Chapter 7

The Singular Value Decomposition (SVD)

7.1 Image Processing by Linear Algebra

1 An image is a large matrix of grayscale values, one for each pixel and color.

2 When nearby pixels are correlated (not random) the image can be compressed.

3 The SVD separates any matrix A into rank one pieces $\boldsymbol{uv}^{\mathrm{T}}$ = (**column**)(**row**).

4 The columns and rows are eigenvectors of symmetric matrices AA^{T} and $A^{\mathrm{T}}A$.

The singular value theorem for $\boldsymbol{A}$ is the eigenvalue theorem for $\boldsymbol{A^{\mathrm{T}}A}$ and $\boldsymbol{AA^{\mathrm{T}}}$.

That is a quick preview of what you will see in this chapter. A has *two* sets of singular vectors (the eigenvectors of $A^{\mathrm{T}}A$ and AA^{T}). There is *one* set of positive singular values (because $A^{\mathrm{T}}A$ has the same positive eigenvalues as AA^{T}). A is often rectangular, but $A^{\mathrm{T}}A$ and AA^{T} are square, symmetric, and positive semidefinite.

The Singular Value Decomposition (SVD) separates any matrix into simple pieces.

Each piece is a column vector times a row vector. An m by n matrix has m times n entries (a big number when the matrix represents an image). But a column and a row only have $\boldsymbol{m+n}$ **components**, **far less than $\boldsymbol{m}$ times $\boldsymbol{n}$**. Those (column)(row) pieces are full size matrices that can be processed with extreme speed—they need only m *plus* n numbers.

Unusually, this image processing application of the SVD is coming before the matrix algebra it depends on. I will start with simple images that only involve one or two pieces. Right now I am thinking of an image as a large rectangular matrix. The entries a_{ij} tell the grayscales of all the pixels in the image. Think of a pixel as a small square, i steps across and j steps up from the lower left corner. Its grayscale is a number (often a whole number in the range $0 \le a_{ij} < 256 = 2^8$). An all-white pixel has $a_{ij} = 255 = 11111111$. That number has eight 1's when the computer writes 255 in binary notation.

You see how an image that has m times n pixels, with each pixel using 8 bits (0 or 1) for its grayscale, becomes an m by n matrix with 256 possible values for each entry a_{ij}.

In short, an image is a large matrix. To copy it perfectly, we need $8\,(m)(n)$ bits of information. High definition television typically has $m = 1080$ and $n = 1920$. Often there are 24 frames each second and you probably like to watch in color (3 color scales). This requires transmitting $(3)(8)(48,470,400)$ bits per second. That is too expensive and it is not done. The transmitter can't keep up with the show.

When compression is well done, you can't see the difference from the original. *Edges in the image* (sudden changes in the grayscale) are the hard parts to compress.

Major success in compression will be impossible if every a_{ij} is an independent random number. We totally depend on the fact that *nearby pixels generally have similar grayscales*. An edge produces a sudden jump when you cross over it. Cartoons are more compressible than real-world images, with edges everywhere.

For a video, the numbers a_{ij} don't change much between frames. **We only transmit the small changes**. This is *difference coding* in the H.264 video compression standard (on this book's website). We compress each change matrix by linear algebra (and by nonlinear "quantization" for an efficient step to integers in the computer).

The natural images that we see every day are absolutely ready and open for compression—but that doesn't make it easy to do.

Low Rank Images (Examples)

The easiest images to compress are all black or all white or all a constant grayscale g. The matrix A has the same number g in every entry : $a_{ij} = g$. When $g = 1$ and $m = n = 6$, here is an extreme example of the central SVD dogma of image processing :

Example 1 **Don't send** $A = \begin{bmatrix} 1&1&1&1&1&1 \\ 1&1&1&1&1&1 \\ 1&1&1&1&1&1 \\ 1&1&1&1&1&1 \\ 1&1&1&1&1&1 \\ 1&1&1&1&1&1 \end{bmatrix}$ **Send this** $A = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1&1&1&1&1&1 \end{bmatrix}$

36 numbers become 12 numbers. With 300 by 300 pixels, $90,000$ numbers become 600. And if we define the all-ones vector $\boldsymbol{x}$ in advance, we only have to send **one number**. That number would be the constant grayscale g that multiplies $\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$ to produce the matrix.

Of course this first example is extreme. But it makes an important point. If there are special vectors like $\boldsymbol{x}$ = **ones** that can usefully be defined in advance, then image processing can be extremely fast. The battle is between **preselected bases** (the Fourier basis allows speed-up from the FFT) and **adaptive bases** determined by the image. The SVD produces bases from the image itself—this is adaptive and it can be expensive.

I am not saying that the SVD always or usually gives the most effective algorithm in practice. The purpose of these next examples is instruction and not production.

Example 2
"ace flag"
French flag $\boldsymbol{A}$
Italian flag $\boldsymbol{A}$
German flag $\boldsymbol{A}^{\mathbf{T}}$

Don't send $A = \begin{bmatrix} a & a & c & c & e & e \\ a & a & c & c & e & e \\ a & a & c & c & e & e \\ a & a & c & c & e & e \\ a & a & c & c & e & e \\ a & a & c & c & e & e \end{bmatrix}$ **Send** $A = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} a & a & c & c & e & e \end{bmatrix}$

This flag has 3 colors but it still has rank 1. We still have one column times one row. The 36 entries could even be all different, provided they keep that rank 1 pattern $A = \boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}$. But when the rank moves up to $r = 2$, we need $\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + \boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}$. Here is one choice:

Example 3
Embedded square

$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$ is equal to $A = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix}$

The 1's and the 0 in A could be blocks of 1's and a block of 0's. *We would still have rank* 2. We would still only need two terms $\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}$ and $\boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}$. A 6 by 6 image would be compressed into 24 numbers. An N by N image (N^2 numbers) would be compressed into $4N$ numbers from the four vectors $\boldsymbol{u}_1, \boldsymbol{v}_1, \boldsymbol{u}_2, \boldsymbol{v}_2$.

Have I made the best choice for the $\boldsymbol{u}$'s and $\boldsymbol{v}$'s ? This is *not* the choice from the SVD! I notice that $\boldsymbol{u}_1 = (1, 1)$ is not orthogonal to $\boldsymbol{u}_2 = (1, 0)$. And $\boldsymbol{v}_1 = (1, 1)$ is not orthogonal to $\boldsymbol{v}_2 = (0, 1)$. The theory says that orthogonality will produce a smaller second piece $c_2\boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}$. **(The SVD chooses rank one pieces in order of importance.)**

If the rank of A is much higher than 2, as we expect for real images, then A will add up many rank one pieces. We want the small ones to be really small—they can be discarded with no loss to visual quality. Image compression becomes lossy, but good image compression is virtually undetectable by the human visual system.

The question becomes: **What are the orthogonal choices from the SVD**?

Eigenvectors for the SVD

I want to introduce the use of eigenvectors. But the eigenvectors of most images are not orthogonal. Furthermore the eigenvectors $\boldsymbol{x}_1, \boldsymbol{x}_2$ give only one set of vectors, and we want two sets ($\boldsymbol{u}$'s and $\boldsymbol{v}$'s). The answer to both of those difficulties is the SVD idea:

Use the eigenvectors u of $\boldsymbol{AA}^{\mathbf{T}}$ and the eigenvectors v of $\boldsymbol{A}^{\mathbf{T}}\boldsymbol{A}$.

Since AA^{T} and $A^{\mathrm{T}}A$ are automatically symmetric (but not usually equal!) the $\boldsymbol{u}$'s will be one orthogonal set and the eigenvectors $\boldsymbol{v}$ will be another orthogonal set. We can and will make them all unit vectors: $||\boldsymbol{u}_i|| = 1$ and $||\boldsymbol{v}_i|| = 1$. Then our rank 2 matrix will be $A = \sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}$. The size of those numbers σ_1 and σ_2 will decide whether they can be ignored in compression. *We keep larger σ's, we discard small σ's.*

The $\boldsymbol{u}$'s from the SVD are called **left singular vectors** (unit eigenvectors of AA^{T}). The $\boldsymbol{v}$'s are **right singular vectors** (unit eigenvectors of $A^{\mathrm{T}}A$). The σ's are **singular values**, square roots of the equal eigenvalues of AA^{T} and $A^{\mathrm{T}}A$:

Choices from the SVD $AA^{\mathrm{T}}\boldsymbol{u}_i = \sigma_i^2\boldsymbol{u}_i$ $A^{\mathrm{T}}A\boldsymbol{v}_i = \sigma_i^2\boldsymbol{v}_i$ $A\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i$ (1)

In Example 3 (the embedded square), here are the symmetric matrices AA^{T} and $A^{\mathrm{T}}A$:

$$AA^{\mathrm{T}} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{2} \end{bmatrix} \qquad A^{\mathrm{T}}A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{2} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} \end{bmatrix}.$$

Their determinants are 1, so $\lambda_1\lambda_2 = 1$. Their traces (diagonal sums) are 3:

$$\det\begin{bmatrix} 1-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} = \lambda^2 - 3\lambda + 1 = 0 \quad \text{gives} \quad \lambda_1 = \frac{3+\sqrt{5}}{2} \quad \text{and} \quad \lambda_2 = \frac{3-\sqrt{5}}{2}.$$

The square roots of λ_1 and λ_2 are $\sigma_1 = \dfrac{\sqrt{5}+1}{2}$ and $\sigma_2 = \dfrac{\sqrt{5}-1}{2}$ with $\sigma_1\,\sigma_2 = 1$.

The nearest rank 1 matrix to A will be $\sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}}$. The error is only $\sigma_2 \approx 0.6 =$ best possible.

The orthonormal eigenvectors of AA^{T} and $A^{\mathrm{T}}A$ are

$$\boldsymbol{u}_1 = \begin{bmatrix} 1 \\ \sigma_1 \end{bmatrix} \quad \boldsymbol{u}_2 = \begin{bmatrix} \sigma_1 \\ -1 \end{bmatrix} \quad \boldsymbol{v}_1 = \begin{bmatrix} \sigma_1 \\ 1 \end{bmatrix} \quad \boldsymbol{v}_2 = \begin{bmatrix} 1 \\ -\sigma_1 \end{bmatrix} \quad \text{all divided by } \sqrt{1+\sigma_1^2}. \quad (2)$$

Every reader understands that in real life those calculations are done by computers! (Certainly not by unreliable professors. I corrected myself using $\mathsf{svd}\,(A)$ in MATLAB.) And we can check that the matrix A is correctly recovered from $\sigma_1\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + \sigma_2\boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}$:

$$A = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 \end{bmatrix}\begin{bmatrix} \sigma_1 & \\ & \sigma_2 \end{bmatrix}\begin{bmatrix} \boldsymbol{v}_1^{\mathrm{T}} \\ \boldsymbol{v}_2^{\mathrm{T}} \end{bmatrix} \text{ or more simply } A\begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 \end{bmatrix} = \begin{bmatrix} \sigma_1\,\boldsymbol{u}_1 & \sigma_2\,\boldsymbol{u}_2 \end{bmatrix} \quad (3)$$

Important The key point is not that images tend to have low rank. **No**: Images mostly have full rank. But they do have **low effective rank**. This means: Many singular values are small and can be set to zero. *We transmit a low rank approximation.*

Example 4 Suppose the flag has two triangles of different colors. The lower left triangle has 1's and the upper right triangle has 0's. The main diagonal is included with the 1's. Here is the image matrix when $n = 4$. It has full rank $r = 4$ so it is invertible:

Triangular flag matrix $$A = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \quad \text{and} \quad A^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

With full rank, A has a full set of n singular values σ (all positive). The SVD will produce n pieces $\sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ of rank one. Perfect reproduction needs all n pieces.

In compression *small* σ's can be discarded with no serious loss in image quality. We want to understand and plot the σ's for $n = 4$ and also for large n. Notice that Example 3 was the special case $n = 2$ of this triangular Example 4.

Working by hand, we begin with AA^{T} (a computer would proceed differently):

$$AA^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{bmatrix} \quad \text{and} \quad (AA^{\mathrm{T}})^{-1} = (A^{-1})^{\mathrm{T}} A^{-1} = \begin{bmatrix} \mathbf{2} & -1 & 0 & 0 \\ -1 & \mathbf{2} & -1 & 0 \\ 0 & -1 & \mathbf{2} & -1 \\ 0 & 0 & -1 & \mathbf{1} \end{bmatrix}. \quad (4)$$

That $-1, 2, -1$ inverse matrix is included because its eigenvalues all have the form $2 - 2\cos\theta$. So we know the λ's for AA^{T} and the σ's for A:

$$\lambda = \frac{1}{2 - 2\cos\theta} = \frac{1}{4\sin^2(\theta/2)} \quad \text{gives} \quad \sigma = \sqrt{\lambda} = \frac{1}{2\sin(\theta/2)}. \quad (5)$$

The n different angles θ are equally spaced, which makes this example so exceptional:

$$\theta = \frac{\pi}{2n+1}, \frac{3\pi}{2n+1}, \ldots, \frac{(2n-1)\,\pi}{2n+1} \qquad \left(n = 4 \text{ includes } \theta = \frac{3\pi}{9} \text{ with } 2\sin\frac{\theta}{2} = 1 \right).$$

That special case gives $\lambda = 1$ as an eigenvalue of AA^{T} when $n = 4$. So $\sigma = \sqrt{\lambda} = 1$ is a singular value of A. You can check that the vector $\boldsymbol{u} = (1, 1, 0, -1)$ has $AA^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{u}$ (a truly special case).

The important point is to graph the n singular values of A. Those numbers drop off (unlike the eigenvalues of A, which are all 1). But the dropoff is not steep. So the SVD gives only moderate compression of this triangular flag. *Great compression for Hilbert.*

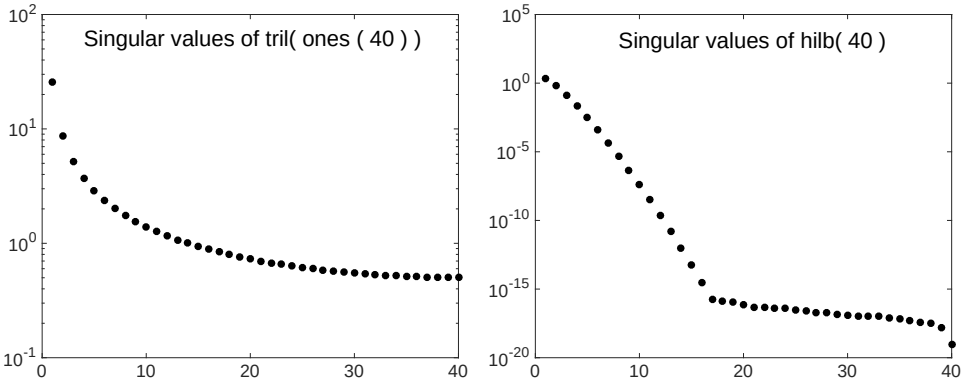


Figure 7.1: Singular values of the triangle of 1's in Examples 3-4 (not compressible) and the evil Hilbert matrix $H(i, j) = (i + j - 1)^{-1}$ in Section 8.3: compress it to work with it.

Your faithful author has continued research on the ranks of flags. Quite a few are based on horizontal or vertical stripes. Those have *rank one*—all rows or all columns are multiples of the *ones* vector $(1, 1, \ldots, 1)$. Armenia, Austria, Belgium, Bulgaria, Chad, Colombia, Ireland, Madagascar, Mali, Netherlands, Nigeria, Romania, Russia (and more) have three stripes. Indonesia and Poland have two ! Libya was the extreme case in the Gadaffi years 1977 to 2011 (*the whole flag was green*).

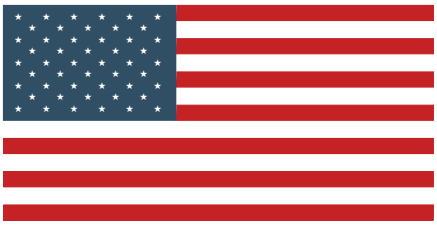
At the other extreme, many flags include diagonal lines. Those could be long diagonals as in the British flag. Or they could be short diagonals coming from the edges of a star—as in the US flag. The text example of a triangle of ones shows how those flag matrices will have large rank. The rank increases to infinity as the pixel sizes get small.

Other flags have circles or crescents or various curved shapes. Their ranks are large and also increasing to infinity. These are still compressible! The compressed image won't be perfect but our eyes won't see the difference (with enough terms $\sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^{\mathrm{T}}$ from the SVD). Those examples actually bring out the main purpose of image compression:

Visual quality can be preserved even with a big reduction in the rank.

For fun I looked back at the flags with finite rank. They can have stripes and they can also have crosses—provided the edges of the cross are horizontal or vertical. Some flags have a thin outline around the cross. This artistic touch will increase the rank. Right now my champion is the flag of Greece shown below, with a cross and also stripes. Its rank is **four** by my counting. I see no US State Flags of finite rank !

The reader could google "national flags" to see the variety of designs and colors. I would be glad to know any finite rank examples with rank > 4. Good examples of all kinds will go on the book's website **math.mit.edu/linearalgebra** (and flags in full color).



Problem Set 7.1

1 What are the ranks r for these matrices with entries i times j and i plus j? Write A and B as the sum of r pieces $\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ of rank one. Not requiring $\boldsymbol{u}_1^{\mathrm{T}}\boldsymbol{u}_2 = \boldsymbol{v}_1^{\mathrm{T}}\boldsymbol{v}_2 = \mathbf{0}$.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 9 & 12 \\ 4 & 8 & 12 & 16 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

2 We usually think that the identity matrix I is as simple as possible. But why is I completely incompressible? *Draw a rank* 5 *flag with a cross.*

3 These flags have rank 2. Write A and B in any way as $\boldsymbol{u}_1\boldsymbol{v}_1^{\mathrm{T}} + \boldsymbol{u}_2\boldsymbol{v}_2^{\mathrm{T}}$.

$$\boldsymbol{A}_{\mathbf{Sweden}} = \boldsymbol{A}_{\mathbf{Finland}} = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 \end{bmatrix} \qquad \boldsymbol{B}_{\mathbf{Benin}} = \begin{bmatrix} 1 & 2 & 2 \\ 1 & 3 & 3 \end{bmatrix}$$

4 Now find the trace and determinant of BB^{T} and $B^{\mathrm{T}}B$ in Problem 3. The singular values of B are close to $\sigma_1^2 = 28 - \frac{1}{14}$ and $\sigma_2^2 = \frac{1}{14}$. Is B compressible or not?

5 Use $[U, S, V] = \mathsf{svd}\,(A)$ to find two orthogonal pieces $\sigma\boldsymbol{u}\boldsymbol{v}^{\mathrm{T}}$ of $\boldsymbol{A}_{\mathbf{Sweden}}$.

6 Find the eigenvalues and the singular values of this 2 by 2 matrix A.

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 2 \end{bmatrix} \quad \text{with} \quad A^{\mathrm{T}}A = \begin{bmatrix} 20 & 10 \\ 10 & 5 \end{bmatrix} \quad \text{and} \quad AA^{\mathrm{T}} = \begin{bmatrix} 5 & 10 \\ 10 & 20 \end{bmatrix}.$$

The eigenvectors $(1, 2)$ and $(1, -2)$ of A are not orthogonal. How do you know the eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2$ of $A^{\mathrm{T}}A$ are orthogonal? Notice that $A^{\mathrm{T}}A$ and AA^{T} have the same eigenvalues (25 and 0).

7 How does the second form $\boldsymbol{AV = U\Sigma}$ in equation (3) follow from the first form $\boldsymbol{A = U\Sigma V^{\mathrm{T}}}$? That is the most famous form of the SVD.

8 The two columns of $AV = U\Sigma$ are $A\boldsymbol{v}_1 = \sigma_1\boldsymbol{u}_1$ and $A\boldsymbol{v}_2 = \sigma_2\boldsymbol{u}_2$. So we hope that

$$A\boldsymbol{v}_1 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \sigma_1 \\ 1 \end{bmatrix} = \sigma_1\begin{bmatrix} 1 \\ \sigma_1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -\sigma_1 \end{bmatrix} = \sigma_2\begin{bmatrix} \sigma_1 \\ -1 \end{bmatrix}$$

The first needs $\sigma_1 + 1 = \sigma_1^2$ and the second needs $1 - \sigma_1 = -\sigma_2$. Are those true?

9 The MATLAB commands $A = \mathsf{rand}\,(20, 40)$ and $B = \mathsf{randn}\,(20, 40)$ produce 20 by 40 random matrices. The entries of A are between 0 and 1 with uniform probability. The entries of B have a normal "bell-shaped" probability distribution. Using an svd command, find and graph their singular values σ_1 to σ_{20}. Why do they have 20 σ's?