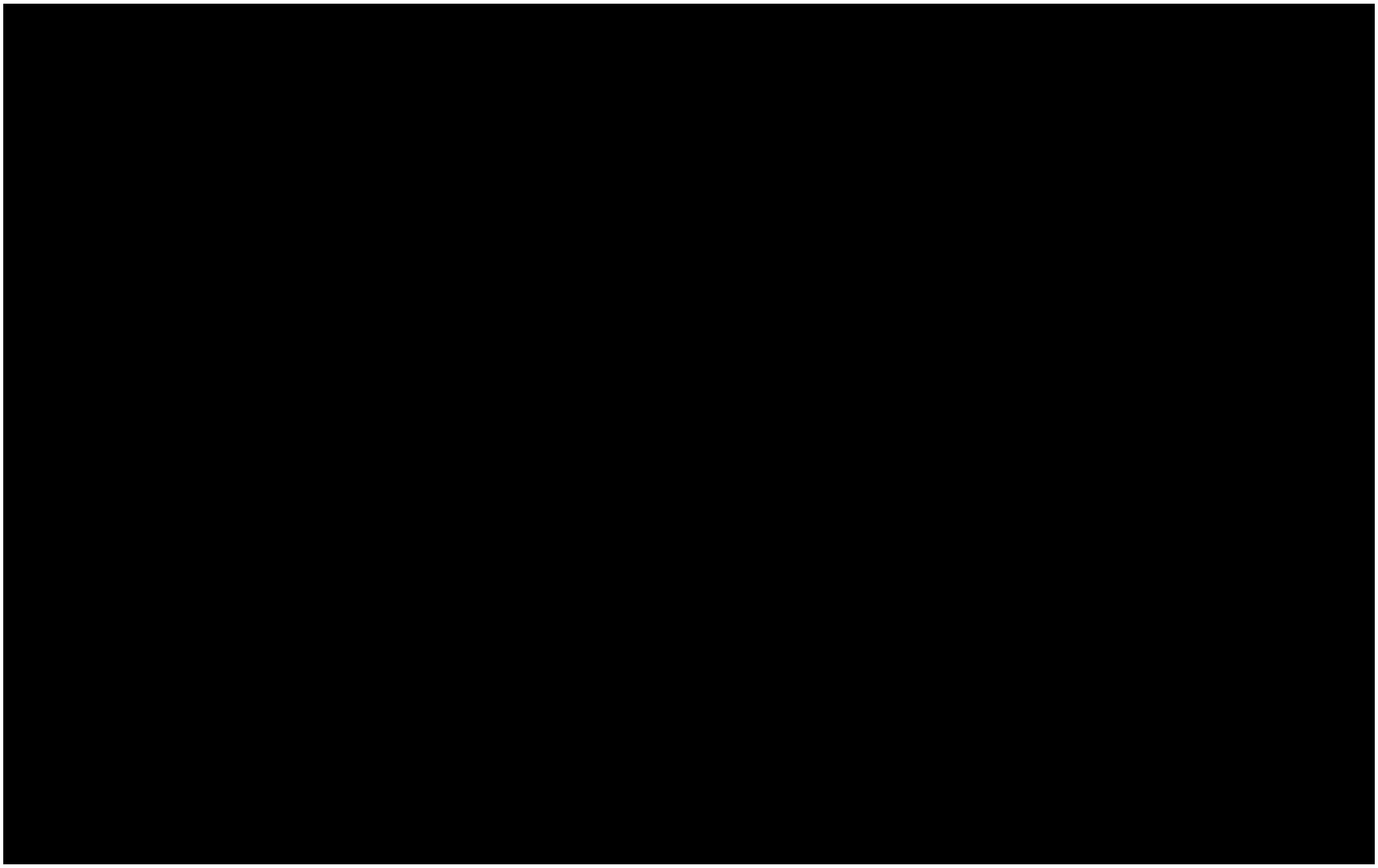
Solution:

```
%code
%2
[k, U, S, V] = problem5(2, myImage);
approxImg = recursiveApproximation(k, U, S, V);
rmse_value = problem6(approxImg, myImage)
```



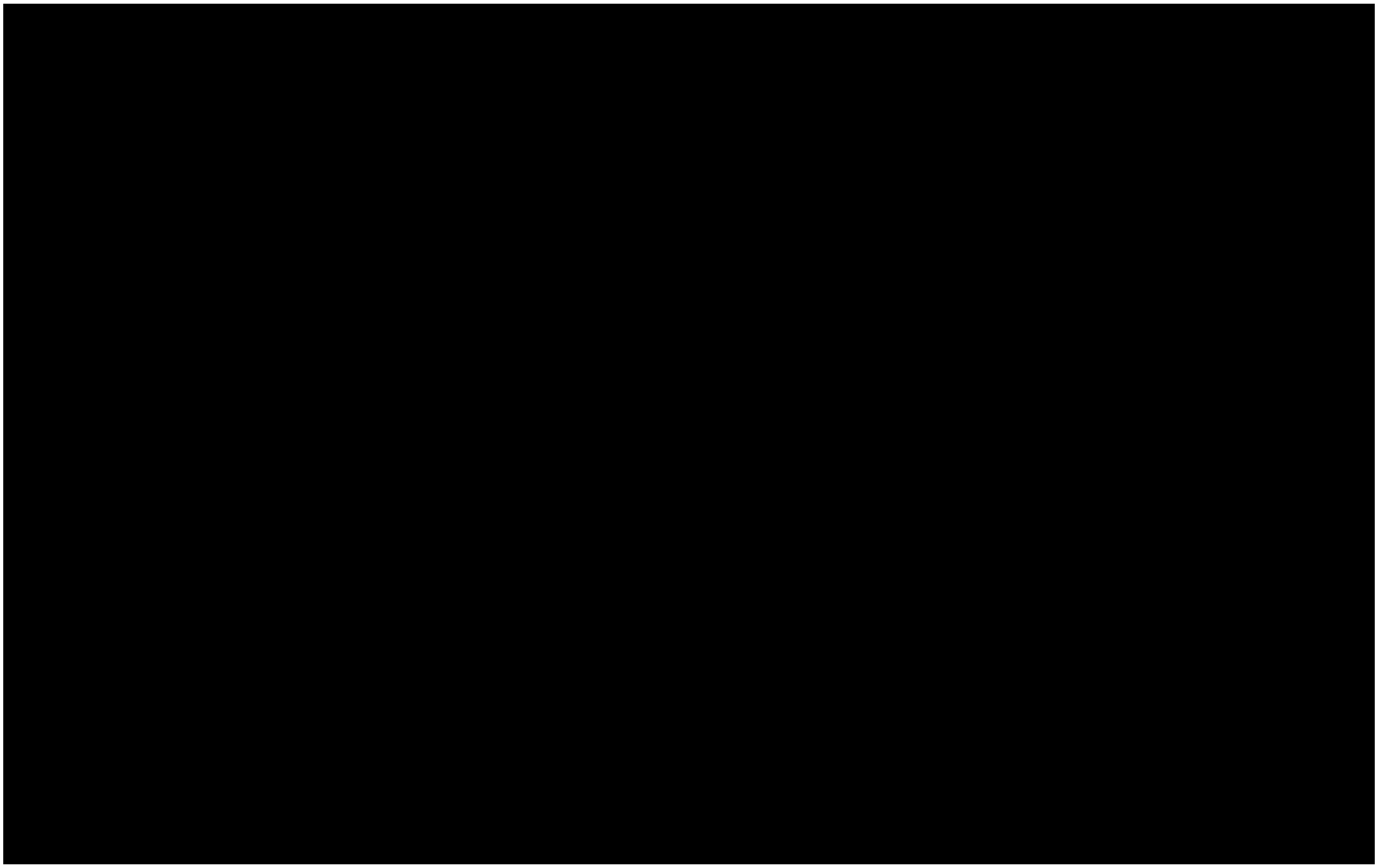
```
rmse_value =
3.1539
```

```

%10
[k, U, S, V] = problem5(10, myImage);
approxImg = recursiveApproximation(k, U, S, V);
rmse_value = problem6(approxImg, myImage)
```



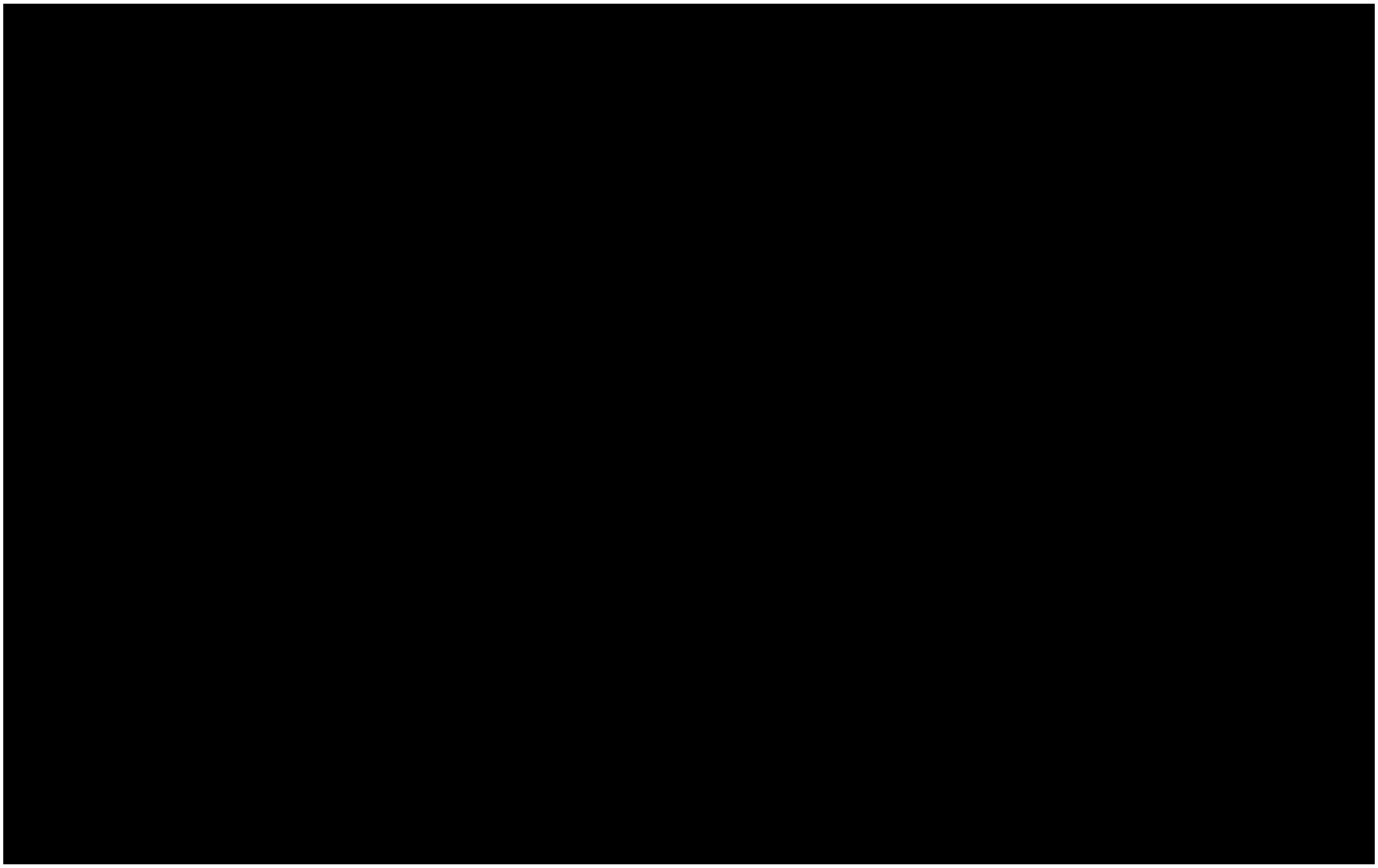
```
rmse_value =
8.2118
```

```
%25
[k, U, S, V] = problem5(25, myImage);
approxImg = recursiveApproximation(k, U, S, V);
rmse_value = problem6(approxImg, myImage)
```



```
rmse_value =
12.3039
```

```
%75
[k, U, S, V] = problem5(75, myImage);
approxImg = recursiveApproximation(k, U, S, V);
rmse_value = problem6(approxImg, myImage)
```



```
rmse_value =
18.2656
```

Explain:

After fixing the k value from problem 5, now it is working correctly. We can see the correlation of compression to distortion in the images as we increase from 10 to 75 percent in ascending order.