Project Two

MAT-350: Applied Linear Algebra

Nathan Hallam

February 25th, 2024

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:

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

```
A = 3×3
     1     2     3
     3     3     4
     5     6     7
```

```
% Get SVD
[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
```

```
% Rank-1 Approx
A1 =  U(:,1) * S(1, 1) * V(:,1)'
```

```
A1 = 3×3
    1.7039    2.0313    2.4935
    2.7243    3.2477    3.9867
    4.9087    5.8517    7.1832
```

```
% Check Rank
rank(A1)
```

```
ans = 1
```

```
% Calculate error
RMSEA1 = rmse(A,A1,"all")
```

```
RMSEA1 = 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:

```
% Rank-2 Approx
A2 = A1 + S(2,2) * U(:,2) * V(:,2).'
```

```
A2 = 3×3
    0.9878    2.0324    2.9820
    2.9065    3.2474    3.8624
    5.0561    5.8515    7.0826
```

```
% Check Rank
rank(A2)
```

```
ans = 2
```

```
% Calculate Error
RMSEA2 = rmse(A, A2, "all")
```

```
RMSEA2 = 0.1166
```

Explain:

I believe the rank-2 is a better approximation as more information is incorporated into determining the approximation of each value. Comparing the A1 and A2 matrices we can visually see that A2 has generally closer values to the original matrix while A2's error is closer to zero.

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:

```
% Create vectors from U columns
u1 = U(:,1)
```

```
u1 = 3×1
   -0.2904
   -0.4644
   -0.8367
```

```
u2 = U(:,2)
```

```
u2 = 3×1
    0.9504
   -0.2418
   -0.1957
```

```
u3 = U(:,3)
```

```
u3 = 3×1
   -0.1114
   -0.8520
    0.5115
```

```
%C alculate dot product
d1 = dot(u1, u2)
```

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

```
%Calculate cross product
c = cross(u1, u2)
```

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

```
% Calculate dot product with cross product
d2 = dot(c, u3)
```

```
d2 = 1.0000
```

Explain:

The values returned by the dot product represent the individual importance of each value when compressing. The returns of d1 and d2 show that the first 2 columns of U are lower in value being close to zero, while d2 is one having a high value. We can also see this in the matrices A1 and A2 as columns 1 and 2 have a large margin of error to the original matrix while column 3 is much closer overall.

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:

```
% Reconstruct U from vectors
U = [u1, u2, u3]
```

```
U = 3×3
   -0.2904    0.9504   -0.1114
   -0.4644   -0.2418   -0.8520
   -0.8367   -0.1957    0.5115
```

```
% Compute pivot columns
reducedU = rref(U)
```

```
reducedU = 3×3
     1     0     0
     0     1     0
     0     0     1
```

```
% Check rank
rank(reducedU)
```

```
ans = 3
```

Explain:

Each column matrix is combined into a singular matrix and row reduced to locate the pivot columns. Since we are checking if U spans $\mathbb{R}^3$ we will need to look to see if the rank of U matches the rank of $\mathbb{R}^3$, which in this case is 3. By checking the rank of the row reduced U we can see that U is in teh span of $\mathbb{R}^3$ as the rank of row reduced U is also 3.

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:

```
% Load Image
A = load("Image.mat").A
```

```
A = 2583x4220 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
    ⋮
```

```
% Show Original Image
imshow(A)
```

```
% Get Image SVD
[U, S, V] = svd(double(A))
```

```
U = 2583×2583
   -0.0106   -0.0360   -0.0006    0.0032   -0.0032    0.0041   -0.0066    0.0022 ···
   -0.0105   -0.0361   -0.0006    0.0030   -0.0035    0.0049   -0.0062    0.0020
   -0.0105   -0.0362   -0.0006    0.0034   -0.0037    0.0042   -0.0064    0.0025
   -0.0105   -0.0362   -0.0009    0.0029   -0.0035    0.0052   -0.0056    0.0028
   -0.0106   -0.0361   -0.0011    0.0034   -0.0035    0.0046   -0.0061    0.0022
   -0.0106   -0.0363   -0.0011    0.0031   -0.0030    0.0049   -0.0061    0.0031
   -0.0106   -0.0364   -0.0008    0.0032   -0.0032    0.0043   -0.0057    0.0033
   -0.0106   -0.0365   -0.0006    0.0029   -0.0033    0.0050   -0.0052    0.0031
   -0.0106   -0.0366   -0.0007    0.0033   -0.0031    0.0040   -0.0053    0.0033
   -0.0106   -0.0368   -0.0009    0.0030   -0.0034    0.0044   -0.0052    0.0032
     ⋮
S = 2583×4220
10^5 ×
    4.0600         0         0         0         0         0         0         0 ···
         0    0.8702         0         0         0         0         0         0
         0         0    0.4169         0         0         0         0         0
         0         0         0    0.4104         0         0         0         0
         0         0         0         0    0.3405         0         0         0
         0         0         0         0         0    0.2992         0         0
         0         0         0         0         0         0    0.2550         0
         0         0         0         0         0         0         0    0.2268
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
     ⋮
V = 4220×4220
```

```
   -0.0130    0.0044   -0.0358    0.0028   -0.0085    0.0177    0.0128   -0.0163 ···
   -0.0130    0.0045   -0.0357    0.0024   -0.0079    0.0184    0.0134   -0.0162
   -0.0130    0.0045   -0.0359    0.0025   -0.0078    0.0181    0.0124   -0.0168
   -0.0130    0.0046   -0.0361    0.0030   -0.0087    0.0185    0.0125   -0.0158
   -0.0130    0.0045   -0.0366    0.0032   -0.0095    0.0182    0.0116   -0.0149
   -0.0129    0.0046   -0.0369    0.0034   -0.0095    0.0185    0.0112   -0.0151
   -0.0130    0.0047   -0.0372    0.0046   -0.0104    0.0178    0.0103   -0.0141
   -0.0130    0.0048   -0.0377    0.0045   -0.0097    0.0179    0.0108   -0.0145
   -0.0130    0.0046   -0.0375    0.0049   -0.0094    0.0170    0.0102   -0.0147
   -0.0130    0.0045   -0.0379    0.0043   -0.0090    0.0166    0.0095   -0.0142
     ⋮
```

```
% Get image dimensions
[h, w] = size(A)
```

```
h = 2583
w = 4220
```

```
% Find k
% 2 = (h * w) / k * (h + w + 1)
% 2 * k * (h + w + 1) = (h * w)
% k = (h * w) / (2 * (h + w + 1))
k = round((h * w) / (2 * (h + w + 1)))
```

```
k = 801
```

```
% Calculate Rank-k
Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'
```

```
Ak = 2583×4220
   26.4896   27.2541   30.5810   28.9530   23.3828   25.7705   35.1037   29.3968 ···
   32.6831   34.0733   28.4258   30.5682   27.6882   28.4547   35.6629   31.2002
   35.5230   30.8250   18.7994   19.9743   19.8952   17.0400   24.6764   26.0911
   33.6440   29.7702   26.1173   29.8858   26.5095   15.9739   24.7017   25.8886
   27.7165   26.0012   30.5018   36.7547   35.3434   29.8915   34.5078   25.1416
   27.3996   25.5183   26.6092   29.4463   25.5795   28.8615   33.9147   23.0624
   32.0484   32.4889   27.5089   22.7250   20.3684   25.0803   33.3388   26.7700
   26.0954   32.2044   27.4183   18.1894   21.2836   28.1417   31.7244   26.2813
   23.2187   25.5412   22.1689   25.1362   29.2165   30.3222   34.5845   30.5812
   21.3048   20.9797   19.1568   25.5860   26.9030   24.8220   30.0068   30.1700
     ⋮
```

```
% Check Rank
rank(Ak)
```

```
ans = 801
```

```
% Round and convert to uint8 to be readable by program
Ak = uint8(round(Ak))
```

```
Ak = 2583x4220 uint8 matrix
   26   27   31   29   23   26   35   29   33   36   30   26   32   27   28   34 ···
   33   34   28   31   28   28   36   31   30   27   22   28   40   34   34   34
   36   31   19   20   20   17   25   26   30   23   22   27   35   35   37   34
   34   30   26   30   27   16   25   26   28   25   22   24   33   35   32   30
   28   26   31   37   35   30   35   25   25   25   23   27   33   33   27   27
   27   26   27   29   26   29   34   23   22   27   29   34   33   33   28   26
   32   32   28   23   20   25   33   27   27   29   35   33   27   28   31   29
```

```
26   32   27   18   21   28   32   26   29   32   36   31   28   31   34   34
23   26   22   25   29   30   35   31   30   35   34   27   20   24   34   34
21   21   19   26   27   25   30   30   28   34   29   20   16   22   29   32
  ⋮
```

Explain:

1: Load the image and determine the values of M & N (length and width)

2: Convert loaded image to double from uint8 to compute image SVD

3: Derive k by using conversion ratio equation alongside image height and width

CR = (m * n) / k * (m + n + 1)

2 = (2583 *4220) / k * (2583 + 4220 + 1)

2 = (10900260) / k * (6804)

13608 * k = 10900260

k = 10900260 / 13608

k = 129765 / 162

k = 801.0185

k = 801

4: Calculate and check rank of image at the given compression of 2

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:

```
imshow(Ak)
```

```
%Calculate Error
rmse(double(A), double(Ak), "all")
```

```
ans = 3.1664
```

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:

```
% Calculate rank given CR of 10
k = round((h * w) / (10 * (h + w + 1)))
```

```
k = 160
```

```
% Calculate Rank-160
Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'
```

```
Ak = 2583×4220
   29.0415   29.0508   30.3481   28.6840   28.4125   25.8146   26.5310   25.2709 ···
   29.4990   29.1068   29.9563   28.2658   28.2759   26.2124   27.5634   26.0996
   26.0110   25.7091   26.5596   24.7097   24.6572   23.0067   24.2242   22.2803
   28.3636   28.5688   29.5533   27.3636   27.6176   25.7857   27.0061   24.7262
   28.5023   28.4627   29.3196   27.9087   28.6184   26.5532   27.9189   25.7023
```

```
   26.6380   26.7335   27.2446   25.9346   26.6994   25.3844   27.4323   25.3185
   27.7105   26.9342   27.3695   25.8611   26.2616   24.8003   26.8496   24.7796
   26.6047   26.2936   26.6370   25.8935   26.9915   25.8782   27.7421   25.4926
   25.4761   25.3280   25.3720   24.6828   25.3964   24.6848   26.1862   23.9901
   24.0066   23.4106   23.1904   22.8845   23.3535   22.7318   23.9445   22.0238
     ⋮
```

```
% Round and convert to uint8 to be readable by program
Ak = uint8(round(Ak))
```

```
Ak = 2583x4220 uint8 matrix
   29   29   30   29   28   26   27   25   28   27   26   25   28   27   27   28 ···
   29   29   30   28   28   26   28   26   28   27   26   25   28   27   28   29
   26   26   27   25   25   23   24   22   24   24   24   23   26   25   27   28
   28   29   30   27   28   26   27   25   26   26   26   24   28   27   27   28
   29   28   29   28   29   27   28   26   27   26   27   25   28   27   27   28
   27   27   27   26   27   25   27   25   27   26   26   24   26   26   27   27
   28   27   27   26   26   25   27   25   26   25   26   24   26   26   27   27
   27   26   27   26   27   26   28   25   27   25   25   23   25   24   25   25
   25   25   25   25   25   25   26   24   25   24   24   21   23   23   23   23
   24   23   23   23   23   23   24   22   23   21   21   19   21   21   22   23
     ⋮
```

```
% Show compressed image at Rank-160 : CR 10
imshow(Ak)
```

```
%Calculate Error
rmse(double(A), double(Ak), "all")
```

```
ans = 8.2127
```

```
% Calculate rank given CR of 25
k = round((h * w) / (25 * (h + w + 1)))
```

```
k = 64
```

```
% Calculate Rank-64
Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'
```

```
Ak = 2583×4220
   24.9993   24.3447   22.0100   22.2062   21.9511   21.9226   23.0593   23.7963 ···
   25.2032   24.4691   22.2505   22.4141   22.3977   22.5232   23.5883   24.2551
   23.7727   23.0131   20.8102   21.1738   20.9707   21.3994   22.4306   23.1625
   24.3150   23.8209   21.6649   21.8529   21.8114   22.1064   23.1593   23.8830
   24.8134   24.0232   21.8824   22.0873   22.2178   22.3894   23.4309   24.2761
   24.1187   23.4903   21.2471   21.5617   21.6885   21.8974   22.7524   23.5456
   24.0810   23.2159   21.0505   21.3372   21.4513   21.7045   22.5713   23.3211
   25.0993   24.3390   22.2436   22.6802   23.0230   23.0038   23.7316   24.4781
   23.7390   22.9025   20.8467   21.1917   21.5129   21.5312   22.1834   23.0650
   22.2805   21.5541   19.4265   20.0297   20.3143   20.3821   20.8926   21.9840
     ⋮
```

```
% Round and convert to uint8 to be readable by program
Ak = uint8(round(Ak))
```

```
Ak = 2583x4220 uint8 matrix
   25   24   22   22   22   22   23   24   25   26   27   27   29   29   28   30 ···
   25   24   22   22   22   23   24   24   25   26   27   27   30   29   29   31
   24   23   21   21   21   21   22   23   24   25   26   26   29   28   28   29
   24   24   22   22   22   22   23   24   25   26   27   27   29   28   28   29
   25   24   22   22   22   22   23   24   25   26   27   27   29   29   28   29
   24   23   21   22   22   22   23   24   24   25   26   26   28   28   27   28
   24   23   21   21   21   22   23   23   24   25   26   26   28   27   26   27
   25   24   22   23   23   23   24   24   25   26   27   27   29   28   27   28
   24   23   21   21   22   22   22   23   24   25   26   25   27   27   26   27
   22   22   19   20   20   20   21   22   23   24   25   25   27   26   26   26
     ⋮
```

```
% Show compressed image at Rank-64 : CR 25
imshow(Ak)
```

```
%Calculate Error
rmse(double(A), double(Ak), "all")
```

```
ans = 12.3020
```

```
% Calculate rank given CR of 75
k = round((h * w) / (75 * (h + w + 1)))
```

```
k = 21
```

```
% Calculate Rank-21
Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'
```

```
Ak = 2583×4220
   26.2163   25.6337   24.9907   25.1005   25.5472   26.3732   27.4089   27.1685 ···
   26.5129   25.9007   25.2598   25.3699   25.8707   26.7271   27.7391   27.5112
   26.3007   25.6709   25.0385   25.1923   25.6988   26.5671   27.6553   27.4353
   27.0439   26.5387   25.9096   26.0592   26.5085   27.3315   28.3511   28.1032
   26.9099   26.2511   25.6362   25.6966   26.2742   27.0451   28.0548   27.8644
   26.6065   25.9442   25.3392   25.4571   26.0535   26.8352   27.8248   27.6089
   26.8196   26.1732   25.5741   25.6539   26.1657   26.8895   27.8945   27.6443
   27.2339   26.6413   26.0681   26.0830   26.6782   27.3193   28.2429   27.9771
   26.0167   25.3796   24.8159   24.7942   25.3968   25.9820   26.8889   26.6155
   27.3611   26.7141   26.1752   26.1811   26.7707   27.3992   28.3091   28.0254
     ⋮
```

```
% Round and convert to uint8 to be readable by program
Ak = uint8(round(Ak))
```

```
Ak = 2583x4220 uint8 matrix
   26   26   25   25   26   26   27   27   28   28   29   29   30   31   30   32 ...
   27   26   25   25   26   27   28   28   29   29   30   29   30   31   31   32
   26   26   25   25   26   27   28   27   29   29   29   29   30   31   31   32
   27   27   26   26   27   27   28   28   29   29   30   30   31   32   31   33
   27   26   26   26   26   27   28   28   29   29   30   30   31   32   31   33
   27   26   25   25   26   27   28   28   29   29   30   29   30   31   31   32
   27   26   26   26   26   27   28   28   29   29   30   29   30   31   31   32
   27   27   26   26   27   27   28   28   29   29   30   30   31   32   32   33
   26   25   25   25   25   26   27   27   28   28   29   28   30   30   30   32
   27   27   26   26   27   27   28   28   29   29   30   30   31   32   31   33
    ⋮
```

```
% Show compressed image at Rank-21 : CR 75
imshow(Ak)
```

```
%Calculate Error
rmse(double(A), double(Ak), "all")
```

```
ans = 18.2650
```

Explain:

A trend of decreasing rank and increasing Conversion Rates showed increasingly blurred images. As the images grow increasingly blurry as the conversion rate increases, so to does the image quality drop, with the most noticable drop in quality being between CR 25 and CR 75 but I presume this is due to the large 50 point gap in each images compression ratio.

With this in mind, in order to preserve the integrity of the images quality I would recommend going no higher than CR 25, but for general use CR 10 is better as it preserves more of the images quality while reducing the file size more than CR 2.