SVD Example: Image compression

Author: Seung-Sup Lee

We will provide a visual understanding of how the SVD can be used to compress large matrices. First load a sample image data of *Gwanghwamun* (광화문), retrived from a Wikimedia page.

```
clear

M = imread('Gwanghwamun.jpg'); % Read image data from a picture
```

To see the information of variables, type:

```
whos M
```

```
  Name         Size                 Bytes  Class    Attributes

  M         2730x4855x3            39762450  uint8
```

If you use graphic interface, just look the workspace panel which is usually on the upper-right corner of the MATLAB main window. We see that `M` is $2730 \times 4855 \times 3$ matrix of `uint8` type. The first and second dimensions of `M` indicates the height and width of the photo, and the third dimension encodes RGB data. As `uint8` is unsigned integer each occupying 8 bit (= 1 byte) of memory. Therefore, `M` is about 10 MB!

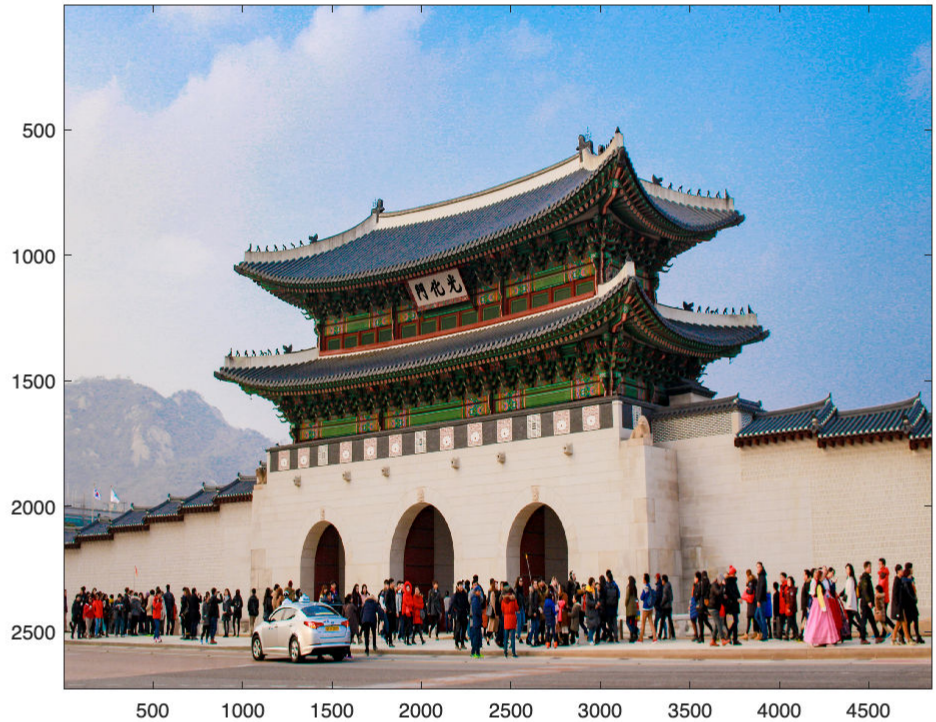
Visualization of matrix

MATLAB provides several functions to visualize matrices.

```
figure; % open new figure window
imshow(M); % display image with matrix of uint8 type (NOT double).
```

`imshow` does not work for double type variables (which are generally used in MATLAB calculations). So for general purpooses, use `imagesc` as below.

```
figure;
imagesc(M); % display image with axes.
```



Note that the height-to-width ratio of pictures by `imshow` is the same as the original picture, but the ratio of pictures by `imagesc` is fitted to the figure window size.

For the rest of this tutorial, we will make `M` as a matrix, by converting it to double and sum over the third dimension.

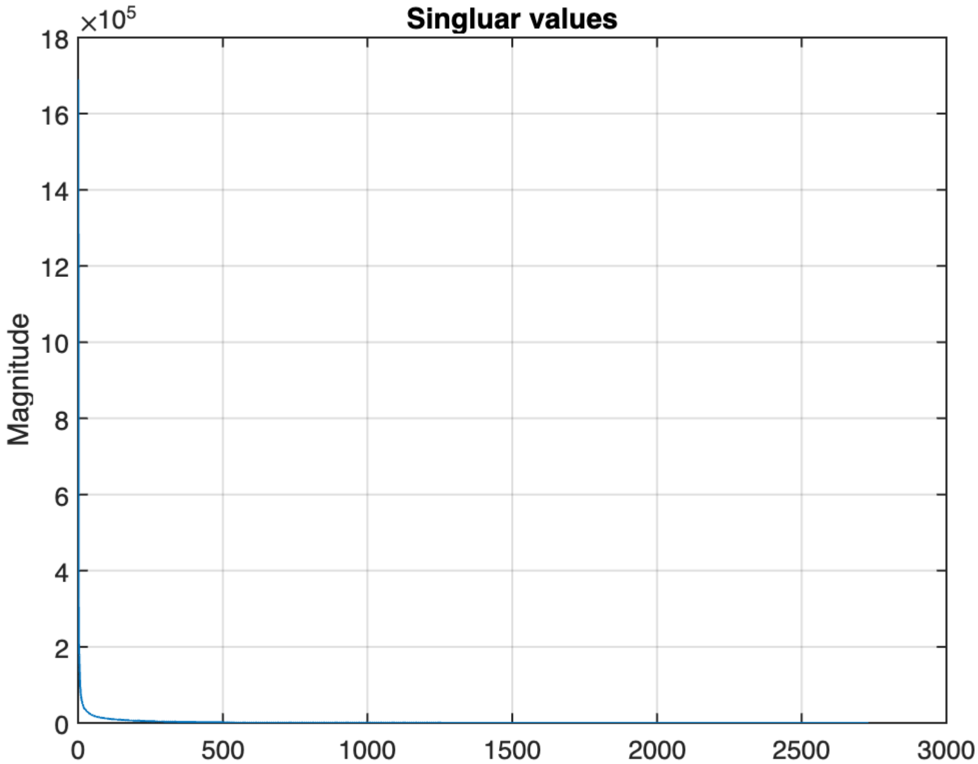
```
M = double(M); % convert data type: uint8 -> double
M = sum(M,3); % sum over the 3rd dim.
```

SVD of picture data

```
[U,S,V] = svd(M); % singular value decomposition
```

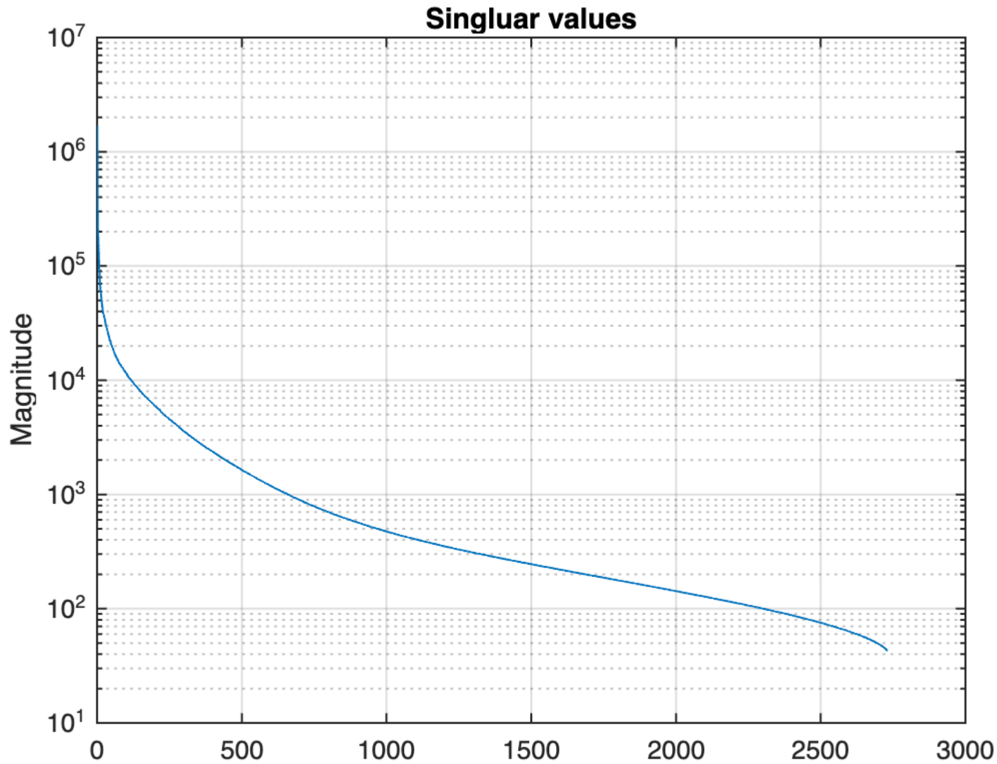
To see the distribution of the singular values, we plot them.

```
figure;
plot(diag(S),'LineWidth',1); % plot diagonal elements of S.
set(gca,'LineWidth',1,'FontSize',13)
title('Singluar values'); % add title
ylabel('Magnitude'); % add y-axis label
grid on; % turn on grid line
```



The magnitude of the singluar values decays exponentially. To better see the exponential decay, plot in log-linear scale.

```
figure;
semilogy(diag(S),'LineWidth',1); % plot diagonal elements of S.
set(gca,'LineWidth',1,'FontSize',13)
title('Singluar values'); % add title
ylabel('Magnitude'); % add y-axis label
grid on; % turn on grid line
```



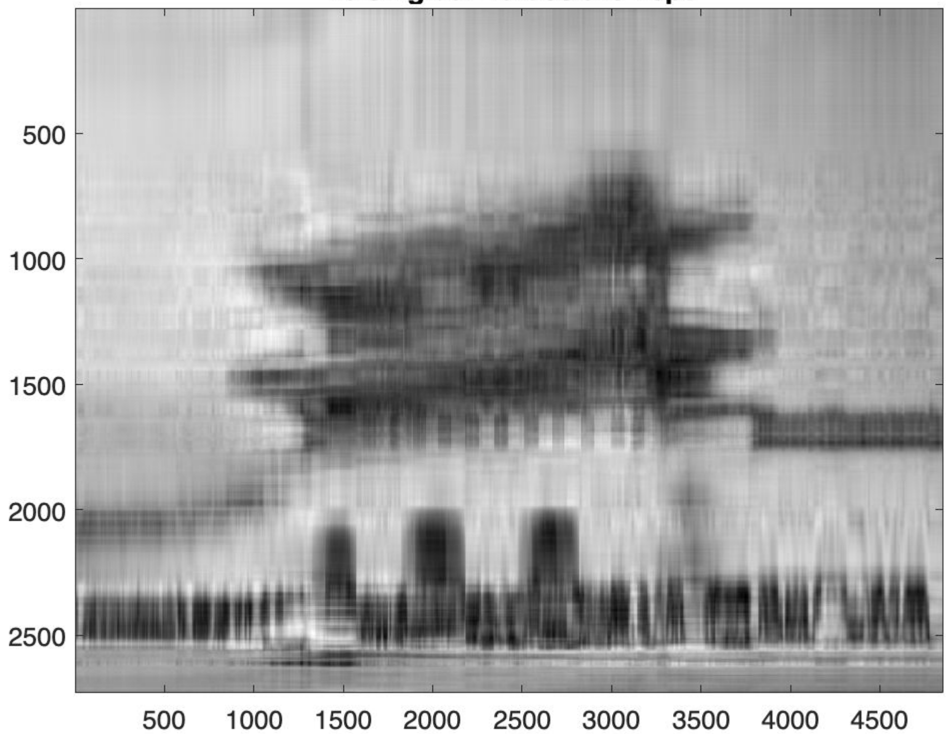
Reconstruction of picture

Now we reconstruct picture from the SVD result of M. It is clear that U*S*V' will return the same matrix as M (up to double precision ~1e-16). But what if we use only the parts of U, S, and V? Based on the exponential decay of the singular values, we can think of an approach that keeps only some of the largest singular values and the corresponding singular vectors.

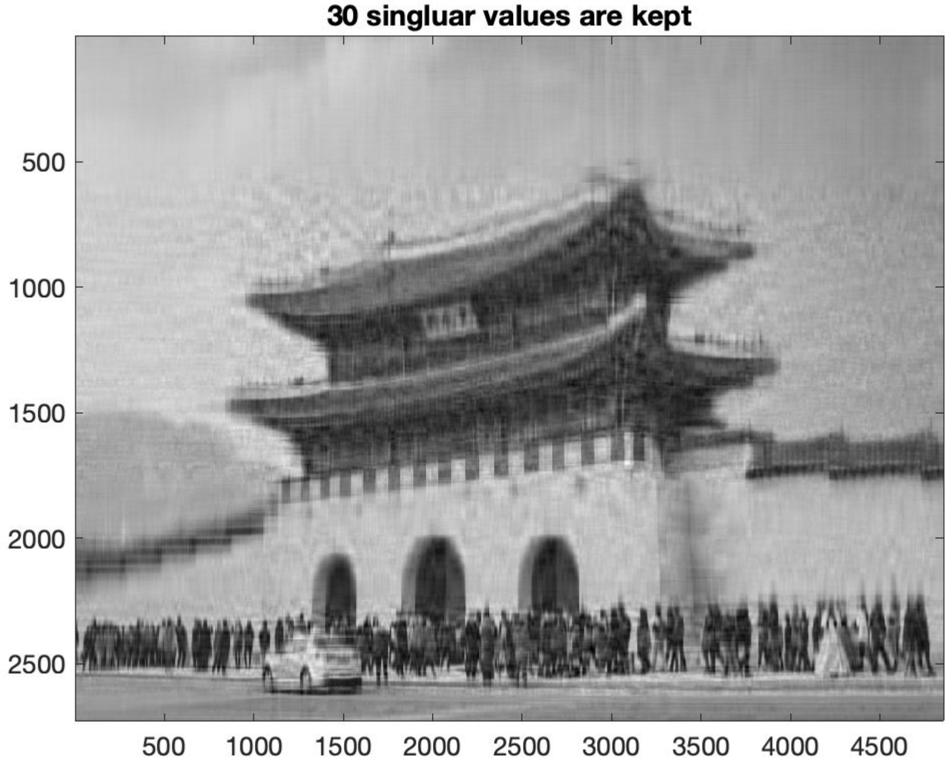
Let's compare how pictures will look like with different number of kept singular values.

```
Nkeep = [10,30,100,300]; % different number of singular values to keep

Ms = cell(numel(Nkeep),1); % cell array to contain matrices

for it = (1:numel(Nkeep))
    Ms{it} = U(:,1:Nkeep(it))*S(1:Nkeep(it),1:Nkeep(it))*V(:,1:Nkeep(it))';

    figure;
    imagesc(Ms{it});
    colormap(gray);
    set(gca,'FontSize',13)
    title([sprintf('%i',Nkeep(it)),' singluar values are kept']);
end
```

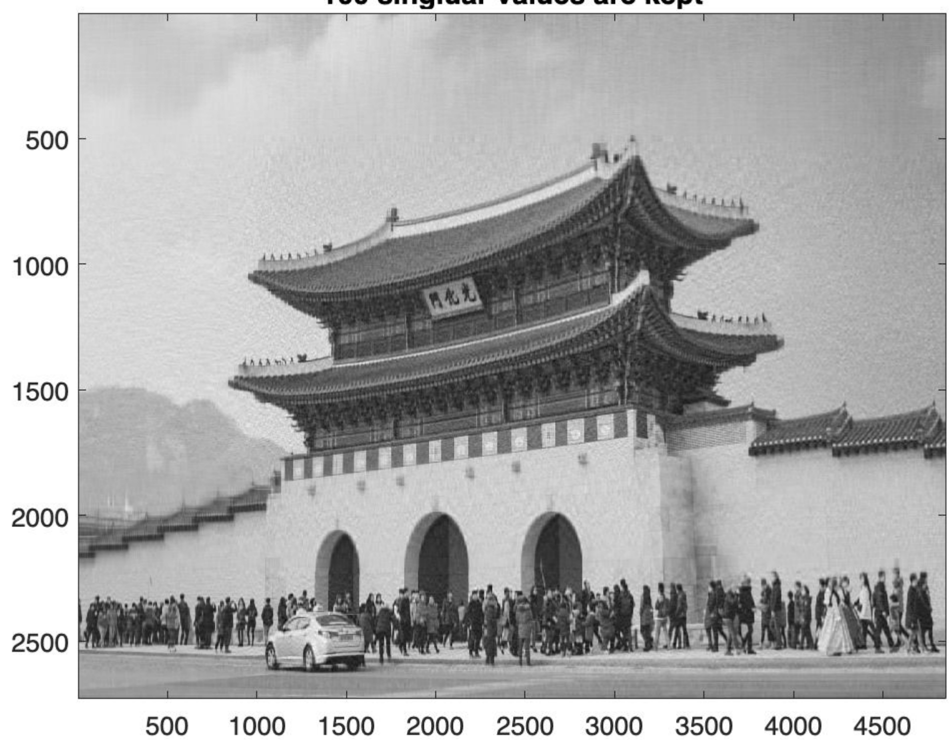
10 singluar values are kept



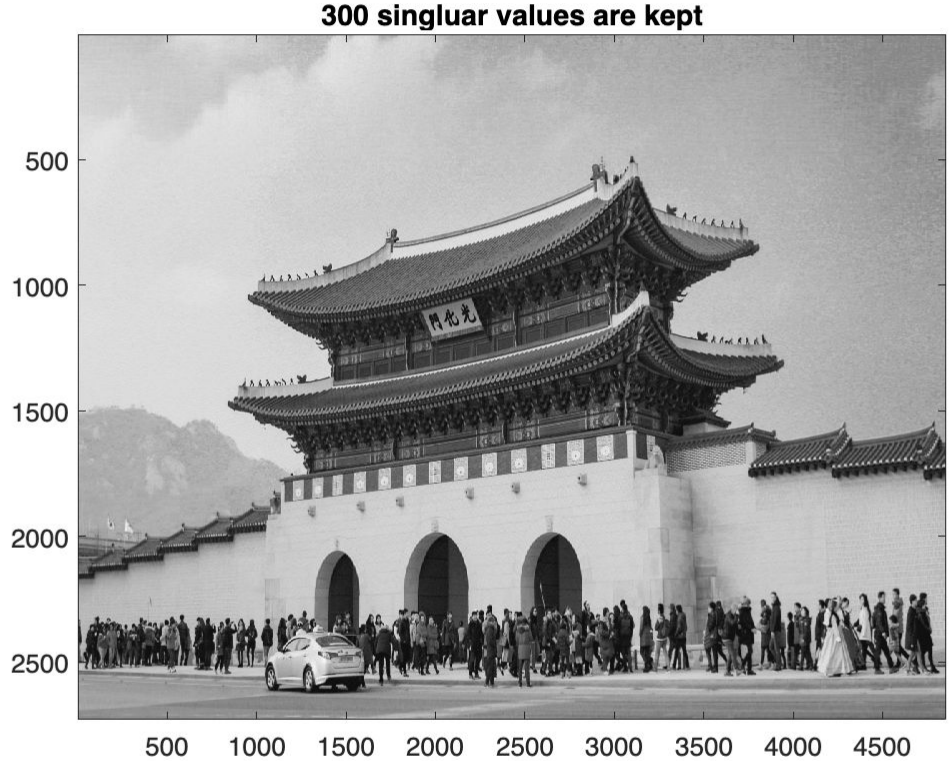
30 singluar values are kept



100 singluar values are kept



300 singluar values are kept



Only with 30 singular values, the rough shape of the building is already visible. With 100 singular values (about 3.7% of total singular values), we can recognize the name (光化門) on the signboard and distinguish pedestrians. Of course, if you zoom in, you will realize that sharp details, such as the roof on the left wall, can be properly resolved by taking 300 singular values.

Exercise (a): Understanding singular vectors

From the demonstration above, we have found that the singular vectors for the largest singular values (e.g. `U(:,(1:10))` and `V(:,(1:10))`) contribute more to the original matrix `M` than the singular vectors for the smallest singular values (e.g. `U(:,(end-9:end))` and `V(:,(end-9:end))`). Can you find the qualitative differences between the vectors for the largest singular values and the vectors for the smallest singular values? Use `fft` (Fast Fourier transform) for analyzing the vectors. The exercise is designed to make students familiar with reading and understanding MATLAB documentation. If you didn't read the documentation for `fft`, please read it through to the end.