**Applied Computational Science I**

# Lab: Data Compression in Image Processing using SVD

Anand Kamble

amk23j@fsu.edu

19th October 2023

---

In [1]:
```
# Import required packages.
import os
import numpy as np
import matplotlib.pyplot as plt
from IPython.display import Markdown, display, Latex
```

---

## Task 1

In [2]:
```
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9],
              [10, 11, 12]])
```

In [3]:
```
U,S,V = np.linalg.svd(A)
```

In [4]:
```
r = 1
approxA = np.outer(U[:,r], S[r] * V[r,:])
```

In [5]:
```
# Error of the norm
error = np.linalg.norm(A - approxA,'fro')/np.linalg.norm(A)
```

In [6]:
```
display(Latex(f'The error is, $$ E = {error} $$ '))
display(Latex('And the rank-1 approx values are,'))
display(approxA)
```

The error is,

$$E = 0.9987177875338429$$

And the rank-1 approx values are,

```
array([[-0.80979033, -0.06082192,  0.68814649],
       [-0.41855027, -0.03143657,  0.35567713],
       [-0.02731022, -0.00205122,  0.02320777],
       [ 0.36392984,  0.02733413, -0.30926159]])
```

---

## Task 2

In this task, we will be using the grayscale image boat-1.png.

---

## A.

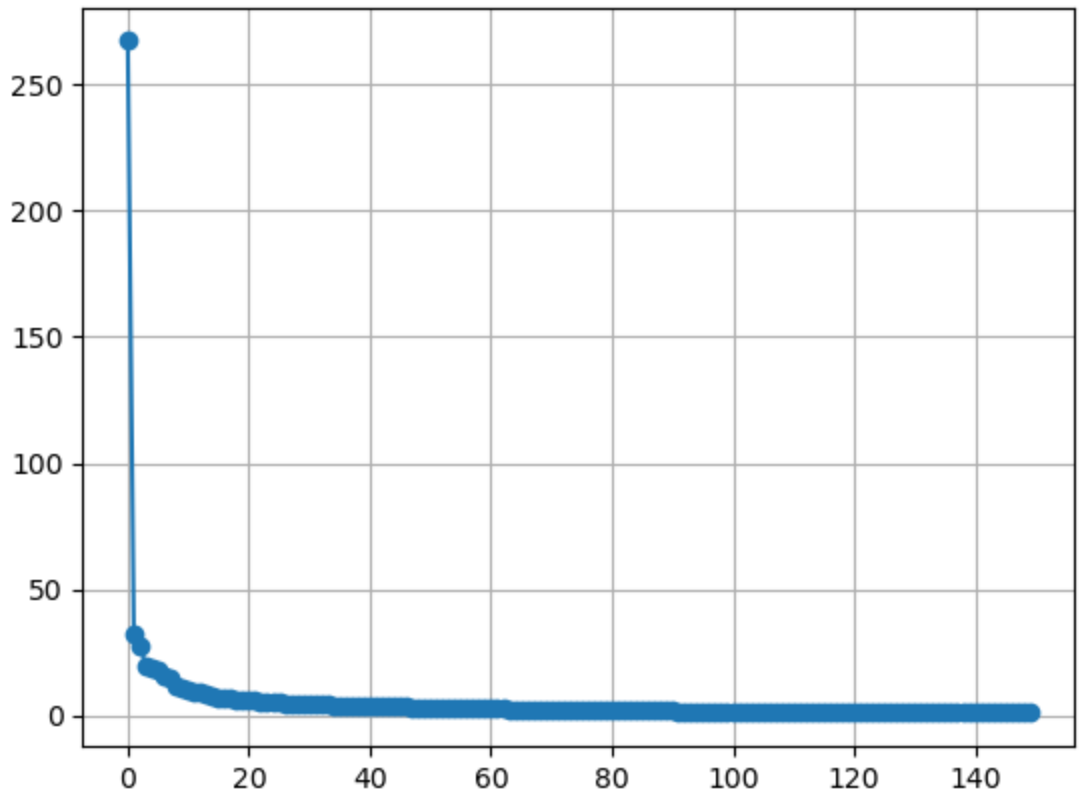
In [7]:
```
image1 = plt.imread("boat-1.png")
```

In [8]:
```
image_matrix = np.array(image1)
```

In [9]:
```
U,S,V = np.linalg.svd(image_matrix)
```

In [10]:
```
plt.grid()
display(Markdown("### First 150 singular values"))
plt.plot(S[:150],marker="o");
```

### First 150 singular values



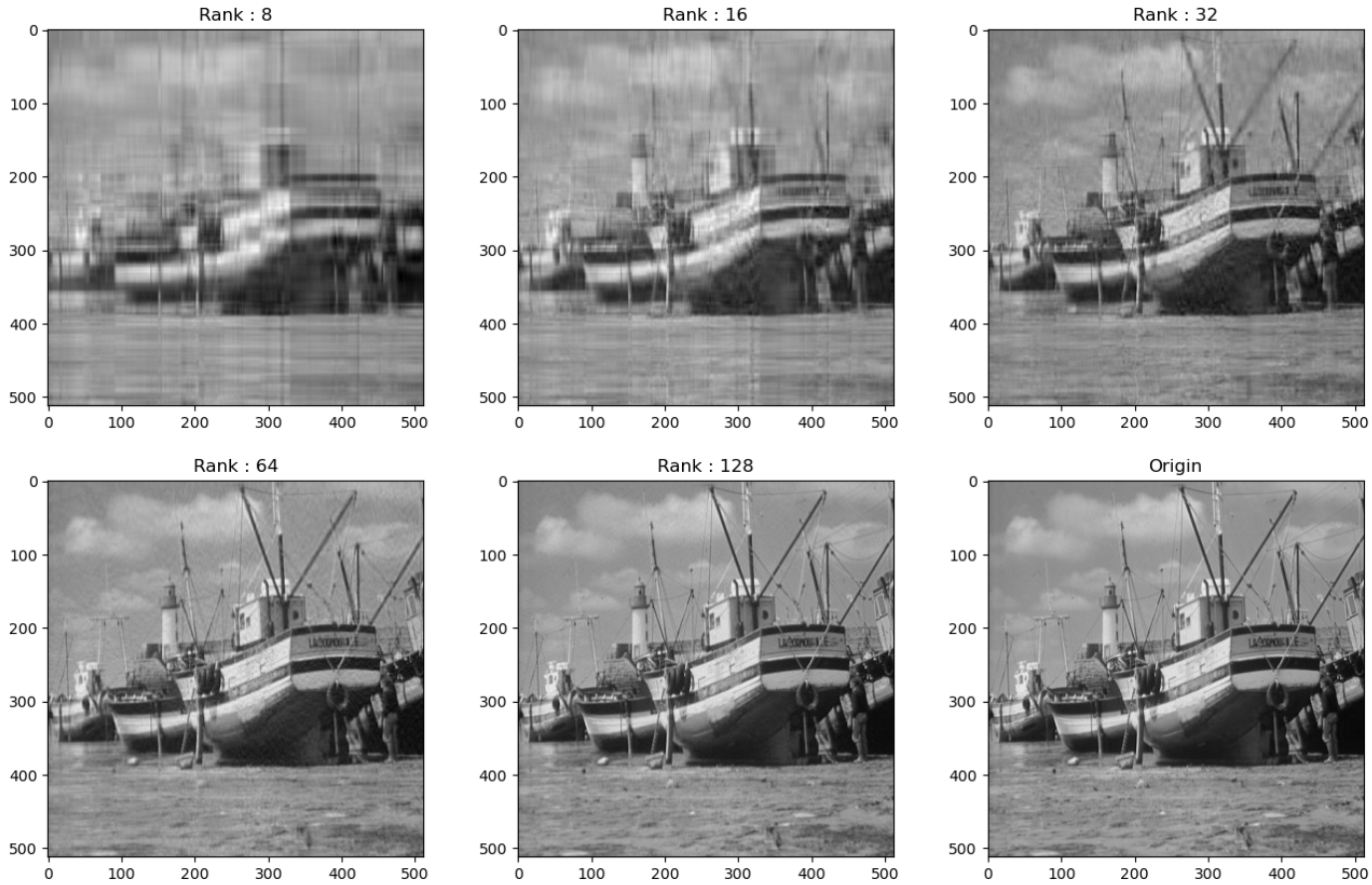

In [11]:
```
def processImage(image,rank):
#     image = plt.imread(image)
    image_matrix = np.array(image)

    U,S,V = np.linalg.svd(image_matrix)
    approxA = np.outer(U[:, 0], S[0] * V[0, :])
    for i in range(1, rank):
        approxA += np.outer(U[:, i], S[i] * V[i, :])
    error = np.linalg.norm( image_matrix - approxA,'fro') / np.linalg.norm(image_matrix)
    return approxA, error
```

---

## B

In [12]:
```
fig,axes = plt.subplots(2,3,figsize=(4 *4, 10))
```

```
ranks = [8, 16, 32, 64, 128]
i = 0
j = 0
for r in ranks:
    img, err = processImage(image1,r)
    axes[i//3,j].set_title(f"Rank : {r}")
    axes[i//3,j].imshow(img,cmap="gray")
    i =i+ 1
    j =j+1
    if j == 3:
        j = 0
axes[1,2].set_title("Origin")
axes[1,2].imshow(plt.imread("boat-1.png"),cmap="gray")
plt.plot();
```




The results above clearly show that when the rank goes up, more details in the image are kept intact. This means that as the rank increases, the image looks clearer with more detailed information.

---

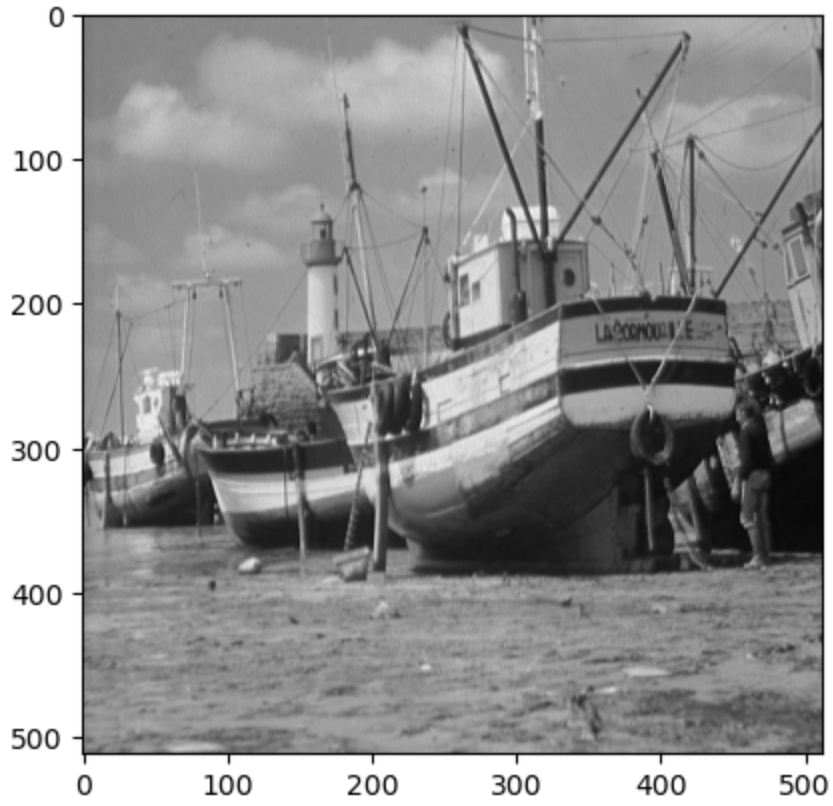
## C.

In [13]:
```
rank = 1
error = float('inf')
max_err = 0.005

for i in range(512):
    global error
    img, err = processImage(image1,rank)
    error = err
    rank = rank + 1
    if error <= max_err:
        break

display(Latex(f'At rank ${rank}$ the error drops below 0.5% which is ${error}$'))
```

At rank 362 the error drops below 0.5% which is 0.004994768649339676

In [14]:
```
img, err = processImage(image1,rank)
plt.imshow(img,cmap="gray")
plt.savefig("Compressed Image.png")

display(Markdown(f"Storage required for the image compressed with approx 0.5% loss is {o
```

Storage required for the image compressed with approx 0.5% loss is 160703 bytes, which is **17059 less bytes** than the original image.




Note : Since the image is saved with some white space around it and the axes, it is taking a bit more space than it should if the matrix was saved directly as a png image using some other package.

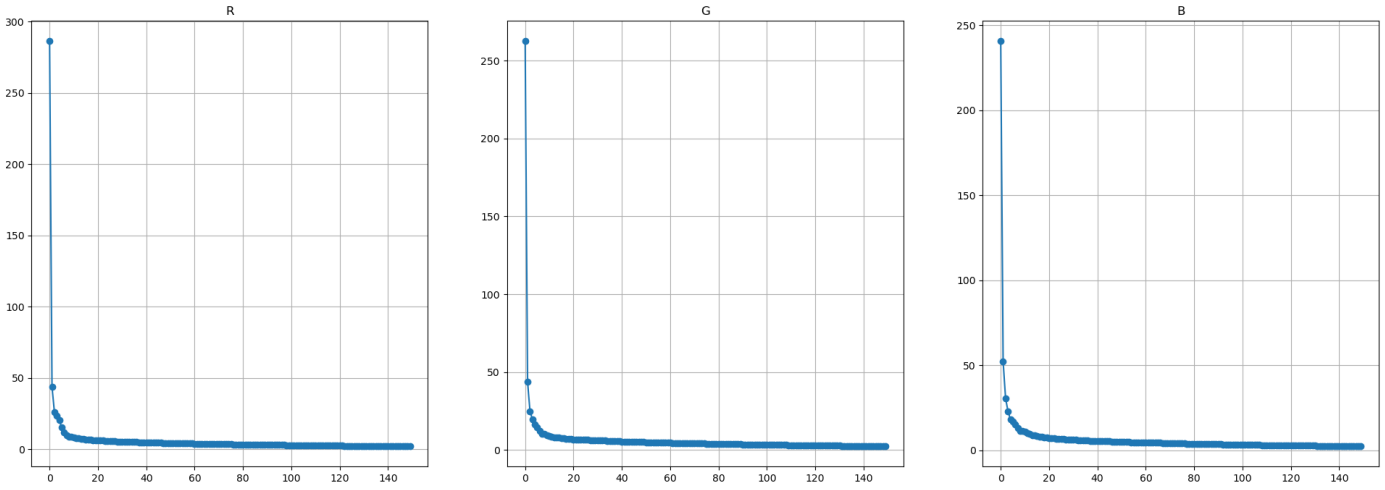
---

# Task 3

Executing identical compression on a color image involves processing data from three distinct channels: Red, Green, and Blue (RGB). This includes the separation of the Red, Green, and Blue values, followed by the computation of singular values for each channel independently.

In [15]:
```
colour_image = plt.imread("mandrill-1.png")

display(Markdown("### First 150 singular values of the R, G, and B."))
fig,axes = plt.subplots(1,3,figsize=(24, 8))
axes[0].set_title("R");
axes[1].set_title("G");
axes[2].set_title("B");
for i in range(3):
    U,S,V = np.linalg.svd(colour_image[:,:,i])
    axes[i].grid();
    axes[i].plot(S[:150],marker="o");
```

## First 150 singular values of the R, G, and B.




The graphs presented above illustrate a rapid decay in singular values upto the range of 15-20, beyond which their decline occurs more gradually with the increasing rank.

---

In [16]:
```
def ProcessColorImage(image,rank):
    image = plt.imread(image)
    image_matrix = np.array(image)

    U_red, S_red, V_red = np.linalg.svd(image_matrix[:, :, 0])
    U_green, S_green, V_green = np.linalg.svd(image_matrix[:, :, 1])
    U_blue, S_blue, V_blue = np.linalg.svd(image_matrix[:, :, 2])

    approxA_red = np.outer(U_red[:, 0], S_red[0] * V_red[0, :])
    approxA_green = np.outer(U_green[:, 0], S_green[0] * V_green[0, :])
    approxA_blue = np.outer(U_blue[:, 0], S_blue[0] * V_blue[0, :])

    for i in range(1, rank):
        approxA_red += np.outer(U_red[:, i], S_red[i] * V_red[i, :])
        approxA_green += np.outer(U_green[:, i], S_green[i] * V_green[i, :])
        approxA_blue += np.outer(U_blue[:, i], S_blue[i] * V_blue[i, :])

    approxA = np.stack([approxA_red, approxA_green, approxA_blue], axis=-1)
    return approxA
```

In [17]:
```
fig,axes = plt.subplots(2,3,figsize=(4 *4, 10))

ranks = [8, 16, 32, 64, 128]
i = 0
j = 0
for r in ranks:
    axes[i//3,j].set_title(f"Rank : {r}")
    axes[i//3,j].imshow(ProcessColorImage("mandrill-1.png",r))
    i =i+ 1
    j =j+1
    if j == 3:
        j = 0
axes[1,2].set_title("Origin")
axes[1,2].imshow(plt.imread("mandrill-1.png"))
plt.plot();
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[0..255] for integers).
```