MA 402: Project 2

Richard Watson, Mountain Chan, Cole Nash

October 4, 2020

Instructions:

- Detailed instructions regarding submission are available on the class website[1].
- The zip file should contain five files hw2.pdf, hw2.tex, classnotes.sty, swift.mat, and deblur.mat.
- More instructions:
  - MATLAB users: use `loadmat` (type `who` to display what variables are in your workspace.
  - Python users: use `scipy.io.loadmat`. This will return a dictionary with all the necessary variables.
- For plotting, you may consider using `imshow`.

1 Pen-and-paper exercises

The problems from this section total 20 points.

1) (10 points) Consider the matrix A with the SVD

$$A = \begin{bmatrix} 4 & 0 \\ -5 & -3 \\ 2 & 6 \end{bmatrix} = U \begin{bmatrix} 6\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \\ 0 & 0 \end{bmatrix} V^\top,$$

where

$$U = \frac{1}{3}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \qquad V = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

(a) (0 points) Verify for yourself that it is indeed the SVD of A, and that U, V are orthogonal.
$A = U\Sigma V^T$:

$$\begin{aligned} A &= \frac{1}{3}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 6\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \\ 0 & 0 \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \frac{3\sqrt{2}}{3\sqrt{2}} \begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -2 \\ -4 & 1 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 4 & 0 \\ -5 & -3 \\ 2 & 6 \end{bmatrix} = A \end{aligned}$$

[1] `https://github.ncsu.edu/asaibab/ma402/blob/master/project.md`

$UU^T = U^TU$:

$$U = \frac{1}{3}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$$

$$U^T = \frac{1}{3}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$$

$$UU^T = \frac{1}{9}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

$$U^TU = \frac{1}{9}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$$

$VV^T = V^TV$:

$$V = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$$

$$V^T = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$VV^T = \frac{1}{2}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

$$V^TV = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

(b) (1 point) What is the rank of this matrix?
Rank is the number of singular values, so the rank of A is 2.

(c) (2 points) From the full SVD of A, write down the thin SVD of A.

$$\begin{aligned} A &= \frac{1}{3}\begin{bmatrix} 1 & -2 \\ -2 & 1 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 6\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & -2 \\ -2 & 1 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\sqrt{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -2 \\ -4 & 1 \\ 4 & 2 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 4 & 0 \\ -5 & -3 \\ 2 & 6 \end{bmatrix} = A \end{aligned}$$

(d) (3 points) Compute the best rank-1 approximation of A.

$$\begin{aligned} A_1 &= \frac{1}{3}\begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix} 6\sqrt{2}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix}\begin{bmatrix} 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ -4 & -4 \\ 4 & 4 \end{bmatrix} \end{aligned}$$

(e) (2 points) Compute the 2-norm and the Frobenius norms of A.

2-norm :

$$||A||_2 = 6\sqrt{2} = 8.48528$$

Frobenius norm :

$$\begin{aligned} ||A||_F &= \left((6\sqrt{2})^2 + (3\sqrt{2})^2\right)^{\frac{1}{2}} \\ &= (64+18)^{\frac{1}{2}} = \sqrt{82} = 9.05539 \end{aligned}$$

(f) (2 points) Using the SVD of A, write down the SVD of $A^\top$ and $A^\top A$.

$$\begin{aligned} A^T &= (U\Sigma V^T)^T \\ &= V\Sigma^T U^T \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 6\sqrt{2} & 0 & 0 \\ 0 & 3\sqrt{2} & 0 \end{bmatrix} \frac{1}{3} \begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 2 & -1 & 0 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 4 & -5 & 2 \\ 0 & -3 & 6 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} A^T A &= (U\Sigma V^T)^T (U\Sigma V^T) \\ &= (V\Sigma^T U^T)(U\Sigma V^T) \\ &= V\Sigma^2 V^T \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 6\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix} \begin{bmatrix} 6\sqrt{2} & 0 \\ 0 & 3\sqrt{2} \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 6 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 6 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 6 & -3 \\ 6 & 3 \end{bmatrix} \begin{bmatrix} 6 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 36 & -9 \\ 36 & 9 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 45 & 27 \\ 27 & 45 \end{bmatrix} \end{aligned}$$

2) (10 points) Let $A \in \mathbb{R}^{m \times n}$. Recall: by definition, the Frobenius norm of A is $\|A\|_F = \left(\sum_i \sum_j |a_{ij}|^2\right)^{1/2}$. In this problem, we will derive the formula

$$\|A\|_F = \left(\sigma_1^2 + \cdots + \sigma_r^2\right)^{1/2}.$$

(a) The trace of a square matrix is the sum of its diagonals entries. Show (an alternative representation for the Frobenius norm):

$$\|A\|_F = \left(\mathsf{trace}\,(A^\top A)\right)^{1/2}.$$

$$A = \begin{bmatrix} \vdots & & \vdots \\ v_1 & \cdots & v_n \\ \vdots & & \vdots \end{bmatrix}$$

$$A^T = \begin{bmatrix} \cdots & v_1 & \cdots \\ & \vdots & \\ \cdots & v_n & \cdots \end{bmatrix}$$

$$A^T A = \begin{bmatrix} v_1 \cdot v_1 & & \\ & \ddots & \\ & & v_n \cdot v_n \end{bmatrix} = \begin{bmatrix} a_{11}^2 + \cdots + a_{n1}^2 & & \\ & \ddots & \\ & & a_{n1}^2 + \cdots + a_{nn}^2 \end{bmatrix}$$

$$\text{trace}(A^T A) = (a_{11}^2 + \cdots + a_{n1}^2) + \cdots + (a_{n1}^2 + \cdots + a_{nn}^2) = \sum_i^n \sum_j^n a_{ij}^2 = ||A||_F^2$$

$$||A||_F = \left(\text{trace}(A^T A)\right)^{\frac{1}{2}}$$

(b) Let C, D be $n \times n$ square matrices. Show: $\mathsf{trace}\,(CD) = \mathsf{trace}\,(DC)$.
Remark: This is known as the cyclic property of trace, which is true despite the fact that in general $CD \neq DC$. A consequence of the cyclic property is: if E has the same size as C, D, it implies

$$\mathsf{trace}\,(CDE) = \mathsf{trace}\,(DEC) = \mathsf{trace}\,(ECD).$$

$$C = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix}$$
$$D = \begin{bmatrix} d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nn} \end{bmatrix}$$
$$CD = \begin{bmatrix} c_{11}d_{11} + \cdots + c_{1n}d_{1n} & & \\ & \ddots & \\ & & c_{n1}d_{n1} + \cdots + c_{nn}d_{nn} \end{bmatrix}$$
$$\text{trace}(CD) = \sum_{i}^{n} \sum_{j}^{n} c_{ij} d_{ij}$$
$$DC = \begin{bmatrix} d_{11}c_{11} + \cdots + d_{1n}c_{1n} & & \\ & \ddots & \\ & & d_{n1}c_{n1} + \cdots + d_{nn}c_{nn} \end{bmatrix}$$
$$\text{trace}(DC) = \sum_{j}^{n} \sum_{i}^{n} d_{ji} c_{ji}$$
$$\text{trace}(CD) = \text{trace}(DC)$$

(c) Using parts (a-c) complete the proof to show $\|A\|_F = \left(\sigma_1^2 + \cdots + \sigma_r^2\right)^{1/2}$.

$$\text{trace}(A^T A) = \text{trace}(V\Sigma^2 V^T) = \text{trace}(\Sigma^2 V V^T) = \text{trace}(\Sigma^2) = \sum_{i=1}^{r} \sigma_i^2$$
$$||A||_F = \left(\sum_{i=1}^{r} \sigma_i^2\right)^{\frac{1}{2}} = \left(\text{trace}(A^T A)\right)^{\frac{1}{2}} = (\sigma_1^2 + \cdots + \sigma_r^2)^{\frac{1}{2}}$$

(d) Show: $\|A\|_2 \leq \|A\|_F \leq \sqrt{r}\|A\|_2$.

$$||A||_2 = \max_{1 \leq i \leq r} |\sigma_i| \leq \sqrt{\sigma_1^2 + \sigma_2^2} \leq \sqrt{\sigma_1^2 \ldots \sigma_r^2} = ||A||_F$$
$$\therefore \quad ||A||_2 \leq ||A||_F$$
$$||A||_F = \sqrt{\sigma_1^2 + \cdots \sigma_r^2} \leq \sqrt{r\sigma_1^2} = \sqrt{r}||A||_2$$
$$\therefore \quad ||A||_F \leq \sqrt{r}||A||_2$$
$$\therefore \quad ||A||_2 \leq ||A||_F \leq \sqrt{r}||A||_2$$

2 Numerical exercises

The problems from this section total 30 points.

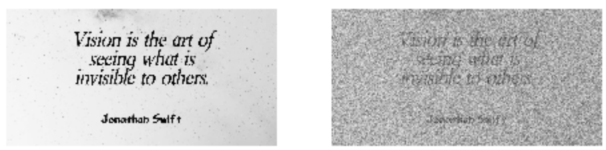
3) (15 points) *Compressing* and *Denoising* images.

(a) (0 points) Load the file 'swift.mat'. You will find the variables `A` and `An` which are both matrices of size 512×1024.

Code

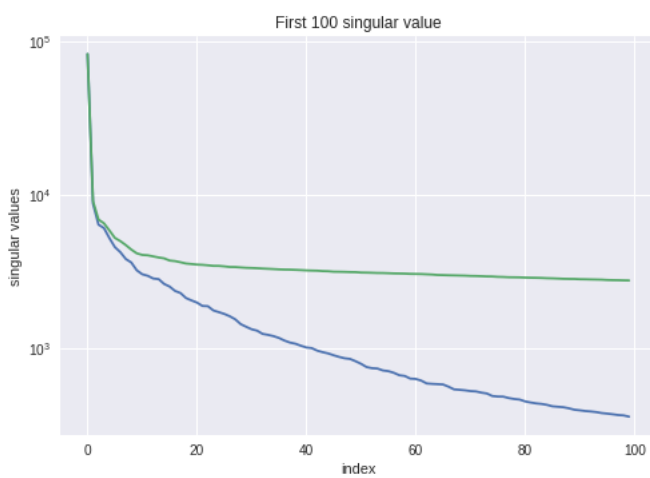
```
mat_contents = sio.loadmat('swift.mat')

A=mat_contents['A']
An=mat_contents['An']
```

(b) (2 points) In a single figure with 2 subplots, plot the clean as well as the noisy matrices as images (use `imshow`). Denote the corresponding matrices as A and $\tilde{A} = A + E$, where E is the amount of noise added to the original image. Unfortunately, in real applications we do not know exactly how much noise is added.




```
fig, (ax1,ax2) = plt.subplots(1,2)
im1 = ax1.imshow(A)
im2 = ax2.imshow(An)

ax1.axis('off')
ax2.axis('off')
```

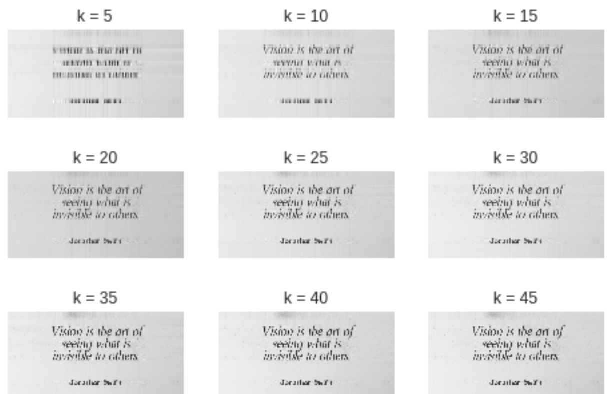
(c) (2 points) Plot the first 100 singular values of A and $\tilde{A}$. (*Hint:* Use the `semilogy` plotting function).




```
A_U, A_S, A_V = np.linalg.svd(A, full_matrices=False)
An_U, An_S, An_V = np.linalg.svd(An, full_matrices=False)
```

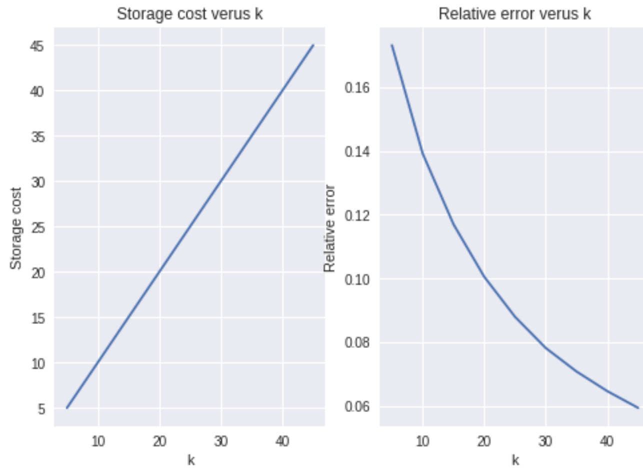
```python
fig2, ax2 = plt.subplots()
plt.semilogy(np.arange(0,100),A_S[np.arange(0,100)])
plt.semilogy(np.arange(0,100),An_S[np.arange(0,100)])
ax2.set_title('First 100 singular value')
ax2.set_xlabel('index')
ax2.set_ylabel('singular values')
```

(d) (3 points) In a single figure with 9 different subplots, plot A_k (the best rank-k approximation to A) as images for $k = 5, 10, \ldots, 45$ (use these same values of k for the rest of this problem).



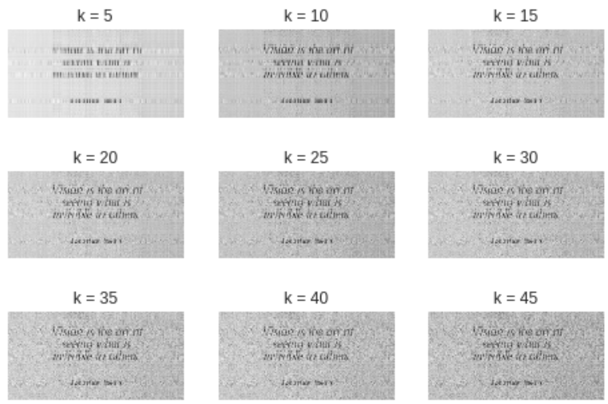

```python
ks = [5*x for x in range(1,10)]
error = np.zeros(9)

fig, ((ax1,ax2,ax3),(ax4,ax5,ax6),(ax7,ax8,ax9)) = plt.subplots(3,3)
axes = (ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9)

for i in range(9):
  k = ks[i]
  ax = axes[i]
  Ek = np.diag(A_S[:k])
  A_k = np.dot(A_U[:,:k],np.dot(Ek, A_V[:k,:]))
  error[i] = np.linalg.norm(A - A_k, 'fro') / np.linalg.norm(A,'fro')


  ax.axis('off')
  ax.imshow(A_k)
  ax.set_title('k = %d' % k)
```

(e) (2 points) As two subplots of the same figure, plot (left panel) the storage cost of the truncated SVD as a function of k, (right panel) relative error of A_k (in the Frobenius norm) as a function of k. Comment on these two subplots. (Assume that each floating point number requires 1 unit of storage.)



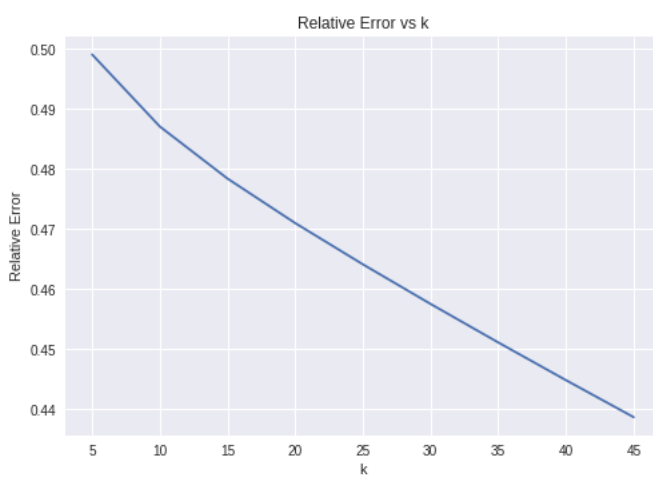

```
fig, (ax1,ax2) = plt.subplots(1,2)
cost = []

for i in range (9):
  cost.append((ks[i] * (A_k.shape[0] * A_k.shape[1] + 1)/(A_k.shape[0]*A_k.shape[1])

ax1.plot(ks, cost)
ax2.plot(ks, error)

ax1.set_title('Storage cost verus k')
ax1.set_xlabel('k')
ax1.set_ylabel('Storage cost')

ax2.set_title('Relative error verus k')
ax2.set_xlabel('k')
ax2.set_ylabel('Relative error')
```

(f) (3 points) Our proposed algorithm to denoise the image is to use a truncated SVD of the matrix corresponding to the noisy image, i.e., computing $\tilde{A}_k$. In a single figure with 9 different subplots, plot $\tilde{A}_k$ (the best rank-k approximation to A) as images for $k = 5, 10, \ldots, 45$. Make sure to label each subplot.




```
ks = [5*x for x in range(1,10)]
error = np.zeros(9)

fig, ((ax1,ax2,ax3),(ax4,ax5,ax6),(ax7,ax8,ax9)) = plt.subplots(3,3)
axes = (ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9)

for i in range(9):
  k = ks[i]
  ax = axes[i]
  Ek = np.diag(An_S[:k])
  An_k = np.dot(An_U[:,:k],np.dot(Ek, An_V[:k,:]))
  error[i] = np.linalg.norm(An - An_k, 'fro') / np.linalg.norm(An,'fro')


  ax.axis('off')
  ax.imshow(An_k)
  ax.set_title('k = %d' % k)
```

(g) (2 points) Plot the relative error of the denoised image $\tilde{A}_k$ (in the Frobenius norm) as a function of the truncation index k. For (approximately) what value of k is the minimum attained?




```
fig, ax1 = plt.subplots()

ax1.plot(ks, error)
```

```
ax1.set_title('Relative Error vs k')
ax1.set_ylabel('Relative Error')
ax1.set_xlabel('k')
```

(h) (2 bonus points) A result in perturbation analysis (due to Herman Weyl) says

$$\max_{1\leq j\leq \min\{m,n\}} |\sigma_j(A+E) - \sigma_j(A)| \leq \|E\|_2.$$

In the context of the noisy images, give an interpretation of the above equation in your words. Based on this formula, can you obtain a lower bound for the amount of noise, measured as $\|E\|_2$?

Instructions: In total, you have to submit 6 separate plots. Make sure to label each plot/subplot, and label the axes of the singular value and the error plots.

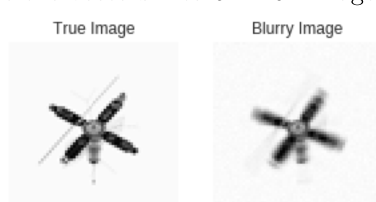
4) (15 points) *Deblurring* an image.

(a) (0 points) Load the file 'deblur.mat'. You will find the variables `A` (blurring operator, size 4096×4096) and `bn` (blurred and noisy image, size 4096×1), `xtrue` (true image, size 4096×1).

Code

```
import numpy as np
from scipy.io import loadmat
from matplotlib import pyplot as plt

# save data
dat = loadmat('deblur.mat')
A = dat['A']
bn = dat['bn']
xtrue = dat['xtrue']
```

(b) (2 points) In a single figure with 2 subplots, plot the true image, and the blurry image with noise. Note that you will have to reshape the vectors into 64×64 images.

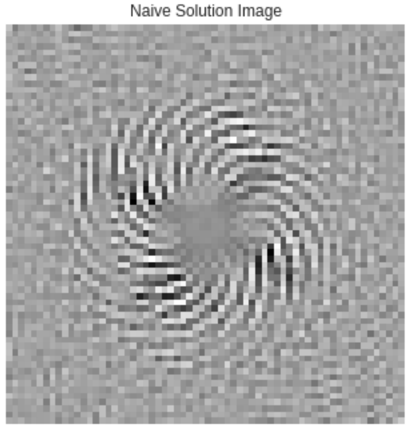


Code

```
import numpy as np
from scipy.io import loadmat
from matplotlib import pyplot as plt

# plot true and blurry image
fig, (ax1,ax2) = plt.subplots(1,2, figsize=(5,10))

# true image
ax1.grid(False)
```

```
ax1.axis('off')
ax1.imshow(np.reshape(xtrue,(64,64)))
ax1.set_title('True Image')

# blurry image
ax2.grid(False)
ax2.axis('off')
ax2.imshow(np.reshape(bn,(64,64)))
ax2.set_title('Blurry Image')
```

(c) (2 points) Recall the naive solution $x_n = A^{-1}b_n$. Plot this solution as an image. (MATLAB users should look up backslash \, and Python users should look up `numpy.linalg.solve`. Do not compute the inverse of the matrix!)



Code

```
import numpy as np
from scipy.io import loadmat
from matplotlib import pyplot as plt

# apply naive solution and plot

xn = np.linalg.solve(A,bn)
fig, ax1 = plt.subplots(1,1)
ax1.grid(False)
ax1.axis('off')
ax1.imshow(np.reshape(xn,(64,64)))
ax1.set_title('Naive Solution Image')
```

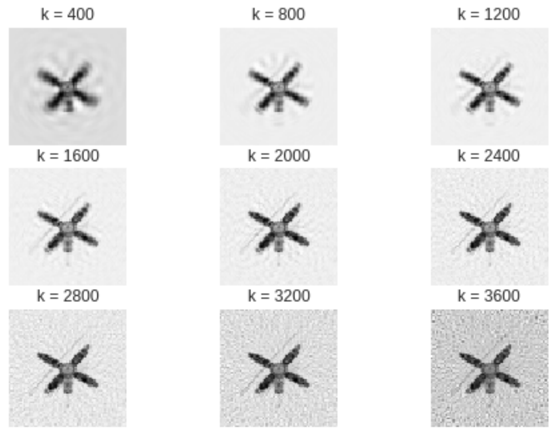
(d) (3 points) Compute the condition number $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2$ of the matrix A. Using perturbation analysis explain why you expect the naive solution to perform poorly (you are given that $\|e\|_2/\|b\|_2 = 0.05$).

The condition number is equal to 5039.94 because it is sooooo large, the error is not well conditioned and very sensitive to perturbation.

(e) (3 points) Implement the truncated SVD formula

$$x_k = \sum_{j=1}^{k} v_j \frac{u_j^\top b_n}{\sigma_j},$$

for $k = 400, 800, \ldots, 3600$. In a single figure with 9 subplots, plot the reconstructed vectors x_k as images.



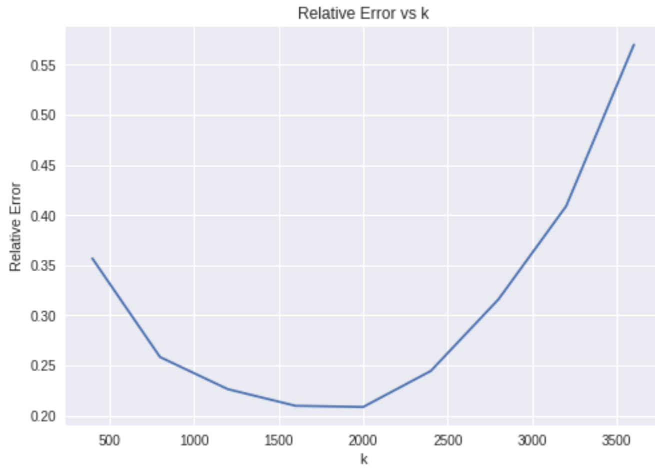
Code

```
import numpy as np
from scipy.io import loadmat
from matplotlib import pyplot as plt

# svd

U, E, VT = np.linalg.svd(A)

# apply truncated svd formula and plot
ks = [400*x for x in range(1,10)]
fig, ((ax1,ax2,ax3),(ax4,ax5,ax6),(ax7,ax8,ax9)) = plt.subplots(3,3)
axes = (ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9)

# transpose matrices
UT = np.matrix.transpose(U)
V = np.matrix.transpose(VT)

# save error
error = np.zeros(9)
trueNorm = np.linalg.norm(xtrue, 2)

for i in range(9):
  k = ks[i]
  ax = axes[i]
  Ek = np.diag(1/E[:k]) # just use k singular values
  # apply formula using matrix multiplication
  xk = np.dot(V[:,:k],np.dot(Ek, UT[:k,:]))
  xk = np.matmul(xk, np.reshape(bn,(bn.shape[0],1)))
  error[i] = np.linalg.norm(xtrue - xk, 2) / trueNorm
  # plot image
  ax.grid(False)
  ax.axis('off')
  ax.imshow(np.reshape(xk,(64,64)))
```

```
ax.set_title('k = %d' % k)
```

(f) (3 points) Plot the relative error in the reconstructed solution as a function of k. For (approximately) what value of k is the minimum attained?

The minimum is attained at $k = 2000$



Code

```python
import numpy as np
from scipy.io import loadmat
from matplotlib import pyplot as plt

# plot error
fig, ax1 = plt.subplots(1,1)

ax1.plot(ks, error)
ax1.set_title('Relative Error vs k')
ax1.set_ylabel('Relative Error')
ax1.set_xlabel('k')
```

(g) (2 points) In your words, explain the behavior of the error as a function of k.

If k is smaller than 2000 then the truncated image is missing certain details and if it is larger than 2000, noise begins to dominate the reconstruction.

Instructions: In total, you have to submit 4 separate plots. Make sure to label each plot/subplot, and label the axes of the error plots.