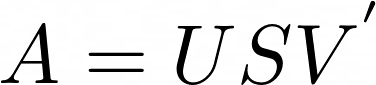
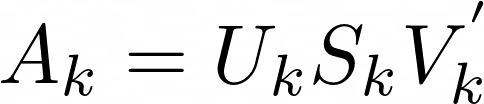
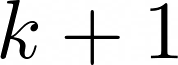
# SVD and Image Compression Overview

Consider the m by n matrix The matrix A with entries represented by “one-byte” integers. One byte equals 8 bits of information, so each value in the matrix is represented by an integer in the range from 0 to 255. Since each matrix entry requires one byte of storage, the total matrix requires m times n bytes of storage space.

The singular value decomposition (SVD) can be used to compute an orthogonal decomposition of a matrix. Given the m by n matrix The matrix A, the SVD can be used to decompose the matrix as , where The matrix U is an orthogonal m by m matrix, The matrix V is an orthogonal n by n matrix, and The matrix S is a m by n rectangular diagonal matrix whose diagonal contains the singular values of The matrix A. This representation is very similar to the eigenvalue diagonalization (called eigendecomposition) already studied in the course. However, a primary difference between SVD and eigendecomposition is that SVD can be used for rectangular matrices, while eigendecomposition can be used only on square matrices.

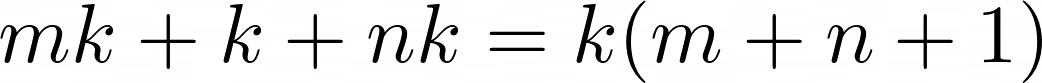
SVD can be used to construct a *low-rank approximation* of the original matrix. If only the first The discrete value k singular values are used, then The matrix A can be approximated by the matrix  defined as:

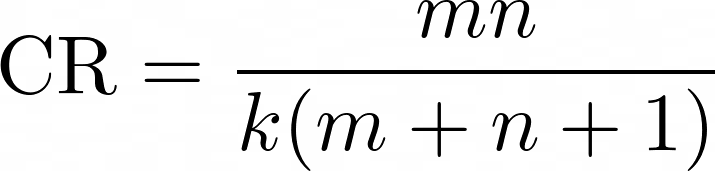


where  is a m by k matrix containing the first The discrete value k columns of The matrix U,  is a k by k matrix containing the first The discrete value k rows and The discrete value k columns of The matrix S, and  is a n by k matrix containing the first The discrete value k columns of The matrix V. Observe that this approximation essentially throws away all of the information stored in the dimensions  and higher.

After computing the image approximation, we must be careful to make sure that the approximation has the correct data format. Each value in the matrix must be an unsigned 8-bit integer data type. Since the approximation now likely has real-valued quantities in it, we will round these entries to the closest integer and then *cast* the data type to an integer. This allows the image to be properly displayed in software programs such as MATLAB. So, after computing , we’ll typically execute a command similar to the following:

Ak = uint8(round(Ak))

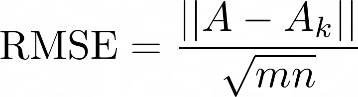
The m by k matrix  has a total of m times k entries that must be stored. Since  is a diagonal matrix, only its The discrete value k diagonal elements need to be stored. The n by k matrix  has n times k entries that must be stored. Thus, the total number of values to store for the low-rank matrix  is . Compared to the original storage requirement of m times n, using only The discrete value k singular values to approximate The matrix A with  achieves a compression ratio of



To provide a specific example, if m equals six hundred, n equals four hundred eighty, and k equals fifty, this would result in a compression ratio of

The compression ratio is equal to a numerator term divided by a denominator term. The numerator term is six hundred times four hundred eighty. The denominator term is 50 times the quantity six hundred plus four hundred eighty plus one.
This expression is approximately equal to 5.32.

While the low-rank approximation can require significantly less storage space than the original matrix, it is an approximation that in general is not equal to the original matrix. To quantify the error between The matrix A and , we can compute the *root mean square error* as:



The matrix norm above is the *Frobenius norm*, which is defined as the square root of the sum of absolute squares of its elements. In general, the RMSE will increase as the value for The discrete value k decreases, i.e., the less storage used, the larger the approximation and the larger the error. This tradeoff between compression and error will be investigated graphically in the project.

## **Additional Useful MATLAB Commands**

* MATLAB can compute the singular value decomposition (SVD) of a matrix using the svd() function. The outputs of the svd() function are the three matrices comprising the SVD decomposition. The MATLAB command is: [U,S,V] = svd(A);
* To display an image in MATLAB, you first need to load the file into the workspace. There are several ways to do this, but one option is the [load command.](https://www.mathworks.com/help/matlab/ref/load.html)
* To display an image in MATLAB, an empty figure must first be initialized. The MATLAB command is: figure;

This command initializes an empty figure. The next command that plots or displays information will appear in the figure.

* The MATLAB command imshow() can be used to display an image in a figure. If the image is stored in matrix The matrix A, the image can be displayed with the command: imshow(A);
* The MATLAB command title() can be used to title a figure. The argument of the title() function is a string with the desired title, e.g., title(‘My Image Title’)
* Files can be easily uploaded to or downloaded from the MATLAB Online interface. Under the **Home** tab, there are **Upload** and **Download** buttons. Clicking the **Upload** button will prompt the user to select the file to upload. If a single file is selected (e.g., project01Solution.pdf), the **Download** button can be clicked to download the file to the user’s local computer.

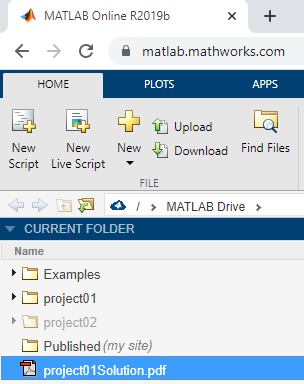


Figure: A folder in MATLAB with the project01Solution.pdf file selected