Using the SVD for Image Compression

Use the code snippet below to transform an image of your choice (extension `.png` or `.jpg`) into gray scale and to load it as an array $A \in \mathbb{R}^{m \times n}$ into Python.

1. Find a `scipy` routine to compute the SVD $U \Sigma V^T = A$.
2. Plot the singular values.
3. For several $1 \leq k \leq \text{rank}(A)$:
   a) Compute the *truncated SVD*: Use only the first k columns of U, the first k singular values $\sigma_1, \dots, \sigma_k$ from Σ and the first k rows of V^T to reconstruct A and plot the resulting image A_k using `plt.imshow(A_k, cmap='gray')`.
   b) For each k, compute the total number of floats that need to be stored for the truncated SVD A_k and compare it to the total number of floats that need to be stored for the full image A.

```
def load_image_as_gray(path_to_image):
    import matplotlib
    img = matplotlib.image.imread(path_to_image)
    print(np.shape(img))
    # ITU-R 601-2 luma transform (rgb to gray)
    img = np.dot(img, [0.2989, 0.5870, 0.1140])
    return img
```

Solution:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as linalg

# load image as matrix
path_to_image = 'spider.jpg'


def load_image_as_gray(path_to_image):
    import matplotlib
    img = matplotlib.image.imread(path_to_image)
    print(np.shape(img))
    # ITU-R 601-2 luma transform (rgb to gray)
    img = np.dot(img, [0.2989, 0.5870, 0.1140])
    return img


A = load_image_as_gray(path_to_image)

# SVD
U, sigma, Vt = linalg.svd(A)
m, n = np.shape(A)

# plot sigma
# note: the subplot routines should be fixed in the future
plt.figure()
plt.subplot(3, 3, 1)
plt.title(r'$\sigma$')
x = np.linspace(1, sigma.size, sigma.size)
plt.semilogy(x, sigma, 'o-', label='sigma', markersize=3)
plt.semilogy([100, 100, 0], [0, sigma[100],sigma[100]], '-', lw=2, color='red')
plt.grid(True)

# plot truncated svd images
K = [1, 3, 5, 10, 20, 50, 100]
for k in K:
    aux = U[:, :k] @ np.diag(sigma[:k]) @Vt[:k, :]
    print("\nrank =", k, "\nrelative storage (truncated-SVD/A):",
          np.round(100 * k * (1. + m + n) / (m * n), 2), "%")
    plt.subplot(3, 3, K.index(k)+2)
    plt.imshow(aux, cmap='gray')
    plt.title('k='+'{:d}'.format(k))
    plt.tight_layout(pad=0.4, w_pad=0.05, h_pad=0.5)

# plot original
plt.subplot(3, 3, 9)
plt.imshow(A, cmap='gray')
plt.title("original")


# ====================== #
#    generate a video
# ====================== #
#from matplotlib import animation
#dpi = 250
#idlist = range(120)
#frames = [] # for storing the generated images
#fig = plt.figure()
#for k in range(len(idlist)):
#    i=idlist[k]
#    aux =  np.zeros((m, 2*n+50))
#    aux[:,:n] = U[:, :i]@np.diag(sigma[:i])@Vt[:i, :]
#    aux[:,-n:] = A
#    storage = np.round(100* i * (1. + m + n)/(m*n), 2)
#    frames.append([plt.imshow(aux, cmap='gray'), plt.text(1,1, "k="+str(idlist[i])+ ",
    storage="+str(storage)+"%",horizontalalignment='left',  verticalalignment='top' ,color = "
    white" ),plt.title("Truncated SVD A_k          VS         Original A         ")])
#    #plt.title("Truncated SVD A_k          VS         Original        \n"+" k =  "+str(idlist
    [i])+ ",  storage = "+str(storage[i])+"%                                    " )])#, vmin =
    Amin, vmax = Amax, animated=True,cmap=cm.Greys_r), plt.text(8, 0.99, 'A_'+str(i), color='
    white')])
#ani = animation.ArtistAnimation(fig, frames, interval=180, blit=True,  repeat_delay=1000)
#ani.save('movie.mp4',dpi=dpi)
#plt.show()
```