Project Two Template

MAT-350: Applied Linear Algebra

Student Name

Date

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
% Compute SVD
clear; close all; clc;

%% Define matrix A)
A = [1 2 3;
     3 3 4;
     5 6 7];

fprintf('Matrix A:\n'); disp(A);
```

```
Matrix A:
     1     2     3
     3     3     4
     5     6     7
```

```

%% ------------------- Problem 1 -------------------
% Use svd() to compute the rank-1 approximation A1 and RMSE

[U,S,V] = svd(double(A));   % SVD

k1 = 1;
A1 = U(:,1:k1) * S(1:k1,1:k1) * V(:,1:k1)';   % rank-1 approx
A1_rounded = round(A1,4);

rmse1 = sqrt(mean((double(A(:)) - A1(:)).^2));

fprintf('\n--- Problem 1 ---\n');
```

```
--- Problem 1 ---
```

```
disp('A1 (rank-1, rounded to 4 decimals):');
```

```
A1 (rank-1, rounded to 4 decimals):
```

```
disp(A1_rounded);
```

```
    1.7039    2.0313    2.4935
    2.7243    3.2477    3.9867
    4.9087    5.8517    7.1832
```

```
fprintf('RMSE(A, A1) = %.7f\n', rmse1);
```

```
RMSE(A, A1) = 0.3256597
```

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
k2 = 2;
A2 = U(:,1:k2) * S(1:k2,1:k2) * V(:,1:k2)';   % rank-2 approx
A2_rounded = round(A2,4);

rmse2 = sqrt(mean((double(A(:)) - A2(:)).^2));

fprintf('\n--- Problem 2 ---\n');
```

```
--- Problem 2 ---
```

```
disp('A2 (rank-2, rounded to 4 decimals):');
```

```
A2 (rank-2, rounded to 4 decimals):
```

```
disp(A2_rounded);
```

```
    0.9878    2.0324    2.9820
    2.9065    3.2474    3.8624
    5.0561    5.8515    7.0826
```

```
fprintf('RMSE(A, A2) = %.7f\n', rmse2);
```

```
RMSE(A, A2) = 0.1166416
```

```

% Which is better?
if rmse2 < rmse1
    fprintf('A2 is better than A1 because RMSE(A,A2)=%.7f <
RMSE(A,A1)=%.7f\n', rmse2, rmse1);
else
```

```
    fprintf('A1 is better (unexpected): RMSE(A,A1)=%.7f, RMSE(A,A2)=%.7f\n',
rmse1, rmse2);
end
```

```
A2 is better than A1 because RMSE(A,A2)=0.1166416 < RMSE(A,A1)=0.3256597
```

Explain:

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
u1 = U(:,1);
u2 = U(:,2);
u3 = U(:,3);

d1 = dot(u1,u2);
c = cross(u1,u2);
d2 = dot(c,u3);

fprintf('\n--- Problem 3 ---\n');
```

```
--- Problem 3 ---
```

```
fprintf('d1 = dot(u1,u2) = %.16f\n', d1);
```

```
d1 = dot(u1,u2) = 0.0000000000000002
```

```
disp('c = cross(u1,u2) ='); disp(c);
```

```
c = cross(u1,u2) =
   -0.1114
   -0.8520
    0.5115
```

```
fprintf('d2 = dot(c,u3) = %.16f\n', d2);
```

```
d2 = dot(c,u3) = 1.0000000000000002
```

Explain:

Problem 4

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

Solution:

```
%code
rankU = rank(U);
fprintf('\n--- Problem 4 ---\n');
```

```
--- Problem 4 ---
```

```
fprintf('rank(U) = %d\n', rankU);
```

```
rank(U) = 3
```

```
if rankU == 3
    fprintf('Conclusion: columns of U span R^3 (they are linearly
independent).\n');
else
    fprintf('Conclusion: columns of U do NOT span R^3.\n');
end
```

```
Conclusion: columns of U span R^3 (they are linearly independent).
```

Explain:

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
imgFile = 'image.mat';
if ~isfile(imgFile)
    warning('image.mat not found in current folder. Problems 5-7 will be
skipped.');
else
    S = load(imgFile);
    vars = fieldnames(S);
    img_found = false;
    for i = 1:numel(vars)
        cand = S.(vars{i});
        if isnumeric(cand) && (ndims(cand) == 2 || ndims(cand) == 3)
            img = cand;
```

```
            img_found = true;
            img_name = vars{i};
            break;
        end
    end
    if ~img_found
        error('No suitable numeric image variable found inside image.mat.');
    end

    fprintf('\n--- Problems 5-7: Image compression ---\n');
    fprintf('Image variable detected: %s\n', img_name);

    img_double = double(img);
    [m,n,chan] = size(img_double);
    if ndims(img_double) == 2
        chan = 1;
    end
```

```
--- Problems 5-7: Image compression ---
Image variable detected: A
```

Explain:

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.

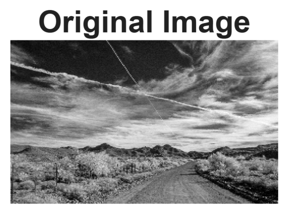
Solution:

```
%code
    compute_k = @(m,n,CR) max(1, round( (m*n) / (CR*(m + n + 1)) ));

    CR_list = [2, 10, 25, 75];
    results = zeros(numel(CR_list), 3); % columns: CR, k, RMSE

    % Precompute SVDs to speed up: if grayscale, one SVD; if RGB, per
channel SVD
    if chan == 1
        [Uim,Sim,Vim] = svd(img_double,'econ');
        for idx = 1:numel(CR_list)
            CR = CR_list(idx);
            k = compute_k(m,n,CR);
            k = min(k, min(m,n));
            % construct approx
            approx = Uim(:,1:k) * Sim(1:k,1:k) * Vim(:,1:k)';
            % RMSE
            rmse_img = sqrt(mean((img_double(:) - approx(:)).^2));
            % prepare displayable image
            if max(approx(:)) <= 1 && min(approx(:)) >= 0
```

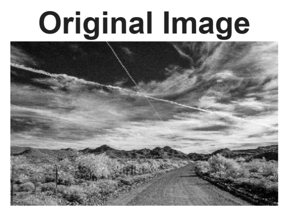
```
                approx_save = im2uint8(mat2gray(approx));
                orig_save = im2uint8(mat2gray(img_double));
            else
                approx_save = uint8(min(max(round(approx),0),255));
                orig_save = uint8(min(max(round(img_double),0),255));
            end
            fname = sprintf('approx_CR_%d_k_%d.png', CR, k);
            imwrite(approx_save, fname);
            fprintf('CR=%d -> k=%d, RMSE=%.6f, saved %s\n', CR, k, rmse_img,
fname);
            % show images
            figure('Name',sprintf('Original (CR=%d)',CR));
imshow(orig_save); title('Original Image');
            figure('Name',sprintf('Approx CR=%d',CR)); imshow(approx_save);
title(sprintf('Approx CR=%d, k=%d, RMSE=%.4f', CR, k, rmse_img));
            results(idx,:) = [CR, k, rmse_img];
        end
    else
        % RGB: compute SVD for each channel once and reuse
        Ucell = cell(1,chan); Scell = cell(1,chan); Vcell = cell(1,chan);
        for c = 1:chan
            [Ucell{c}, Scell{c}, Vcell{c}] = svd(img_double(:,:,c),'econ');
        end
        for idx = 1:numel(CR_list)
            CR = CR_list(idx);
            k = compute_k(m,n,CR);
            k = min(k, min(m,n));
            approx = zeros(size(img_double));
            for c = 1:chan
                Uk = Ucell{c}(:,1:k);
                Sk = Scell{c}(1:k,1:k);
                Vk = Vcell{c}(:,1:k);
                approx(:,:,c) = Uk * Sk * Vk';
            end
            rmse_img = sqrt(mean((img_double(:) - approx(:)).^2));
            if max(approx(:)) <= 1 && min(approx(:)) >= 0
                approx_save = im2uint8(mat2gray(approx));
                orig_save = im2uint8(mat2gray(img_double));
            else
                approx_save = uint8(min(max(round(approx),0),255));
                orig_save = uint8(min(max(round(img_double),0),255));
            end
            fname = sprintf('approx_CR_%d_k_%d.png', CR, k);
            imwrite(approx_save, fname);
            fprintf('CR=%d -> k=%d, RMSE=%.6f, saved %s\n', CR, k, rmse_img,
fname);
            % show images
            figure('Name',sprintf('Original (CR=%d)',CR));
imshow(orig_save); title('Original Image');
```

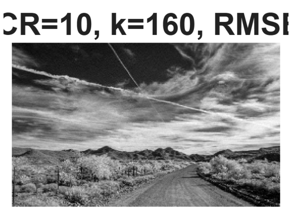
```
            figure('Name',sprintf('Approx CR=%d',CR)); imshow(approx_save);
title(sprintf('Approx CR=%d, k=%d, RMSE=%.4f', CR, k, rmse_img));
            results(idx,:) = [CR, k, rmse_img];
        end
    end
```

```
CR=2 -> k=801, RMSE=3.153904, saved approx_CR_2_k_801.png
```




```
CR=10 -> k=160, RMSE=8.211776, saved approx_CR_10_k_160.png
```

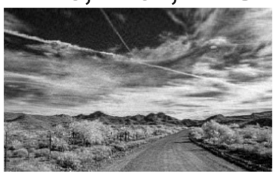





```
CR=25 -> k=64, RMSE=12.303851, saved approx_CR_25_k_64.png
```

Original Image

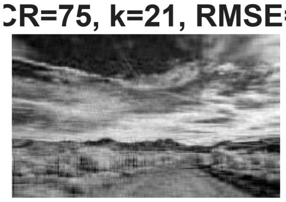
CR=25, k=64, RMSE=



```
CR=75 -> k=21, RMSE=18.265610, saved approx_CR_75_k_21.png
```

Original Image

CR=75, k=21, RMSE=



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
    T = array2table(results,'VariableNames',{'CR','k','RMSE'});
    disp('Summary table (CR, k, RMSE):');
    disp(T);
end
```

```
Summary table (CR, k, RMSE):
    CR     k       RMSE
    __    ___    ______

     2    801    3.1539
    10    160    8.2118
    25     64    12.304
    75     21    18.266
```

```
%% End of script
fprintf('\nAll computations complete. Copy outputs and saved approx images
into your report.\n');
```

```
All computations complete. Copy outputs and saved approx images into your report.
```

Explain: