# MAT 343 Laboratory 8
# The SVD decomposition and Image Compression

In this laboratory session we will learn how to

1. Find the SVD decomposition of a matrix using MATLAB
2. Use the SVD to perform Image Compression.

## Introduction

Let $A$ be an $m \times n$ matrix ($m \geq n$). Then $A$ can be factored as

$$A = USV^T = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_m \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}_{m\times m} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \sigma_n \\ \vdots & \vdots & & 0 \\ 0 & 0 & \dots & 0 \end{bmatrix}_{m\times n} \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_n \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}^T_{n\times n} \tag{L8.1}$$

By convention $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$. $U$ and $V$ are orthogonal matrices (square matrices with orthonormal column vectors).
Note that the matrix $S$ is usually denoted by $\Sigma$, however we will use $S$ to be consistent with MATLAB notation.
One useful application of the SVD is the approximation of a large matrix by another matrix of lower rank. The lower-rank matrix contains less data and so can be represented more compactly than the original one. This is the idea behind one form of signal compression.

## The Frobenius Norm and Lower-Rank Approximation

The *Frobenius norm* is one way to measure the "size" of a matrix

$$\left\| \begin{bmatrix} a & b \\ c & d \end{bmatrix} \right\|_F = \sqrt{a^2 + b^2 + c^2 + d^2}$$

and in general, $\|A\|_F = \left(\sum_{i,j} a_{ij}^2\right)^{1/2}$, that is, the square root of the sum of squares of the entries of $A$.
If $\widehat{A}$ is an approximation to $A$, then we can quantify the goodness of fit by $\|A - \widehat{A}\|_F$. This is just a least-squares measure.
The SVD has the property that, if you want to approximate a matrix $A$ by a matrix $\widehat{A}$ of lower rank, then the matrix that minimizes $\|A - \widehat{A}\|_F$ among all rank-1 matrices is the matrix

$$\widehat{A} = \begin{bmatrix} \vdots \\ \mathbf{u}_1 \\ \vdots \end{bmatrix}_{m\times 1} \begin{bmatrix} \sigma_1 \end{bmatrix}_{1\times 1} \begin{bmatrix} \vdots \\ \mathbf{v}_1 \\ \vdots \end{bmatrix}^T_{1\times n} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$$

In general, the best rank-$r$ approximation to $A$ is given by

$$\begin{aligned} \widehat{A} &= \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_r \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}_{m\times r} \begin{bmatrix} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \sigma_r \end{bmatrix}_{r\times r} \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_r \\ \vdots & \vdots & \dots & \vdots \end{bmatrix}^T_{r\times n} \\ &= \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T \end{aligned}$$

 

and it can be shown that

$$\|A - \widehat{A}\|_F = \sqrt{\sigma_{r+1}^2 + \sigma_{r+2}^2 + \ldots + \sigma_n^2} \tag{L8.2}$$

The MATLAB command

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

returns the SVD decomposition of the matrix $A$, that is, it returns matrices $U$, $S$ and $V$ such that $A = USV^T$.

## Example

The SVD of the following matrix $A$ is:

$$A = \begin{bmatrix} -2 & 8 & 20 \\ 14 & 19 & 10 \\ 2 & -2 & 1 \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & -\frac{4}{5} & 0 \\ \frac{4}{5} & \frac{3}{5} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 30 & 0 & 0 \\ 0 & 15 & 0 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} & -\frac{2}{3} \\ \frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

(a) Enter the matrix $A$ and compute $U$, $S$ and $V$ using the `svd` comand.
Verify that $A = USV^T$.

**Answer:**

```
 >> A=[-2,8,20;14,19,10;2, -2, 1];
 >> [U,S,V]=svd(A)
U =
    0.6000   -0.8000    0.0000
    0.8000    0.6000    0.0000
    0.0000   -0.0000    1.0000
S =
   30.0000         0         0
         0   15.0000         0
         0         0    3.0000
V =
    0.3333    0.6667    0.6667
    0.6667    0.3333   -0.6667
    0.6667   -0.6667    0.3333
```

We can easily verify that `U*S*V'` returns the matrix $A$.

(b) Use MATLAB to find the best rank-1 approximation to $A$ (with respect to the Frobenius norm), that is, $A1 = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T$.
Use the command `rank` to verify that the matrix `A1` has indeed rank one.
Evaluate $\|A - A1\|_F$ by typing `norm(A - A1,'fro')` and verify Eq. (L8.2).

**Answer:**

```
>> A1 = S(1,1)*U(:,1)*V(:,1)'
A1 =
    6.0000   12.0000   12.0000
    8.0000   16.0000   16.0000
    0.0000    0.0000    0.0000
>> rank(A1)
ans =
     1
```

```
>> norm(A-A1,'fro')
ans =
   15.2971
>> sqrt(S(2,2)^2+S(3,3)^2)
ans =
   15.2971
```

Note that, although `A1` is $3 \times 3$ and therefore it contains nine entries, we only need seven values to generate it: (one singular value)+ (one column of $U$) +( one column of $V$) $= 1 + 3 + 3 = 7$.
This is less than the number of entries in the matrix $A$.

(c) Find the best rank-2 approximation to $A$. Check the rank and the Frobenius norm of $A - A2$ using MATLAB and verify (L8.2).

**Answer:**

```
>> A2 = A1+S(2,2)*U(:,2)*V(:,2)'
A2 =
   -2.0000    8.0000   20.0000
   14.0000   19.0000   10.0000
   -0.0000    0.0000    0.0000
>> rank(A2)
ans =
     2
>> norm(A-A2,'fro')
ans =
    3.0000
>> S(3,3)
ans =
     3
```

Note that we need 14 entries to generate `A2`:
(2 singular values) + (2 columns of $U$) + (2 columns of $V$) $= 2 + 2 \cdot 3 + 2 \cdot 3 = 14$.
This is more than the number of entries in the original matrix $A$.

## Image Compression Exercises

**Instructions:** The following problems can be done interactively or by writing the commands in an M-file (or by a combination of the two). In either case, record all MATLAB input commands and output in a text document and edit it according to the instructions of LAB 1 and LAB 2. For problem 2, include a picture of the rank-1 approximation. For problem 3, include a picture of the rank-10 approximation and for problem 4, include a picture of the lower rank approximation that, in your opinion, gives an acceptable approximation to the original picture. Crop and resize the pictures so that they do not take up too much space and paste them into your lab report in the appropriate order.

**Step 1.** Download the file `cauchybw.jpg` and save it to your working MATLAB directory. Then load it into MATLAB with the command

```
A = imread('cauchybw.jpg');        note the semicolon
```

The semicolon is necessary so that MATLAB does not print out many screenfuls of data. The result is a matrix of grayscale values corresponding to a black and white picture of a dog. (The matrix has $104,780$ entries). Now, $A$ is actually $310 \times 338 \times 3$. To create an image from the matrix $A$, MATLAB uses the three values $A(i, j, 1:3)$ as the RGB values to use to color the $ij$th pixel. We have a black and white picture so `A(:,:,1) = A(:,:,2) = A(:,:,3)` and we only need to work with `A(:,:,1)`.

**Step 2**. We need to do some initial processing. Type

```
B = double(A(:,:,1)) + 1;        don't forget the semicolon
```

which converts $A$ into the double-precision format that is needed for the singular value decomposition. Now type

```
B = B/256;       semicolon!
[U S V] = svd(B);       semicolon!
```

This decomposition is just Eq. (L8.1).
The gray scale goes from 0 to 256 in a black- and-white JPEG image. We divide $B$ by 256 to obtain values between 0 and 1, which is required for MATLAB's image routines, which we will use later.

**PROBLEM 1.** What are the dimensions of $U$, $S$, and $V$? (Find out by typing `size(U)` - without the semicolon - and likewise for the others.)
Here $S$ has more columns than rows; in fact, columns 311 to 338 are all zeros (When $A$ has more columns than rows, we pad $S$ on the right with zero columns to turn $S$ into an $m \times n$ matrix ). Otherwise, with this modification, the SVD is just like Eq. (L8.1).

**PROBLEM 2.** Compute the best rank-1 approximation to $B$ and store it in the matrix `rank1` (Use the commands given in the Example).

**Step 3**. Let's visualize `rank1`. To do that, first create

```
C = zeros(size(A));        semicolon!
```

This creates an array of zeros, $C$, of the same dimension as the original image matrix $A$.

**Step 4**. Copy the rank-1 image into $C$ as follows:

```
C(:,:,1)  = rank1;
C(:,:,2)  = rank1;
C(:,:,3)  = rank1;
```

**Step 5**: We are almost done, except for one hitch. MATLAB does all its scaling using values from 0 to 1 (and maps them to values between 0 and 256 for the graphics hardware). Lower-rank approximations to the actual image can have values that are slightly less than 0 and greater than 1. So we will truncate them to fit, as follows:

```
C = max(0,min(1,C));
```

**Step 6**. View the resulting image:

```
image(C)       no semicolon
```

**PROBLEM 3**. Create and view a rank-10 approximation to the original picture (Use Steps 4-6 but with `rank10` instead of `rank1`. If you mess up - for example, you get an all-black picture - then start over from Step 3.) It is convenient to create an M-file with a `for` loop to evaluate $\sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \ldots + \sigma_{10} \mathbf{u}_{10} \mathbf{v}_{10}^T$

**PROBLEM 4.** Repeat with rank-20, 30 and 40 approximations (and any other ranks that you'd like to experiment with). What is the smallest rank that, in your opinion, gives an acceptable approximation to the original picture?
In your lab write-up, include the code that gives an acceptable approximation and the corresponding picture.

**PROBLEM 5.** What rank-$r$ approximation **exactly** reproduces the original picture?

 

**PROBLEM 6.** Suppose that a rank-$k$ approximation, for some $k$, gives an acceptable approximation. How much data is needed to represent the rank-$k$ approximation? (Hint: you need $k$ columns of $U$, $k$ columns of $V$, and $k$ singular values of $S$.) The ratio of the amount of data used for the approximation and the amount of data of the (original format of the) picture is the *compression rate*. Find the compression rate for the value of the rank you determined in Problem 4. What does the compression rate represent?

**PROBLEM 7.** If we use a high rank approximation, then the amount of data needed for the approximation may exceed the amount of data used in the original representation of the picture. Find the smallest value of $k$ such that the rank-$k$ approximation of the matrix uses the same or more amount of data as the original picture. Approximations with ranks higher than this $k$ cannot be used for image compression since they defeat the purpose of using less data than in the original representation.

 