hw6_code

October 6, 2021

Michael Goforth
CAAM 550
HW 6
10/6/21

Problem 1 Part d

```
[11]: import numpy as np
import pandas as pd
nvec = [10, 20, 30]
columnnames = ["n", "1/sigma_n", "||x-xcomp||2", "||x-xcomp||2*sigma_n"]
data = pd.DataFrame(columns = columnnames)
for n in nvec:
    # Construct A matrix
    A = np.zeros((n, n))
    for i in range(0, n):
        ti = -1 + 2 * (i) / (n - 1)
        for j in range(0, n):
            A[i, j] = ti**j
    xex = np.ones((n, 1))
    b = A @ xex
    xcomp = np.linalg.solve(A, b)
    comp2 = np.linalg.norm(xex-xcomp, 2)
    U, Sigma, VT = np.linalg.svd(A)
    sigma_n = Sigma[-1]
    comp1 = 1/ sigma_n
    comp3 = comp2 * sigma_n
    data = data.append({'n': n, "1/sigma_n" : comp1, "||x-xcomp||2" : comp2,␣
 ↪"||x-xcomp||2*sigma_n" : comp3}, ignore_index=True)
data['n'] = data['n'].apply('{:.0f}'.format)
print(data.to_string(index=False))
```

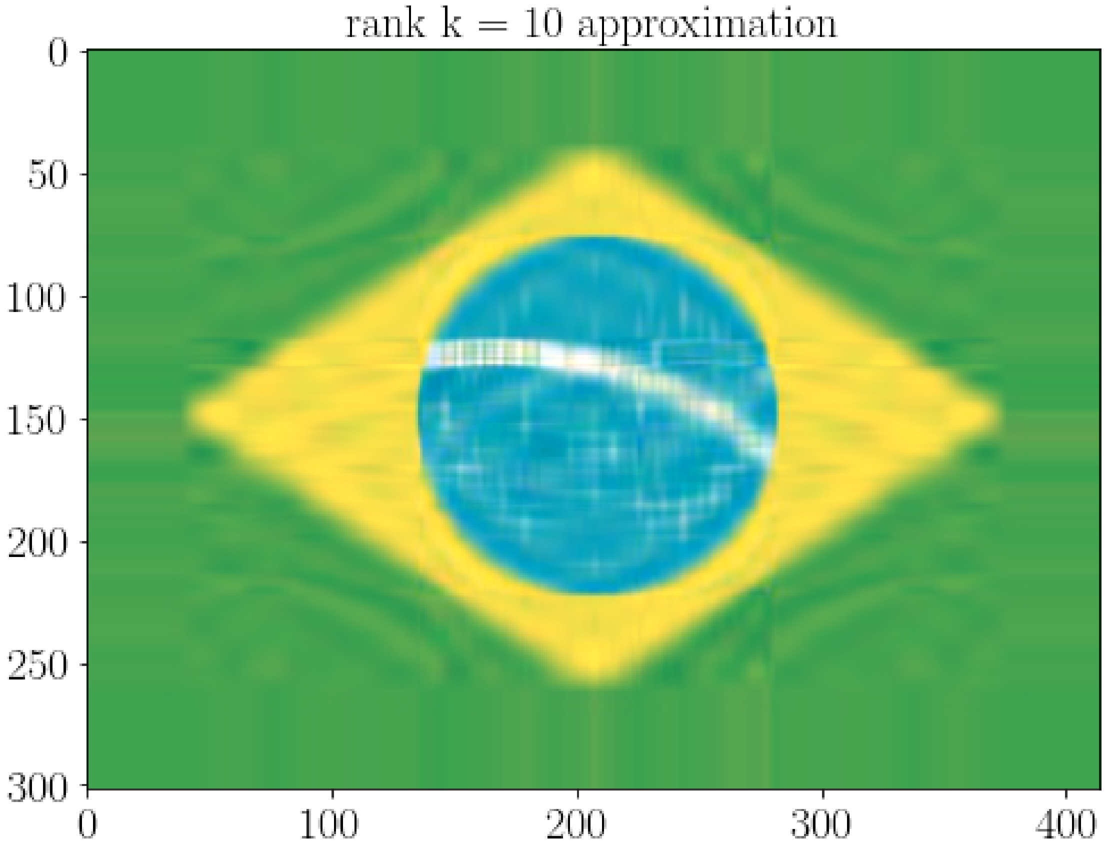
```
 n     1/sigma_n  ||x-xcomp||2  ||x-xcomp||2*sigma_n
10  1.173155e+03  3.800178e-13          3.239279e-16
20  4.789833e+07  2.600080e-08          5.428332e-16
30  2.626480e+12  6.328064e-04          2.409332e-16
```

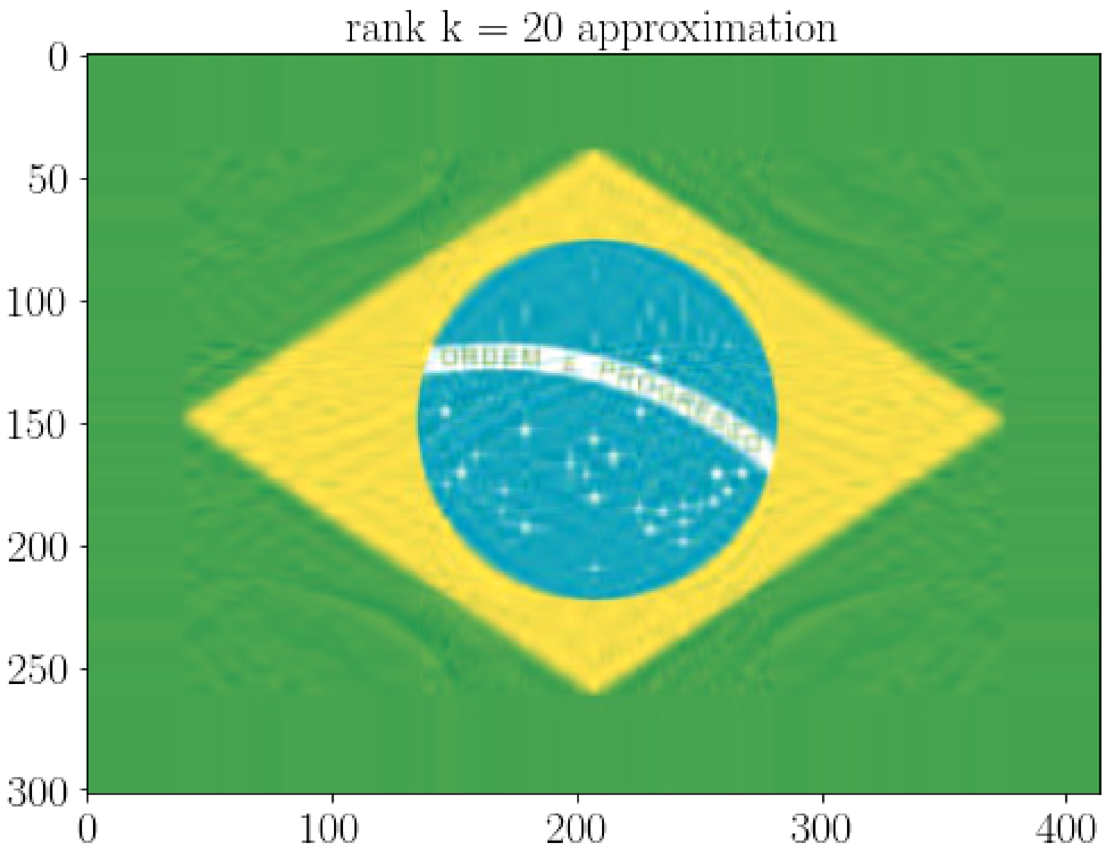
Problem 2 Much of the code is from demo_svd_image_compression.ipynb given in notes, original author Mattias Heinkenschloss.

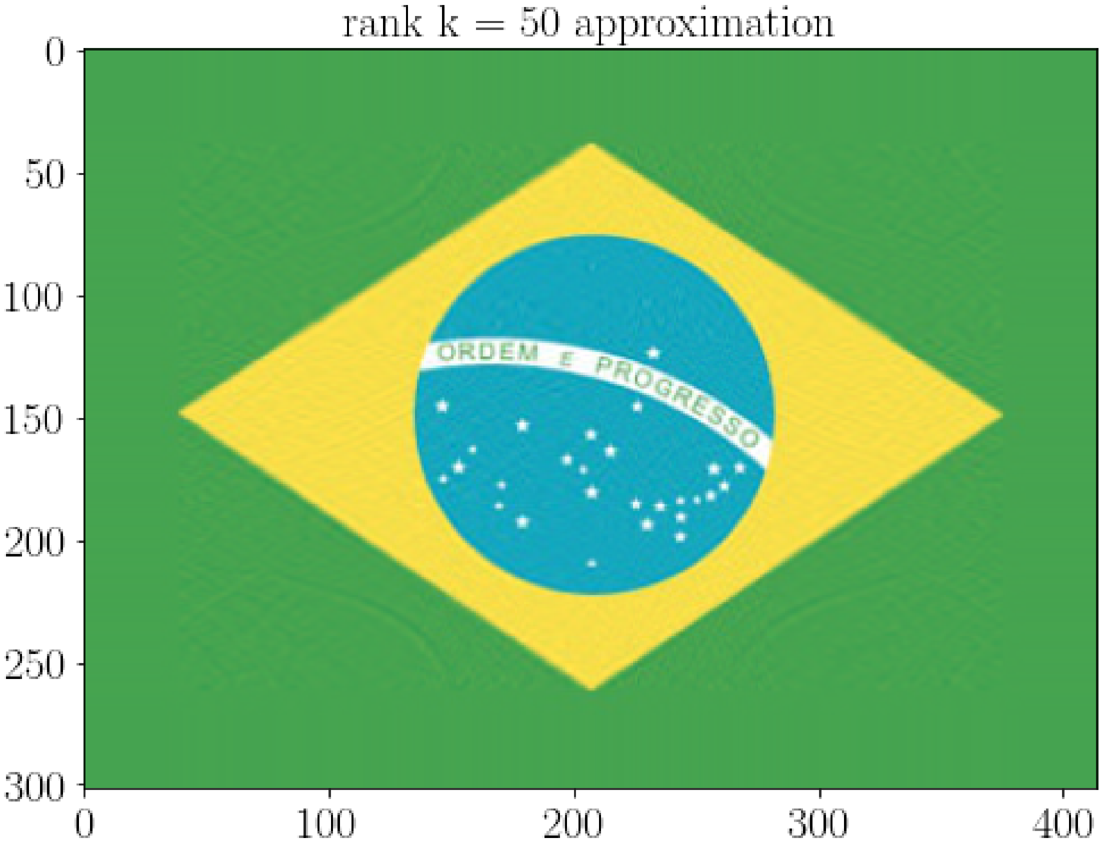
[12]:

```
# Brazil flag

# import packages
import numpy as np
# load and display an image with Matplotlib
from matplotlib import rc, rcParams
rcParams['font.size'] = 20
rc('font', family='sans-serif')
rcParams['font.family'] = 'Serif'
rcParams['font.weight'] = 'light'
rcParams['mathtext.fontset'] = 'cm'
rcParams['text.usetex'] = True
from matplotlib import image
from matplotlib import pyplot

# load image as pixel array
data = image.imread('Brazil_flag.jpg')
# summarize shape of the pixel array
#print(data.shape)
# display the array of pixels as an image
#pyplot.figure(figsize=(8,8))
#pyplot.imshow(data)
#pyplot.show()

# convert to real and scale to [0,1]. Images stores as reals need tk be between␣
 ↪[0,1]
data = data.astype(float)/255.0
#pyplot.figure(figsize=(8,8))
#pyplot.imshow(data)
#pyplot.show()

# compute SVDs or red, green, blue image matrices
Ur, Sr, Vtr = np.linalg.svd(data[:,:,0], full_matrices=False)
Ug, Sg, Vtg = np.linalg.svd(data[:,:,1], full_matrices=False)
Ub, Sb, Vtb = np.linalg.svd(data[:,:,2], full_matrices=False)

#pyplot.semilogy(Sr/Sr[0], marker='o', color = 'red')
#pyplot.semilogy(Sg/Sg[0], marker='x', color = 'green')
#pyplot.semilogy(Sb/Sb[0], marker='*', color = 'blue')
#pyplot.xlabel("j", fontsize="20")
#pyplot.ylabel("$\sigma_j/\sigma_o$", fontsize="20")
#pyplot.title("normalized singular values", fontsize="20")
#pyplot.legend(['red', 'green', 'blue'])
#pyplot.show()

data_r = np.zeros(data.shape)
```

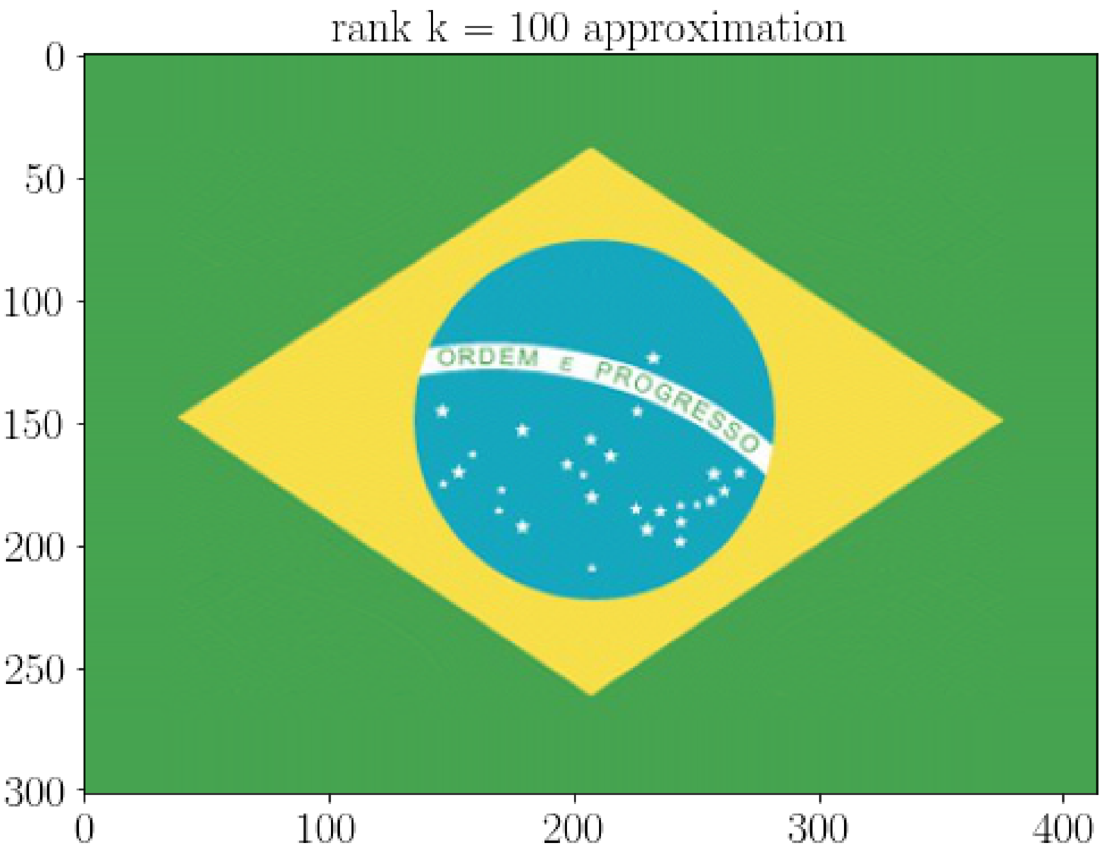
```
for i in [10, 20, 50, 100]:
    r = i
    data_r[:,:,0] = Ur[:,0:r] @ np.diag(Sr[0:r]) @ Vtr[0:r,:]
    data_r[:,:,1] = Ug[:,0:r] @ np.diag(Sg[0:r]) @ Vtg[0:r,:]
    data_r[:,:,2] = Ub[:,0:r] @ np.diag(Sb[0:r]) @ Vtb[0:r,:]

    data_r[data_r<0] = 0
    data_r[data_r>1] = 1

    pyplot.figure(figsize=(8,8))
    pyplot.imshow(data_r)
    pyplot.title('rank k = {0:5d} approximation'.format(r), fontsize="20")
    pyplot.show()
```




rank k = 20 approximation
0
50
100
150
200
250
300
0
100
200
300
400
ORDEM E PROGRESSO



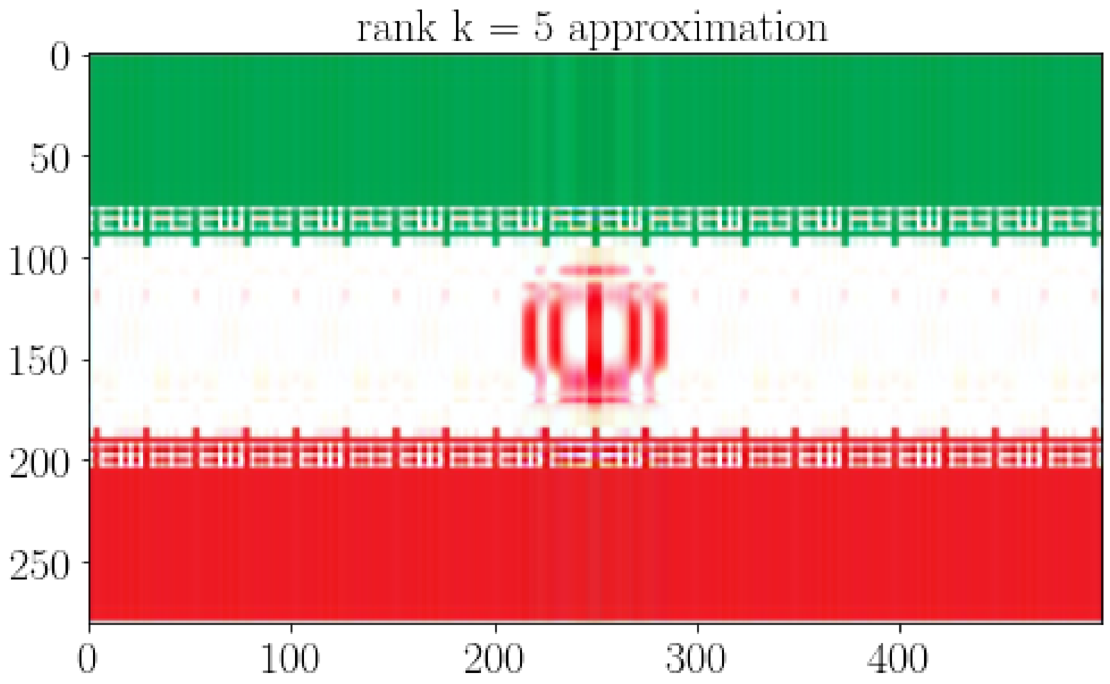
rank k = 50 approximation

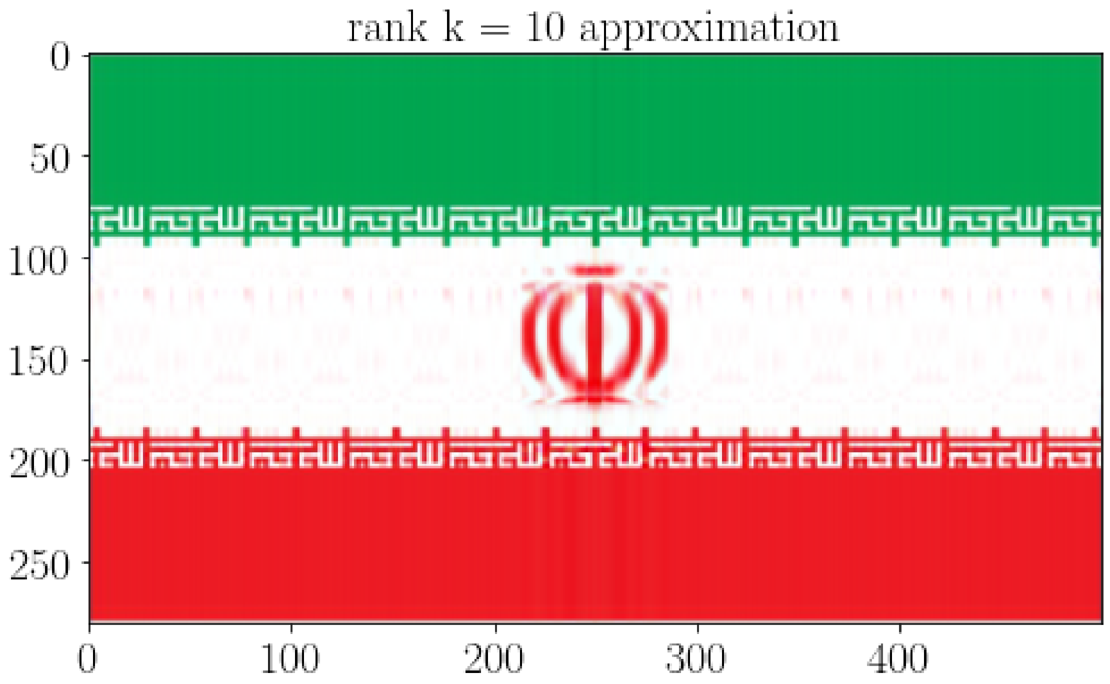


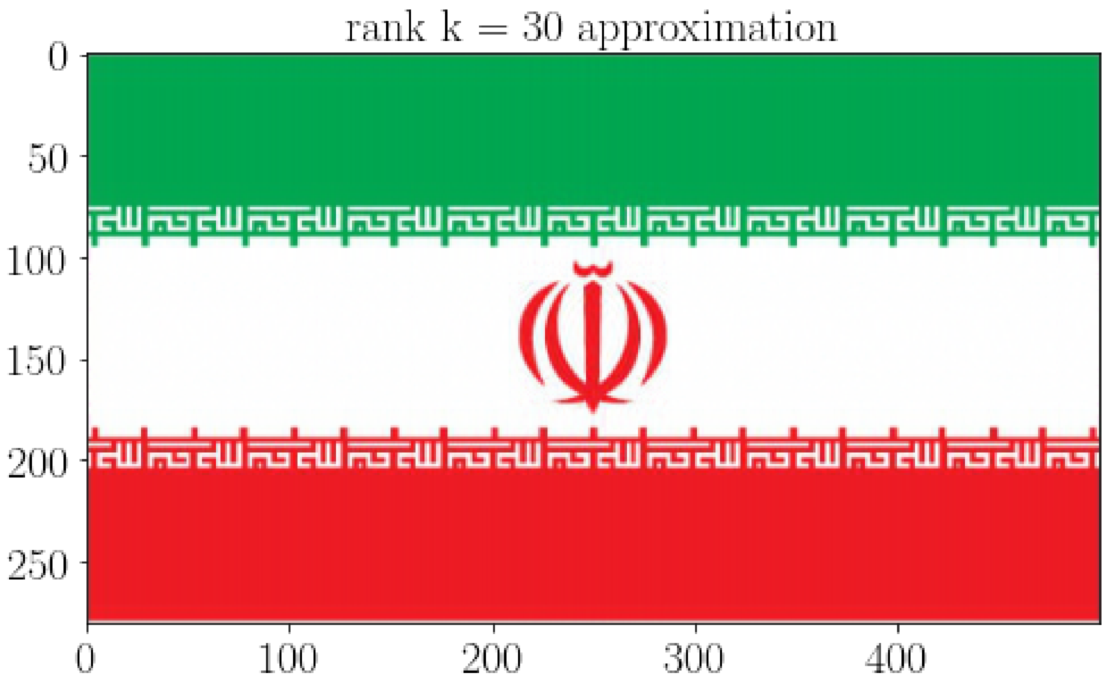
[8]:

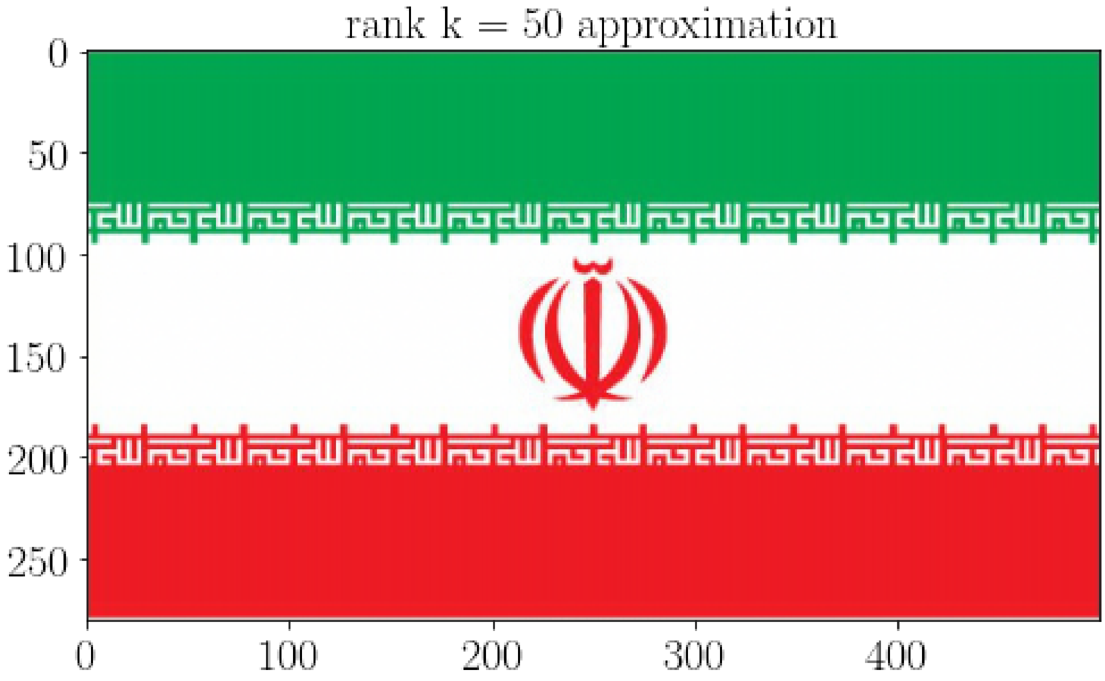
```
# Iran flag

# import packages
import numpy as np
# load and display an image with Matplotlib
from matplotlib import rc, rcParams
rcParams['font.size'] = 20
rc('font', family='sans-serif')
rcParams['font.family'] = 'Serif'
rcParams['font.weight'] = 'light'
rcParams['mathtext.fontset'] = 'cm'
rcParams['text.usetex'] = True
from matplotlib import image
from matplotlib import pyplot

# load image as pixel array
data = image.imread('Iran_flag.jpg')
# summarize shape of the pixel array
#print(data.shape)
# display the array of pixels as an image
```

```
#pyplot.figure(figsize=(8,8))
#pyplot.imshow(data)
#pyplot.show()

# convert to real and scale to [0,1]. Images stores as reals need tk be between [0,1]
data = data.astype(float)/255.0
#pyplot.figure(figsize=(8,8))
#pyplot.imshow(data)
#pyplot.show()

# compute SVDs or red, green, blue image matrices
Ur, Sr, Vtr = np.linalg.svd(data[:,:,0], full_matrices=False)
Ug, Sg, Vtg = np.linalg.svd(data[:,:,1], full_matrices=False)
Ub, Sb, Vtb = np.linalg.svd(data[:,:,2], full_matrices=False)

#pyplot.semilogy(Sr/Sr[0], marker='o', color = 'red')
#pyplot.semilogy(Sg/Sg[0], marker='x', color = 'green')
#pyplot.semilogy(Sb/Sb[0], marker='*', color = 'blue')
#pyplot.xlabel("j", fontsize="20")
#pyplot.ylabel("$\sigma_j/\sigma_o$", fontsize="20")
#pyplot.title("normalized singular values", fontsize="20")
#pyplot.legend(['red', 'green', 'blue'])
#pyplot.show()

data_r = np.zeros(data.shape)

for i in [5, 10, 30, 50]:
    r = i
    data_r[:,:,0] = Ur[:,0:r] @ np.diag(Sr[0:r]) @ Vtr[0:r,:]
    data_r[:,:,1] = Ug[:,0:r] @ np.diag(Sg[0:r]) @ Vtg[0:r,:]
    data_r[:,:,2] = Ub[:,0:r] @ np.diag(Sb[0:r]) @ Vtb[0:r,:]

    data_r[data_r<0] = 0
    data_r[data_r>1] = 1

    pyplot.figure(figsize=(8,8))
    pyplot.imshow(data_r)
    pyplot.title('rank k = {0:5d} approximation'.format(r), fontsize="20")
    pyplot.show()
```

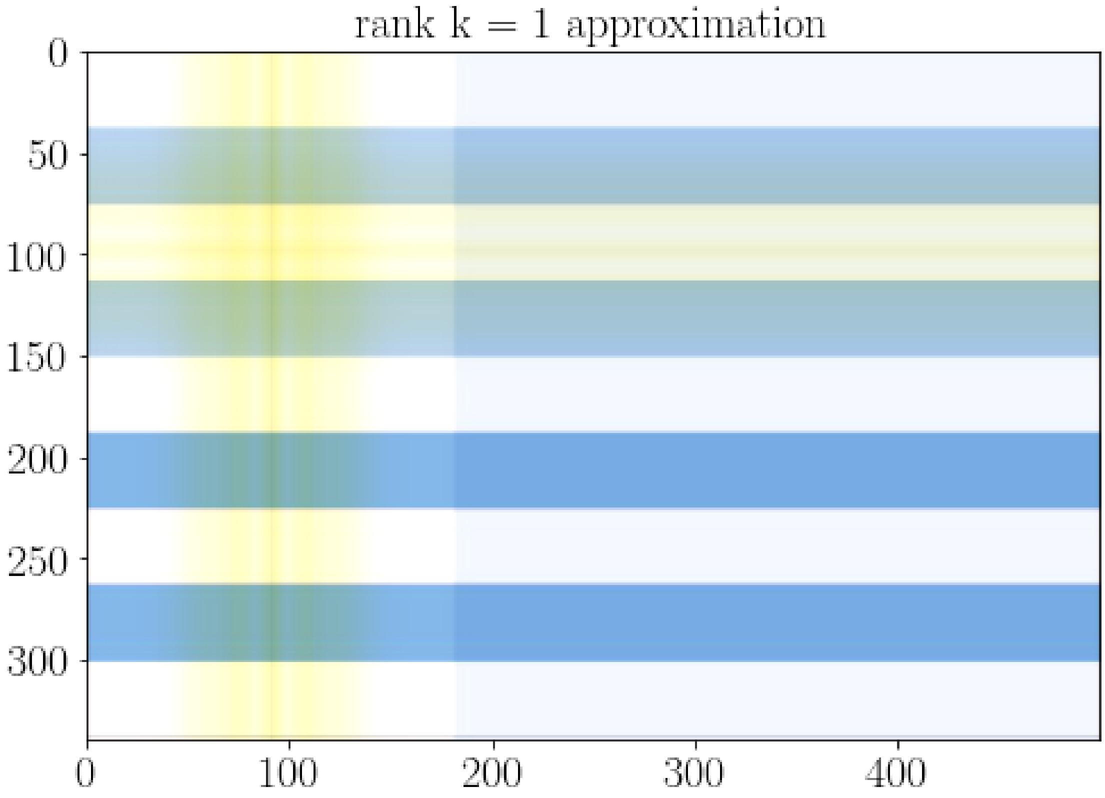

rank k = 5 approximation

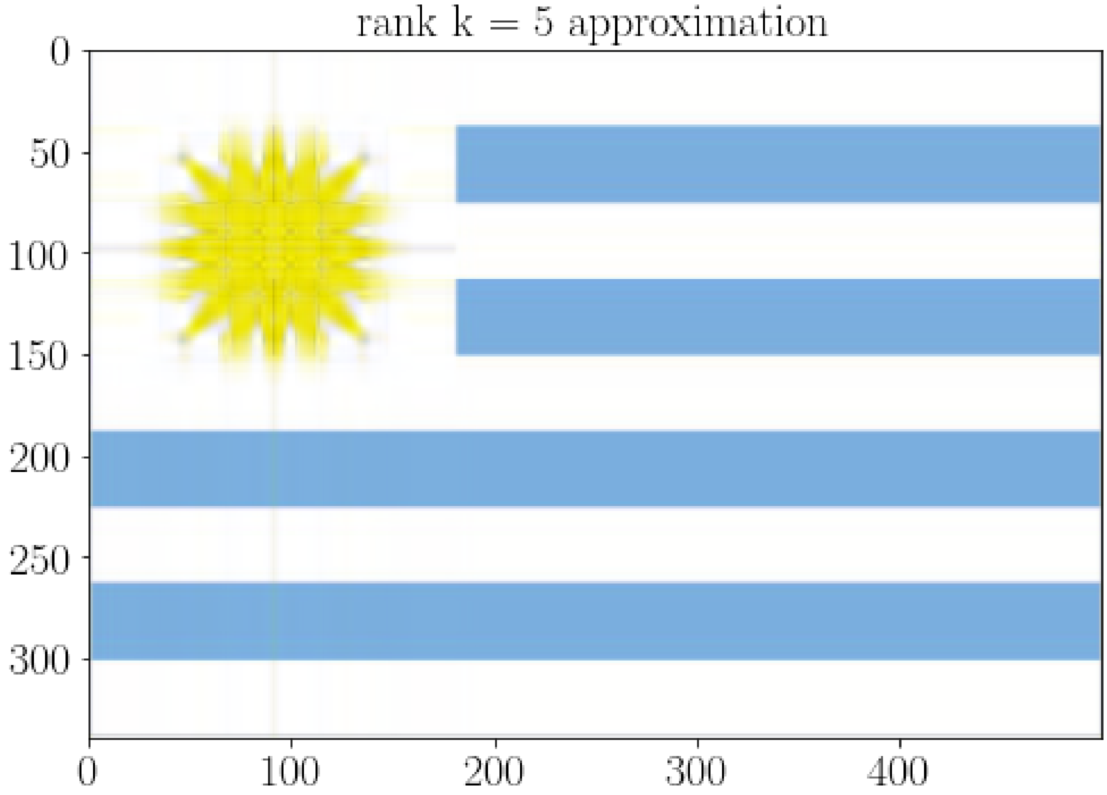

rank k = 10 approximation

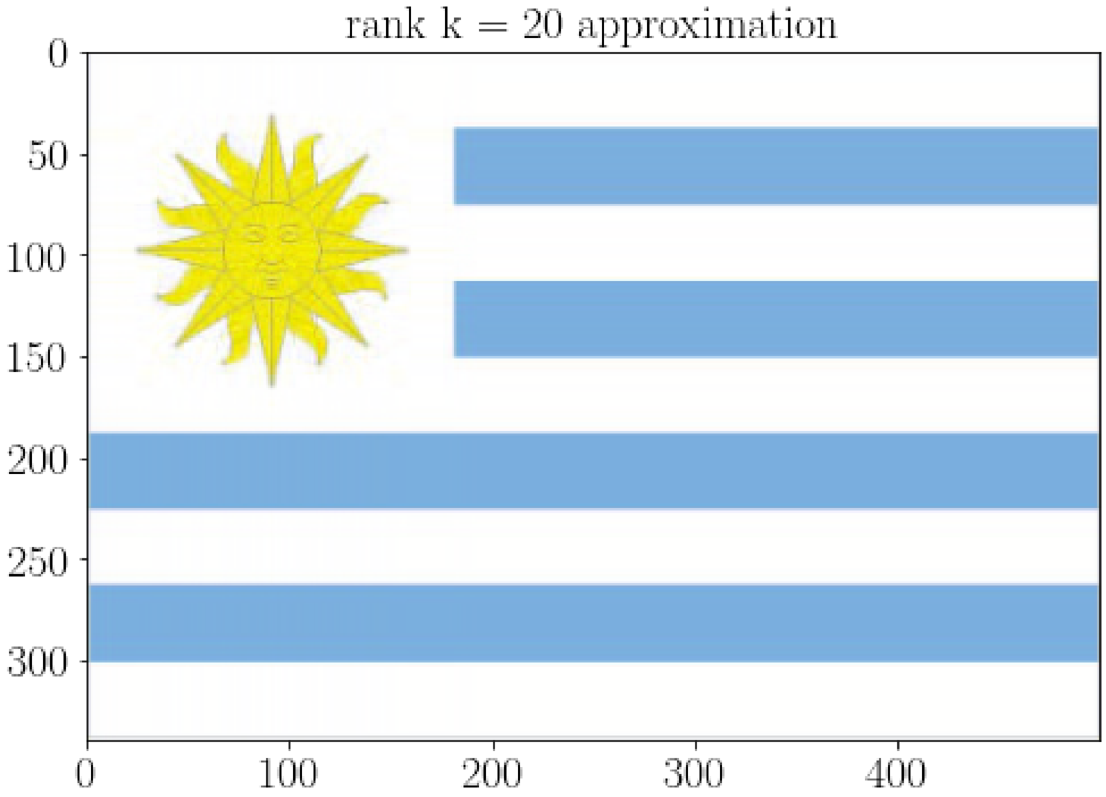


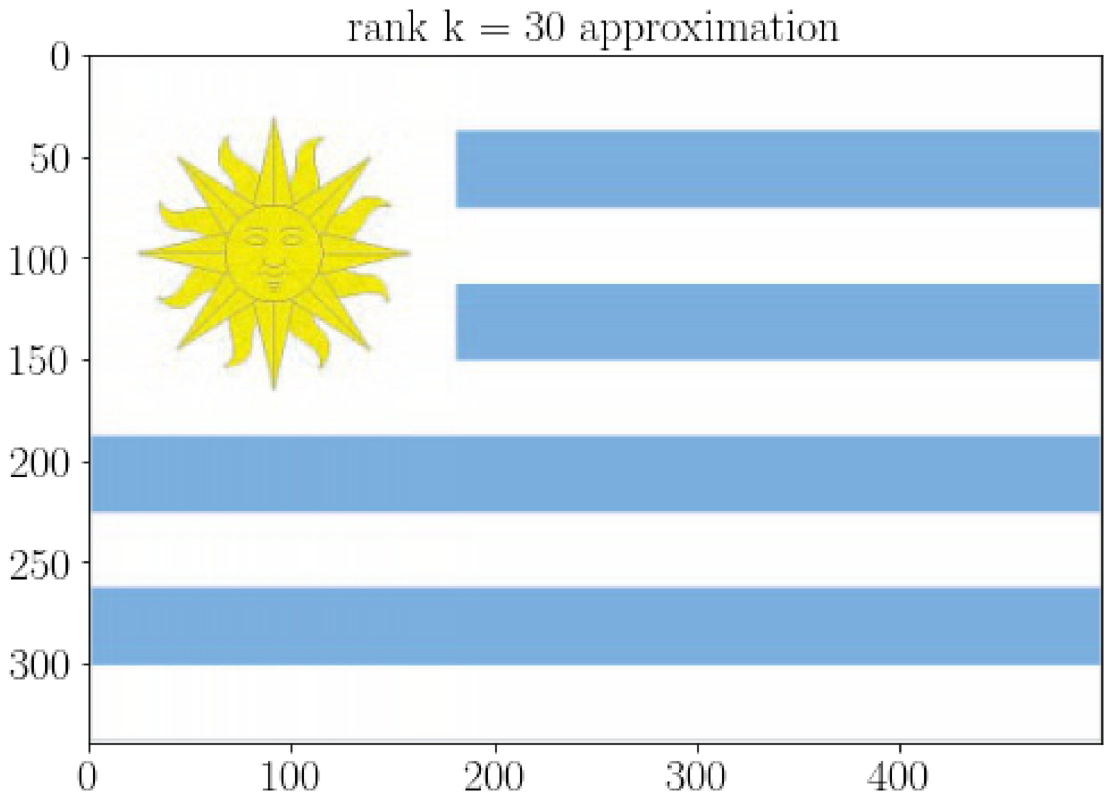


[13]:
```
# Uruguay flag
# import packages
import numpy as np
# load and display an image with Matplotlib
from matplotlib import rc, rcParams
rcParams['font.size'] = 20
rc('font', family='sans-serif')
rcParams['font.family'] = 'Serif'
rcParams['font.weight'] = 'light'
rcParams['mathtext.fontset'] = 'cm'
rcParams['text.usetex'] = True
from matplotlib import image
from matplotlib import pyplot

# load image as pixel array
data = image.imread('Uruguay_flag.jpg')
# summarize shape of the pixel array
#print(data.shape)
# display the array of pixels as an image
#pyplot.figure(figsize=(8,8))
#pyplot.imshow(data)
#pyplot.show()

# convert to real and scale to [0,1]. Images stores as reals need tk be between [0,1]
data = data.astype(float)/255.0
#pyplot.figure(figsize=(8,8))
#pyplot.imshow(data)
#pyplot.show()

# compute SVDs or red, green, blue image matrices
Ur, Sr, Vtr = np.linalg.svd(data[:,:,0], full_matrices=False)
Ug, Sg, Vtg = np.linalg.svd(data[:,:,1], full_matrices=False)
Ub, Sb, Vtb = np.linalg.svd(data[:,:,2], full_matrices=False)

#pyplot.semilogy(Sr/Sr[0], marker='o', color = 'red')
#pyplot.semilogy(Sg/Sg[0], marker='x', color = 'green')
#pyplot.semilogy(Sb/Sb[0], marker='*', color = 'blue')
#pyplot.xlabel("j", fontsize="20")
#pyplot.ylabel("$\sigma_j/\sigma_o$", fontsize="20")
#pyplot.title("normalized singular values", fontsize="20")
#pyplot.legend(['red', 'green', 'blue'])
#pyplot.show()

data_r = np.zeros(data.shape)

for i in [1, 5, 20, 30]:
    r = i
    data_r[:,:,0] = Ur[:,0:r] @ np.diag(Sr[0:r]) @ Vtr[0:r,:]
    data_r[:,:,1] = Ug[:,0:r] @ np.diag(Sg[0:r]) @ Vtg[0:r,:]
    data_r[:,:,2] = Ub[:,0:r] @ np.diag(Sb[0:r]) @ Vtb[0:r,:]

    data_r[data_r<0] = 0
    data_r[data_r>1] = 1

    pyplot.figure(figsize=(8,8))
    pyplot.imshow(data_r)
    pyplot.title('rank k = {0:5d} approximation'.format(r), fontsize="20")
    pyplot.show()
```




rank k = 5 approximation
0
50
100
150
200
250
300
0
100
200
300
400


rank k = 20 approximation
0
50
100
150
200
250
300
0
100
200
300
400



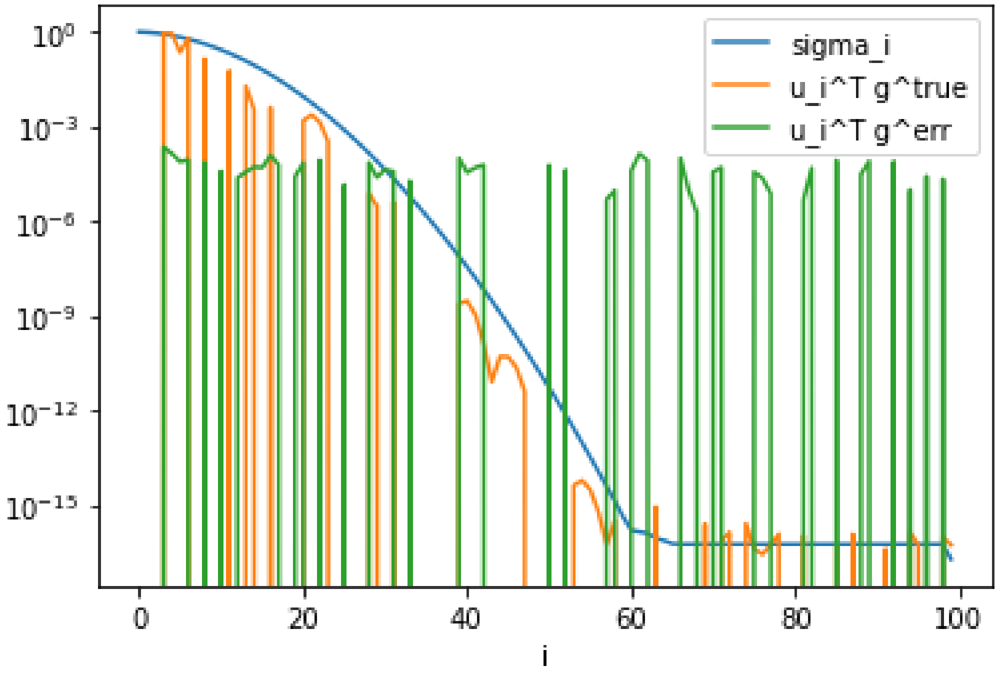
Problem 3 part i

[3]:

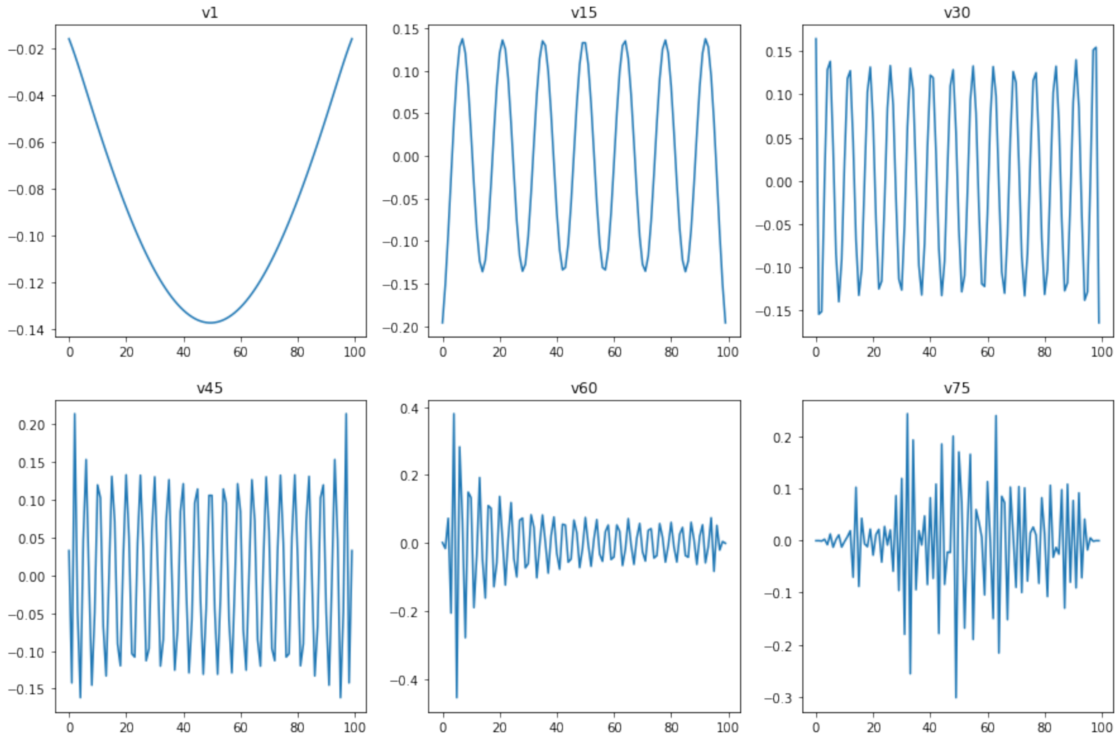
```
# Need to restart kernel to undo changes from flag examples before running this.

# import packages
import numpy as np
import math as m
import matplotlib.pyplot as plt

# specify problem
n = 100
h = 1/n
gamma = 0.05
xi = np.arange(1/2, n, dtype=float)/n
# true image
ftrue = np.exp( -(xi-0.75)**2 *70 )
ind = np.all([0.1<=xi, xi<=0.25], axis =0)  # indices for which xi in [0.1,0.25]
ftrue[ind] = 0.8
ind = np.all([0.3<=xi, xi<=0.35], axis =0)  # indices for which xi in [0.3,0.35]
ftrue[ind] = 0.3
# matrix K
```

```
C = 1/(gamma*np.sqrt(2*np.pi))
K = np.zeros((n,n))
for i in np.arange(n):
    for j in np.arange(n):
        K[i,j] = C*h* np.exp( -(xi[i]-xi[j])**2 / (2*gamma**2) )

gtrue = np.dot(K, ftrue)

#fig, ax = plt.subplots()
#ax.plot(xi, ftrue, '-k')
#ax.plot(xi, gtrue, '-.r')
#ax.legend(['true image','blurred image'])
#ax.set(xlabel='xi')

#fig.savefig("denoise_1d_ftrue")
#plt.show()

# add error to true image
gerr = 0.001*np.multiply( 0.5 - np.random.uniform(0.05,0.5,n) , gtrue )
g = gtrue + gerr
gerror = 0.5*np.linalg.norm(gerr, 2)**2

U, Sigma, VT = np.linalg.svd(K)
V = np.transpose(VT)
x = [i for i in range(g.size)]
y2 = [np.inner(U[:, j], gtrue) for j in x]
y3 = [np.inner(U[:, j], gerr) for j in x]

plt.semilogy(x, Sigma)
plt.semilogy(x, y2)
plt.semilogy(x, y3)
plt.legend(['sigma_i', 'u_i^T g^true', 'u_i^T g^err'])
plt.xlabel('i')
plt.show()

fig, ax = plt.subplots(2, 3, figsize=(15,10))
ax[0, 0].plot(V[:, 0])
ax[0, 0].set_title('v1')
ax[0, 1].plot(V[:, 14])
ax[0, 1].set_title('v15')
ax[0, 2].plot(V[:, 29])
ax[0, 2].set_title('v30')
ax[1, 0].plot(V[:, 44])
ax[1, 0].set_title('v45')
ax[1, 1].plot(V[:, 59])
ax[1, 1].set_title('v60')
ax[1, 2].plot(V[:, 74])
```

```
ax[1, 2].set_title('v75')
```



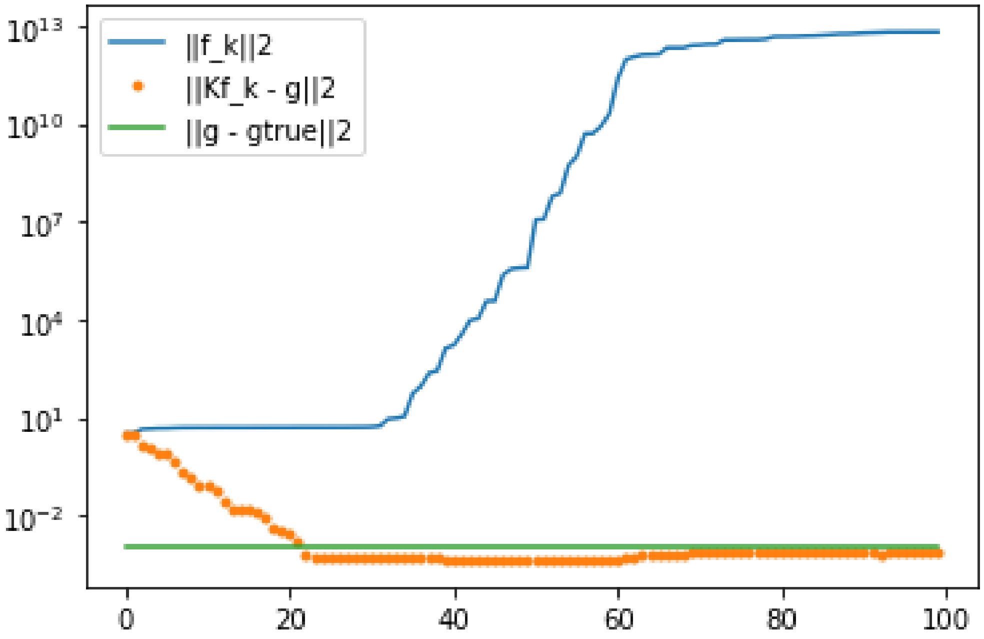
```
[3]: Text(0.5, 1.0, 'v75')
```

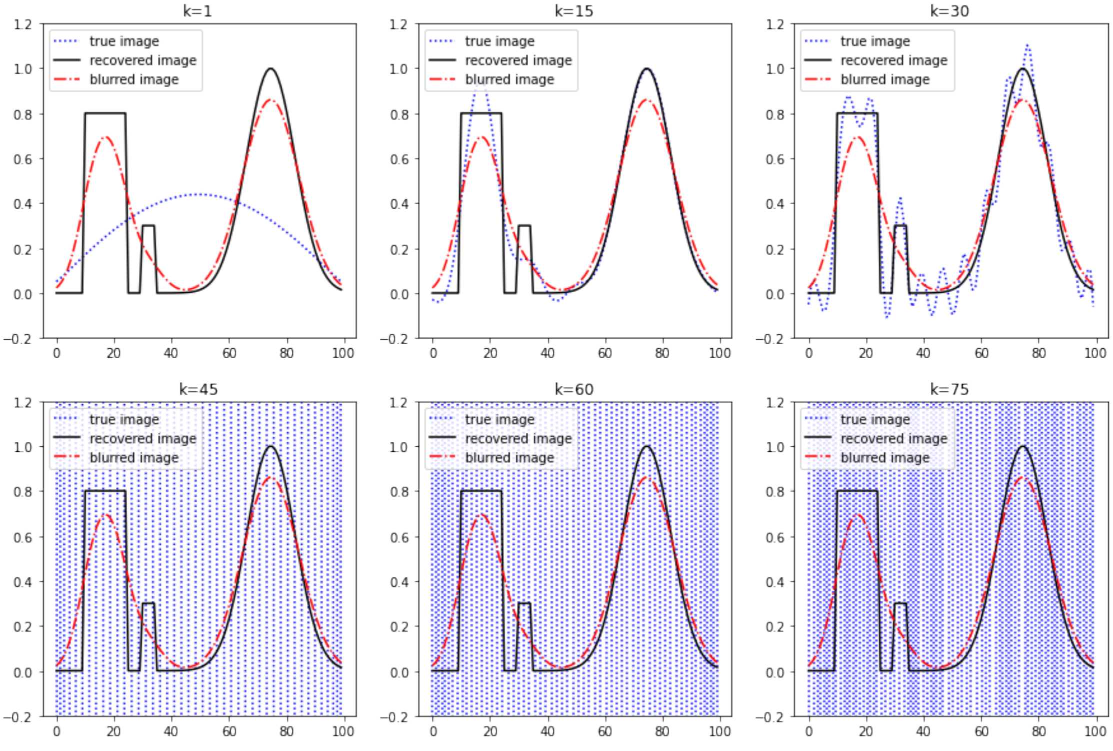


part ii

[9]:

```
x = [i for i in range(g.size)]
fk = np.zeros(g.size)
y1 = np.zeros(g.size)
y2 = np.zeros(g.size)
y3 = np.linalg.norm(g - gtrue, 2) * np.ones(g.size)
for k in range(g.size):
    fk = fk + np.inner(U[:, k], g) / Sigma[k] * V[:, k]
    y1[k] = np.linalg.norm(fk, 2)
    y2[k] = np.linalg.norm(K @ fk - g, 2)
plt.semilogy(x, y1)
plt.semilogy(x, y2, '.')
plt.semilogy(x, y3)
plt.legend(['||f_k||2', '||Kf_k - g||2', '||g - gtrue||2'])
plt.show()

fig, ax = plt.subplots(2, 3, figsize=(15,10))
fvec = [1, 15, 30, 45, 60, 75]

for k in range(len(fvec)):
    fk = np.zeros(g.size)
    for i in range(fvec[k]):
        fk = fk + np.inner(U[:, i], g) / Sigma[i] * V[:, i]
    ax[int(k/3), k%3].plot(x, fk, ':b')
    ax[int(k/3), k%3].plot(x, ftrue, '-k')
    ax[int(k/3), k%3].plot(x, gtrue, '-.r')
    ax[int(k/3), k%3].legend(['true image','recovered image','blurred image'])
    ax[int(k/3), k%3].set_ylim(-0.2, 1.2)
    ax[int(k/3), k%3].set_title('k=' + str(fvec[k]))
```


||f_k||2
||Kf_k - g||2
||g - gtrue||2
10^{13}
10^{10}
10^{7}
10^{4}
10^{1}
10^{-2}
0
20
40
60
80
100


k=1
k=15
k=30
k=45
k=60
k=75
true image
recovered image
blurred image
−0.2
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
20
40
60
80
100

part iii

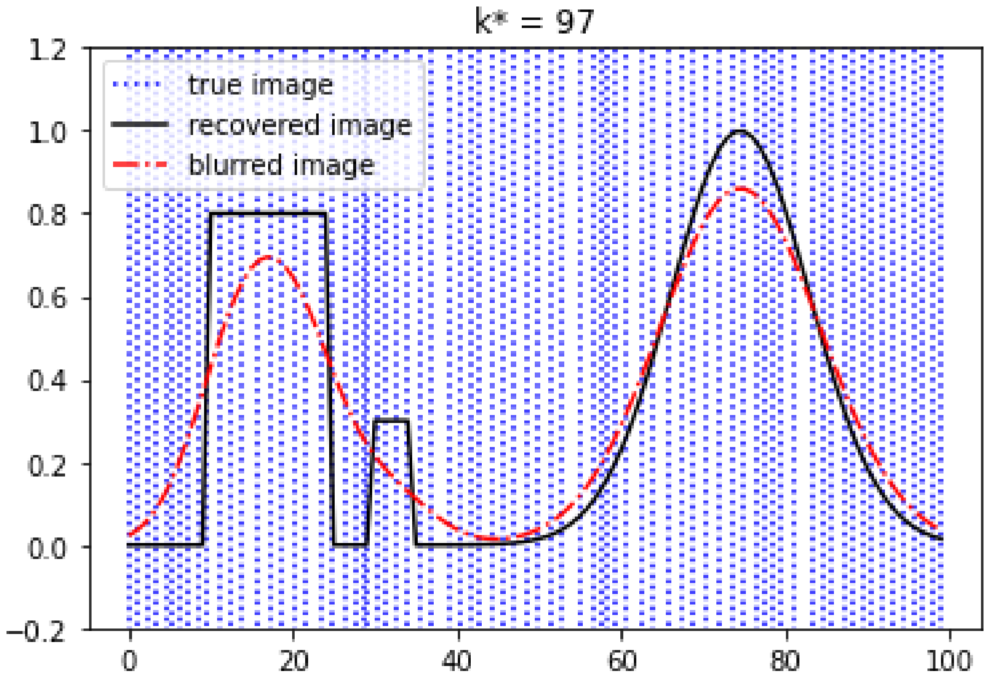
[44]:

```
fk = np.zeros(g.size)
result = np.zeros(g.size)
for i in range(g.size):
    fk = fk + np.inner(U[:, i], g) / Sigma[i] * V[:, i]
    result[i] = abs(np.linalg.norm(K @ fk - g, 2) - np.linalg.norm(gerr, 2))
kstar = np.argmin(result)
print('K* = ' + str(kstar))
fk = np.zeros(g.size)
for i in range(kstar):
    fk = fk + np.inner(U[:, i], g) / Sigma[i] * V[:, i]
fig, ax = plt.subplots()
ax.plot(x, fk, ':b')
ax.plot(x, ftrue, '-k')
ax.plot(x, gtrue, '-.r')
ax.legend(['true image','recovered image','blurred image'])
ax.set_ylim(-0.2, 1.2)
ax.set_title('k* = ' + str(kstar))
```

```
K* = 97
```

[44]:

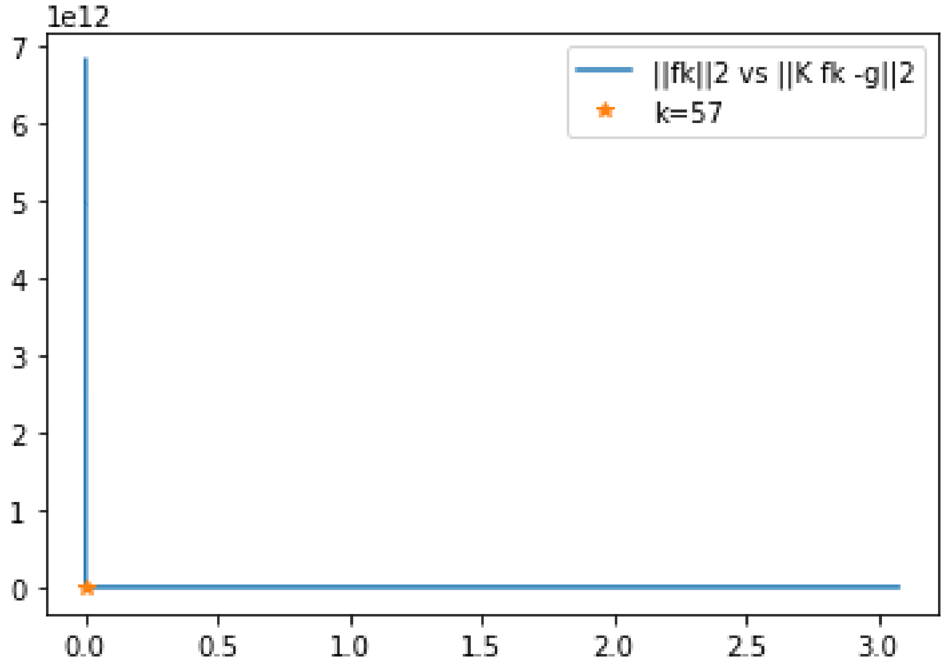
```
Text(0.5, 1.0, 'k* = 97')
```



part iv

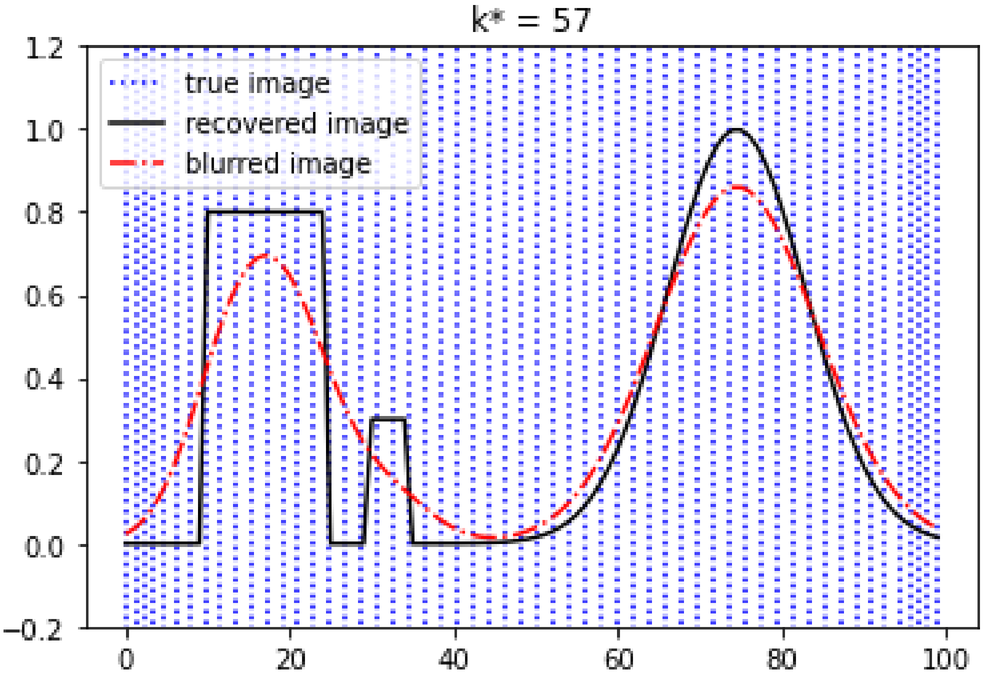
[45]:

```
plt.plot(y2, y1)
k = 57
plt.plot(y2[k], y1[k], '*')
plt.legend(['||fk||2 vs ||K fk -g||2', 'k=57'])
plt.show()

kstar = 57
fk = np.zeros(g.size)
for i in range(kstar):
    fk = fk + np.inner(U[:, i], g) / Sigma[i] * V[:, i]
fig, ax = plt.subplots()
ax.plot(x, fk, ':b')
ax.plot(x, ftrue, '-k')
ax.plot(x, gtrue, '-.r')
ax.legend(['true image','recovered image','blurred image'])
ax.set_ylim(-0.2, 1.2)
ax.set_title('k* = ' + str(kstar))
```



[45]:

```
Text(0.5, 1.0, 'k* = 57')
```


k* = 57
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0
20
40
60
80
100
true image
recovered image
blurred image