KhuatDangSon_20002159_Lab4

October 10, 2023

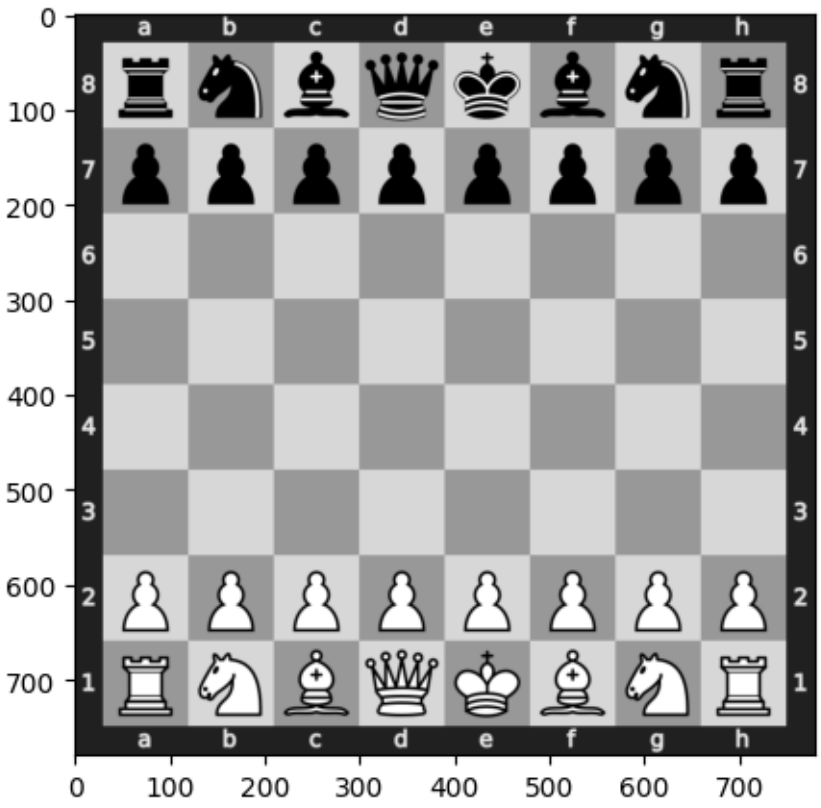
1 Import Library

[137]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

2 Low Pass Frequency by using Distance smooth function

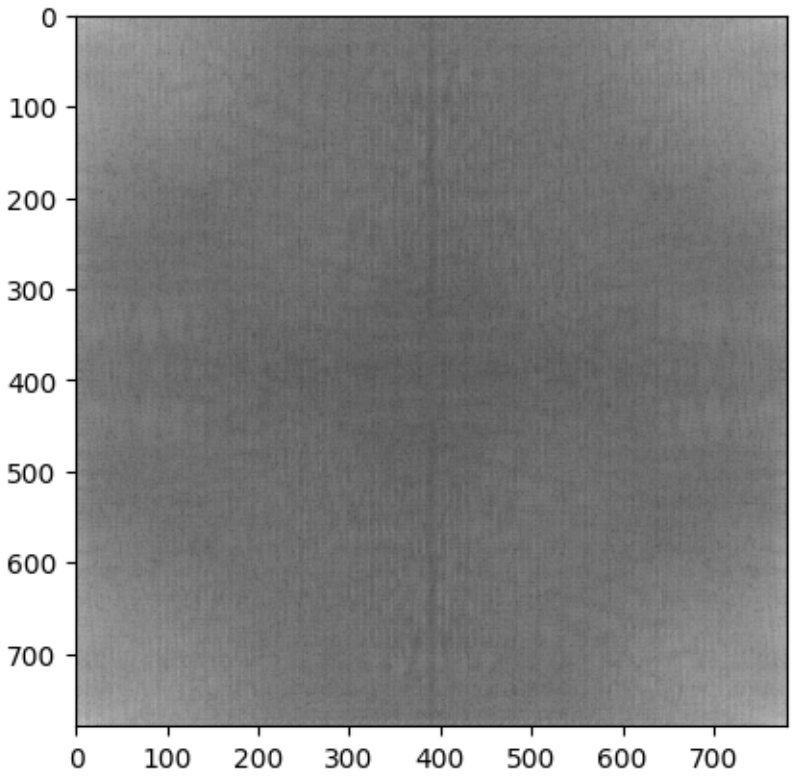
2.1 Read Image and Convert Image to Gray

[138]:
```
img = cv2.imread('chess_board.png')
# Convert Image to gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(5, 5))
plt.imshow(gray, cmap='gray')
plt.show()
```



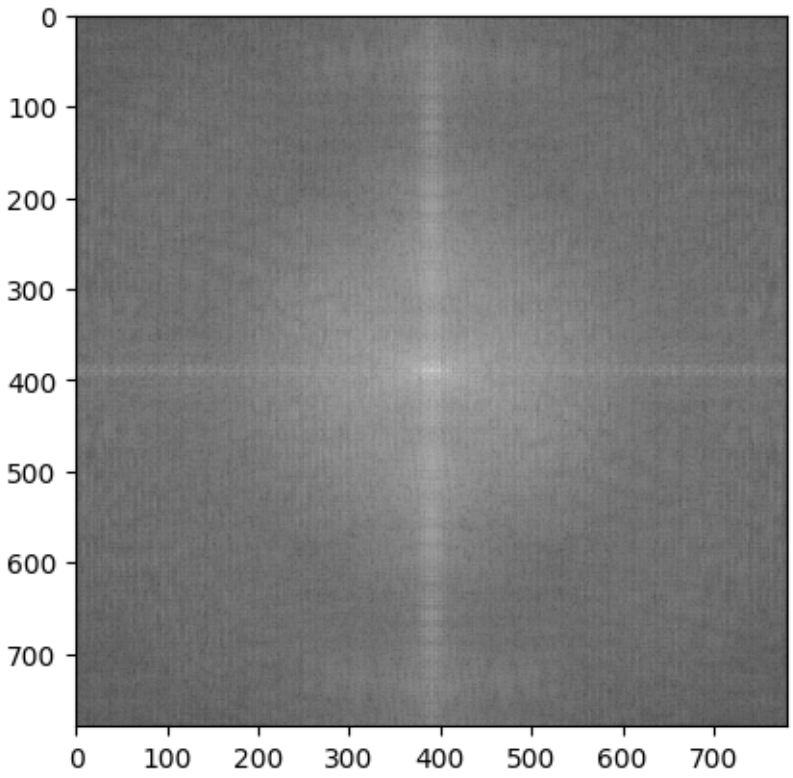
2.2 Fourier Transformation

[139]:
```
# Fourier transform mation
f = np.fft.fft2(gray)
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```



2.3 Frequency Shift

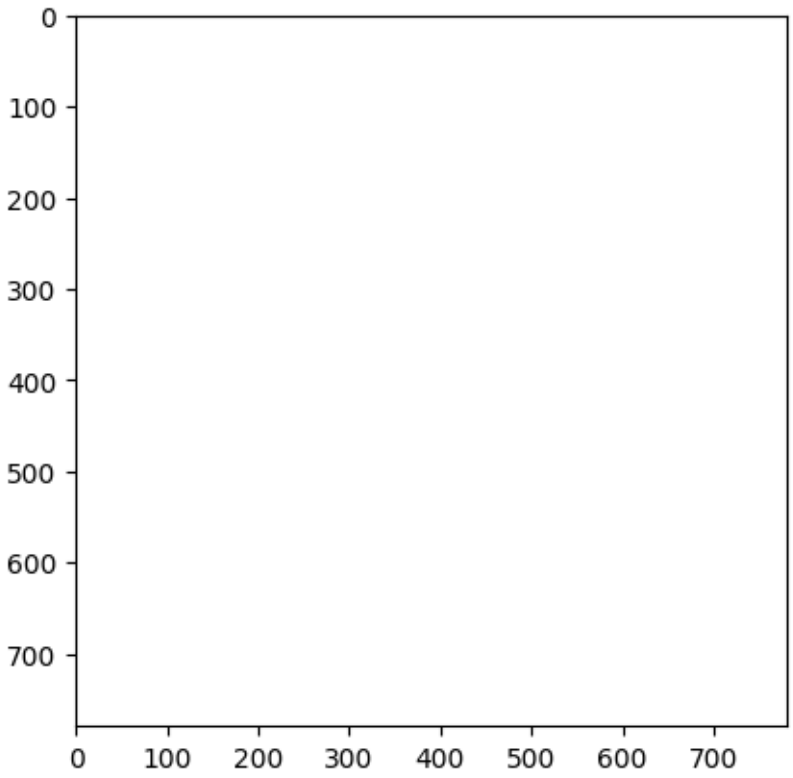
[140]:
```
shifted_f = np.fft.fftshift(f)
plt.imshow(np.log(np.abs(shifted_f)), cmap='gray')
plt.show()
```



2.4 Low Pass Frequency

[141]:
```
low_pass = np.ones(shape=gray.shape)
plt.imshow(low_pass, cmap='gray', vmax=1, vmin=0)
```

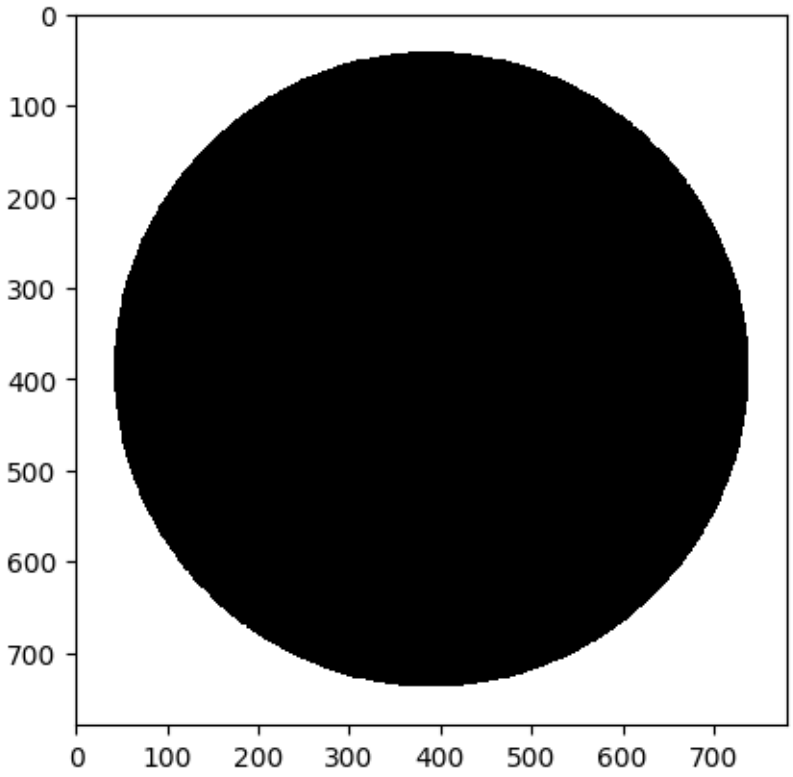
[141]: <matplotlib.image.AxesImage at 0x7bf7d0cd8f40>



[142]:
```
# Lay tam hinh tron
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2

# Ban kinh (Do)
Do = 350

# Tinh toan khoang cach tu 1 diem den tam voi cong thuc: np.sqrt((x-xc)**2 +␣
↪(y-yc)**2)
def d(x, y):
  return np.sqrt((x-xc)**2 + (y-yc)**2)

for x in range(gray.shape[1]):
  for y in range(gray.shape[0]):
    if d(x, y) > Do:
      low_pass[y,x] = 0

f = f * low_pass
plt.imshow(np.log(np.abs(low_pass)), cmap='gray')
plt.show()
```

```
<ipython-input-142-822910c362a5>:18: RuntimeWarning: divide by zero encountered
```

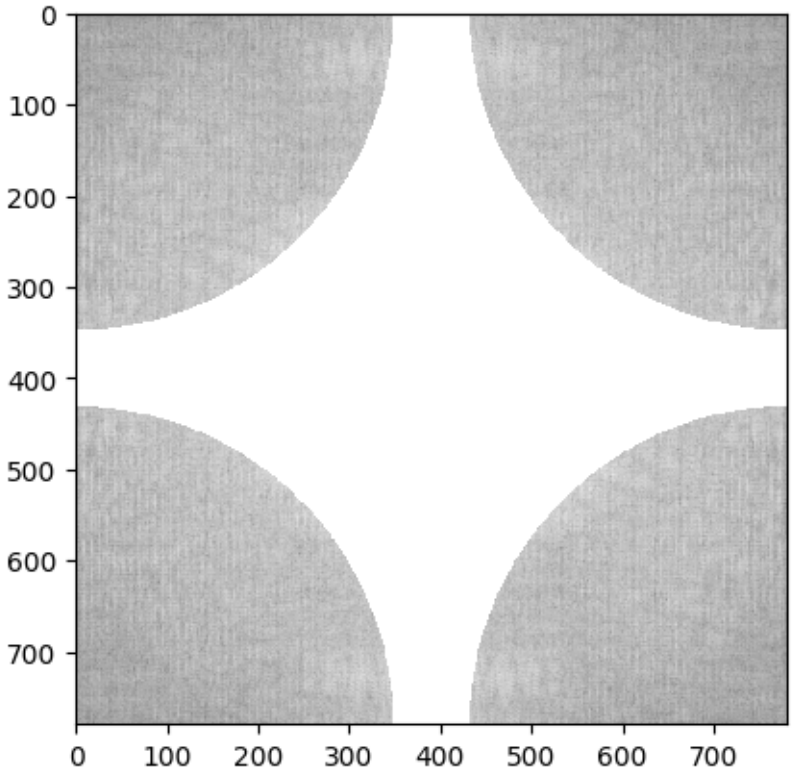
```
in log
  plt.imshow(np.log(np.abs(low_pass)), cmap='gray')
```



2.5 Invert Shift

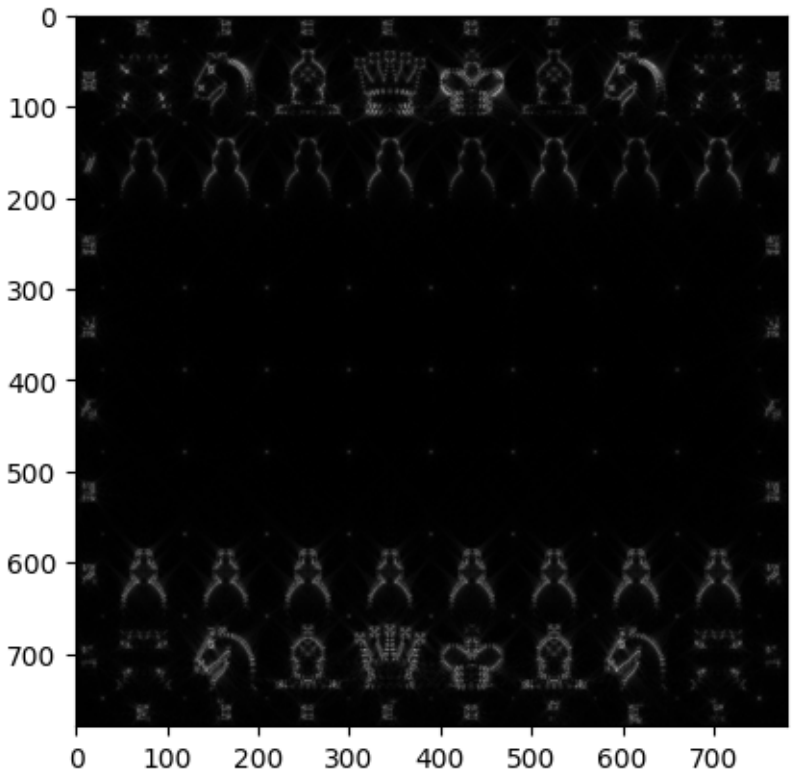
[143]:
```
f = np.fft.ifftshift(f)
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```

```
<ipython-input-143-cb49ad914418>:2: RuntimeWarning: divide by zero encountered
in log
  plt.imshow(np.log(np.abs(f)), cmap='gray')
```



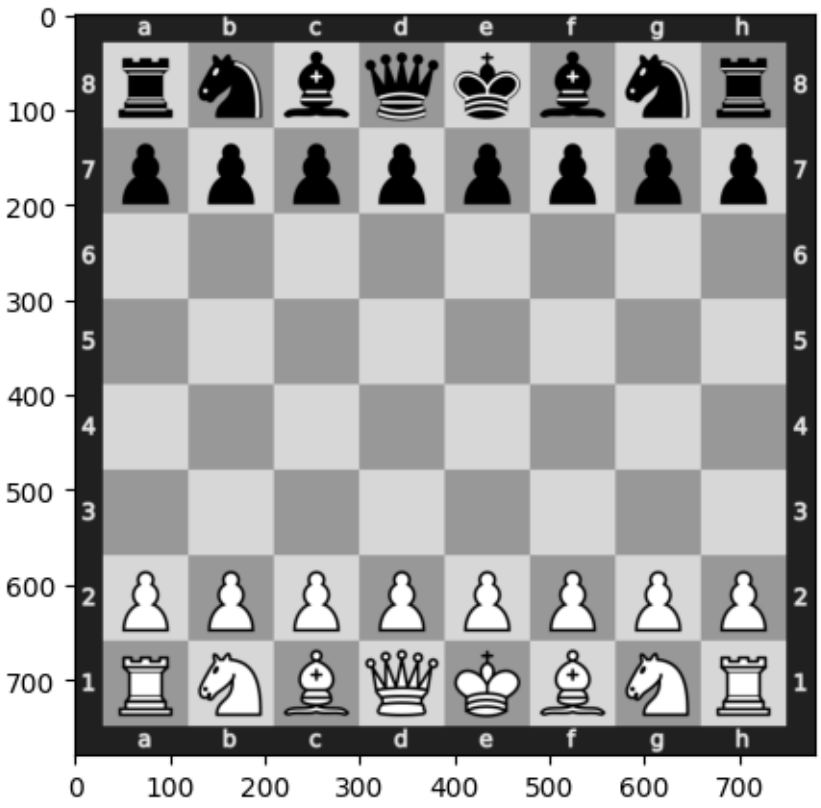
2.6 Invert Fourier Transform

[144]:
```
new_img = np.abs(np.fft.ifft2(f))
plt.imshow(new_img, cmap='gray')
plt.show()
```

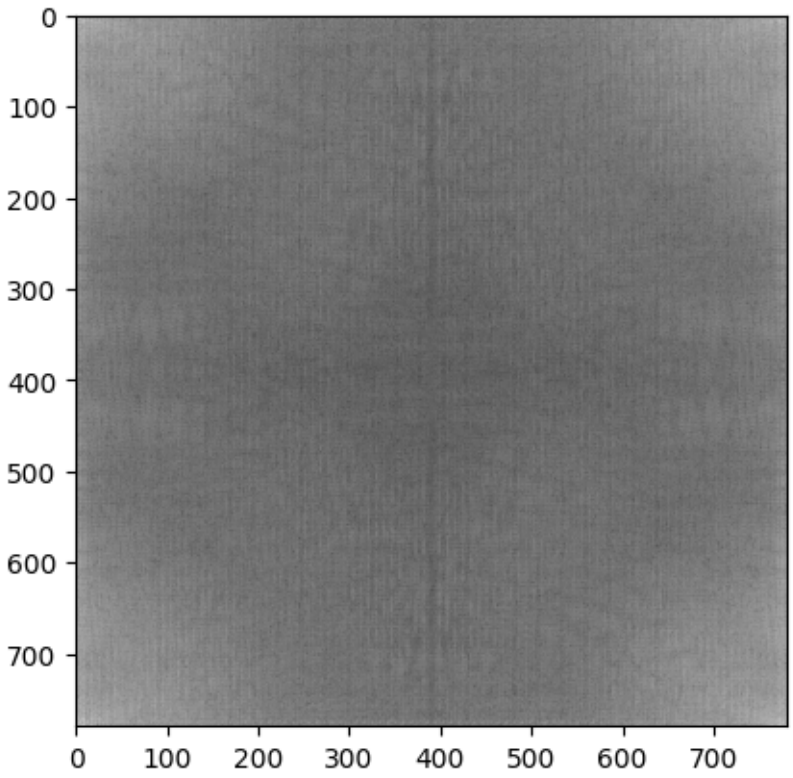


3 High Pass Frequency by using Distance smooth function

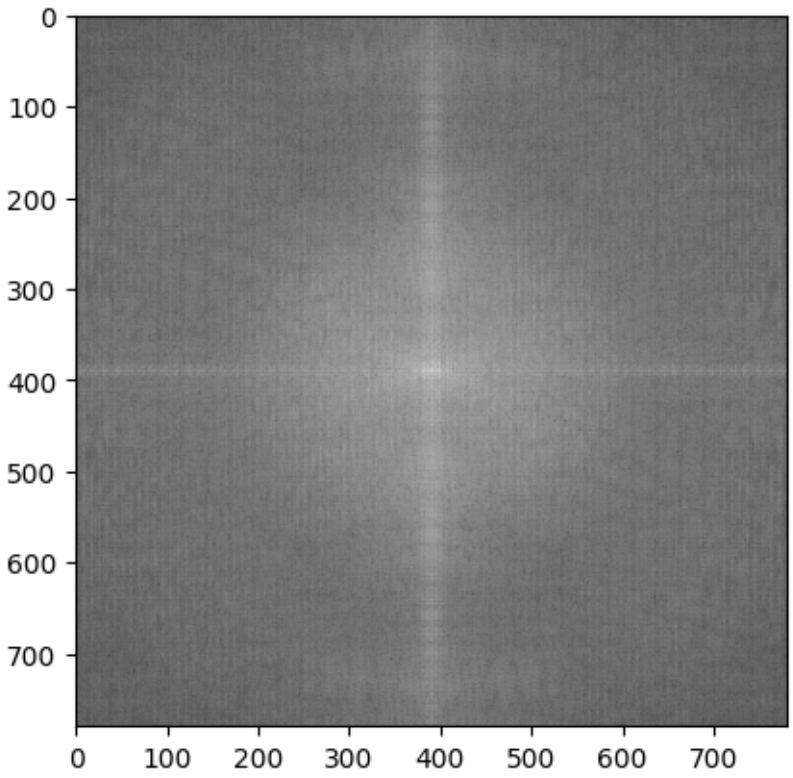
[145]:
```
img = cv2.imread('chess_board.png')
# Convert Image to gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.figure(figsize=(5, 5))
plt.imshow(gray, cmap='gray')
plt.show()
```



[146]:
```
# Fourier transform mation
f = np.fft.fft2(gray)
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```

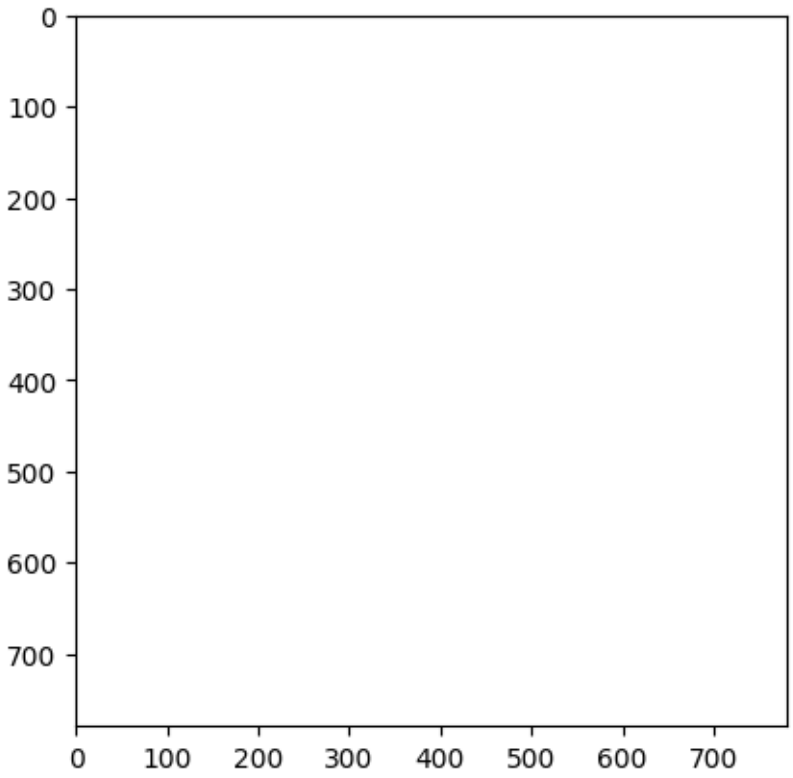


[147]:
```
shifted_f = np.fft.fftshift(f)
plt.imshow(np.log(np.abs(shifted_f)), cmap='gray')
plt.show()
```



[148]:
```
low_pass = np.ones(shape=gray.shape)
plt.imshow(low_pass, cmap='gray', vmax=1, vmin=0)
```

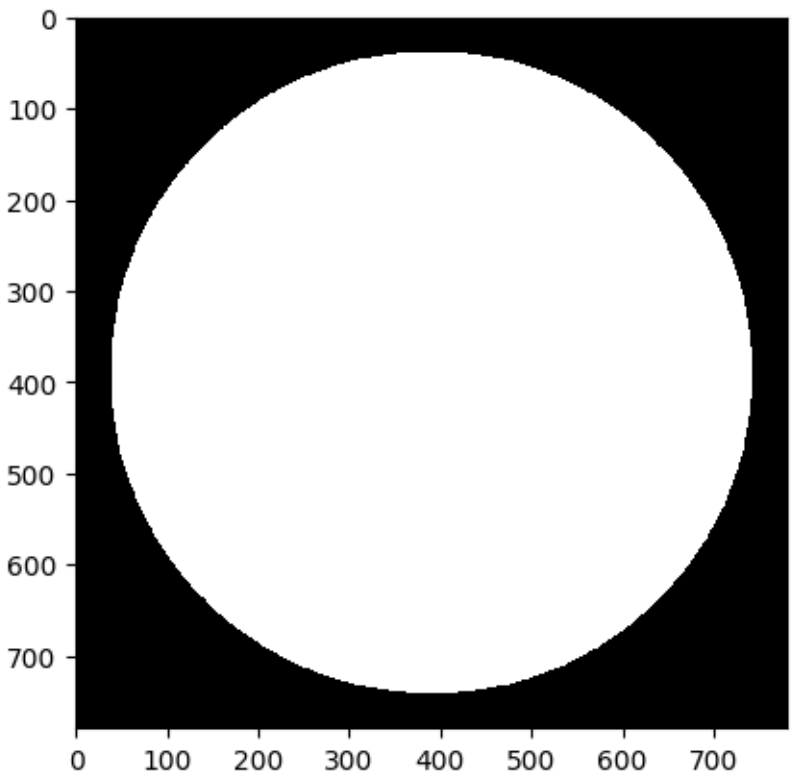
[148]: <matplotlib.image.AxesImage at 0x7bf82521dbd0>



[149]:
```
# Lay tam hinh tron
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2

# Ban kinh (Do)
Do = 350

# Tinh toan khoang cach tu 1 diem den tam voi cong thuc: np.sqrt((x-xc)**2 +␣
↪(y-yc)**2)
def d(x, y):
  return np.sqrt((x-xc)**2 + (y-yc)**2)

for x in range(gray.shape[1]):
  for y in range(gray.shape[0]):
    if d(x, y) < Do:
      low_pass[y,x] = 0

f = f * low_pass
plt.imshow(np.log(np.abs(low_pass)), cmap='gray')
plt.show()
```

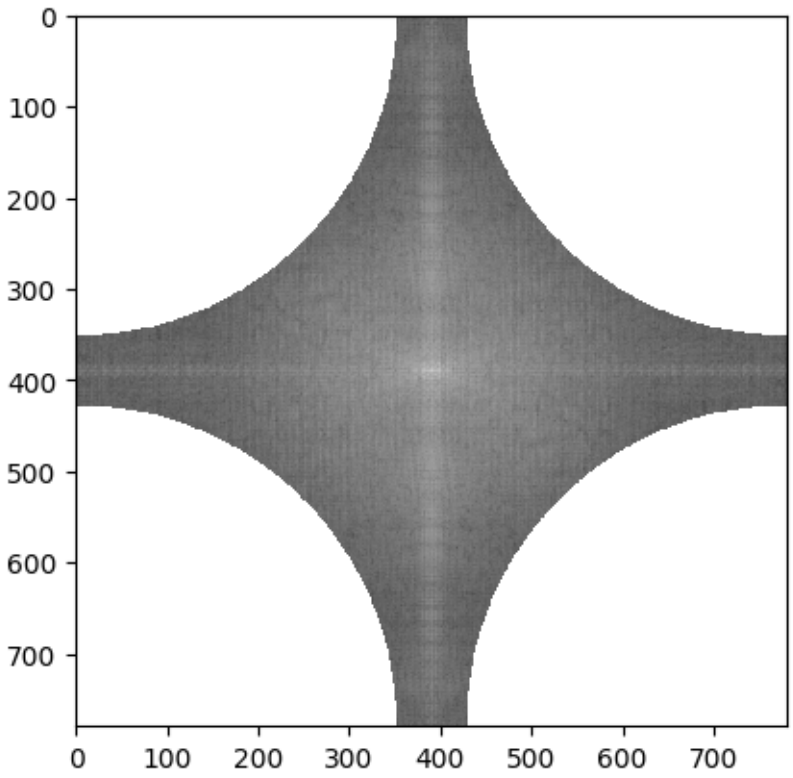
```
<ipython-input-149-4cc656998629>:18: RuntimeWarning: divide by zero encountered
```

```
in log
  plt.imshow(np.log(np.abs(low_pass)), cmap='gray')
```

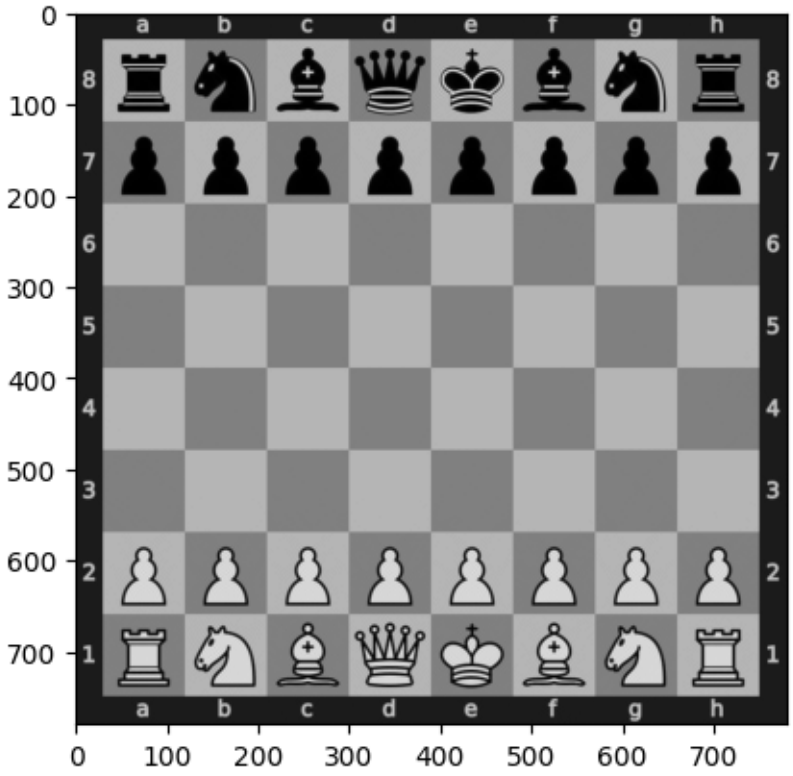


[150]:
```
f = np.fft.ifftshift(f)
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```

```
<ipython-input-150-cb49ad914418>:2: RuntimeWarning: divide by zero encountered
in log
  plt.imshow(np.log(np.abs(f)), cmap='gray')
```



[151]:
```
new_img = np.abs(np.fft.ifft2(f))
plt.imshow(new_img, cmap='gray')
plt.show()
```

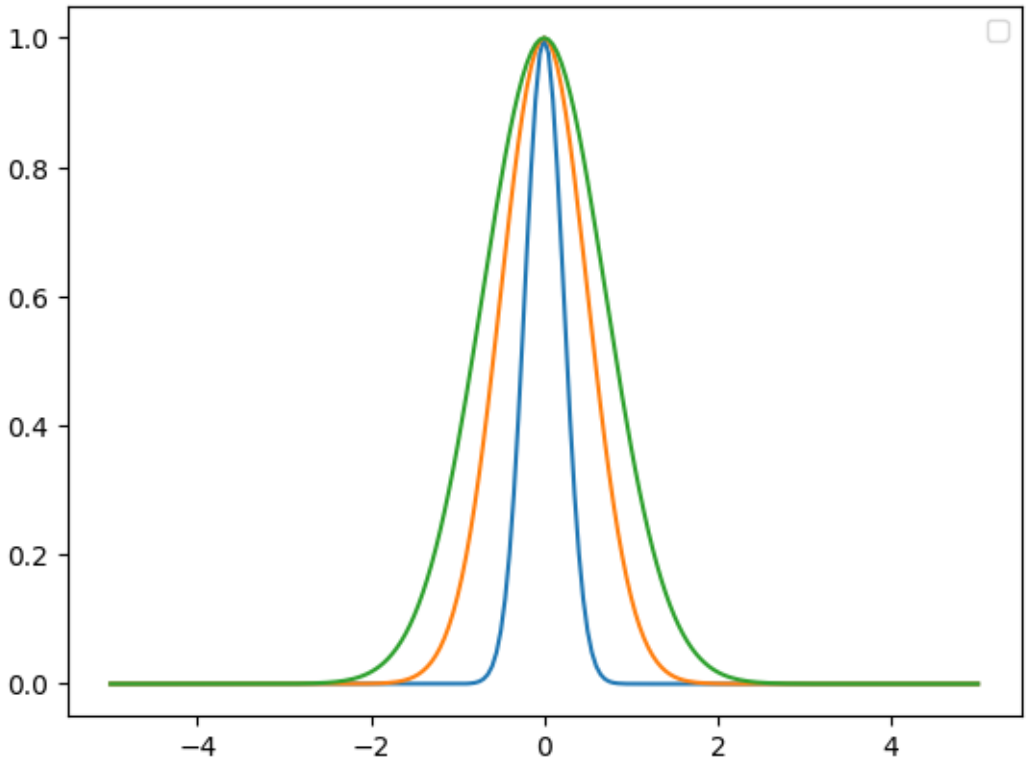


4 Fourier Transform

[152]:
```
for sigma in [0.1, 0.5, 1.0]:
  x = np.linspace(-5, 5, 201)
  y = np.exp(-x**2 / sigma)
  plt.plot(x, y, label = "".format(sigma))

plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
```

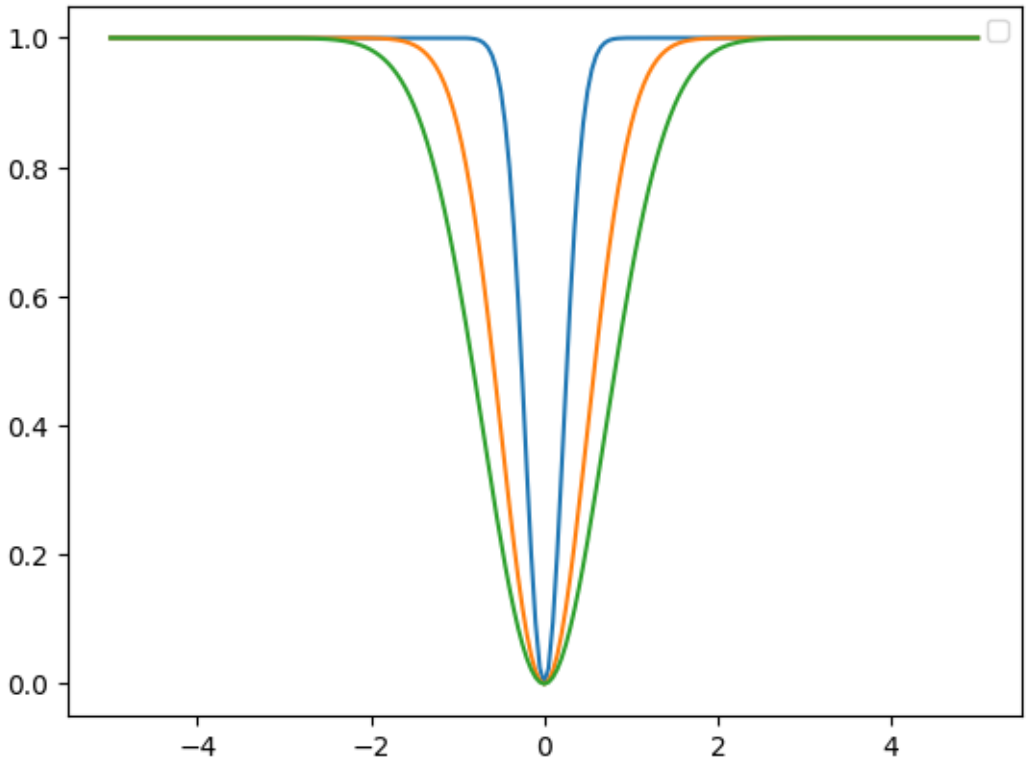
[152]: <matplotlib.legend.Legend at 0x7bf7d1185450>



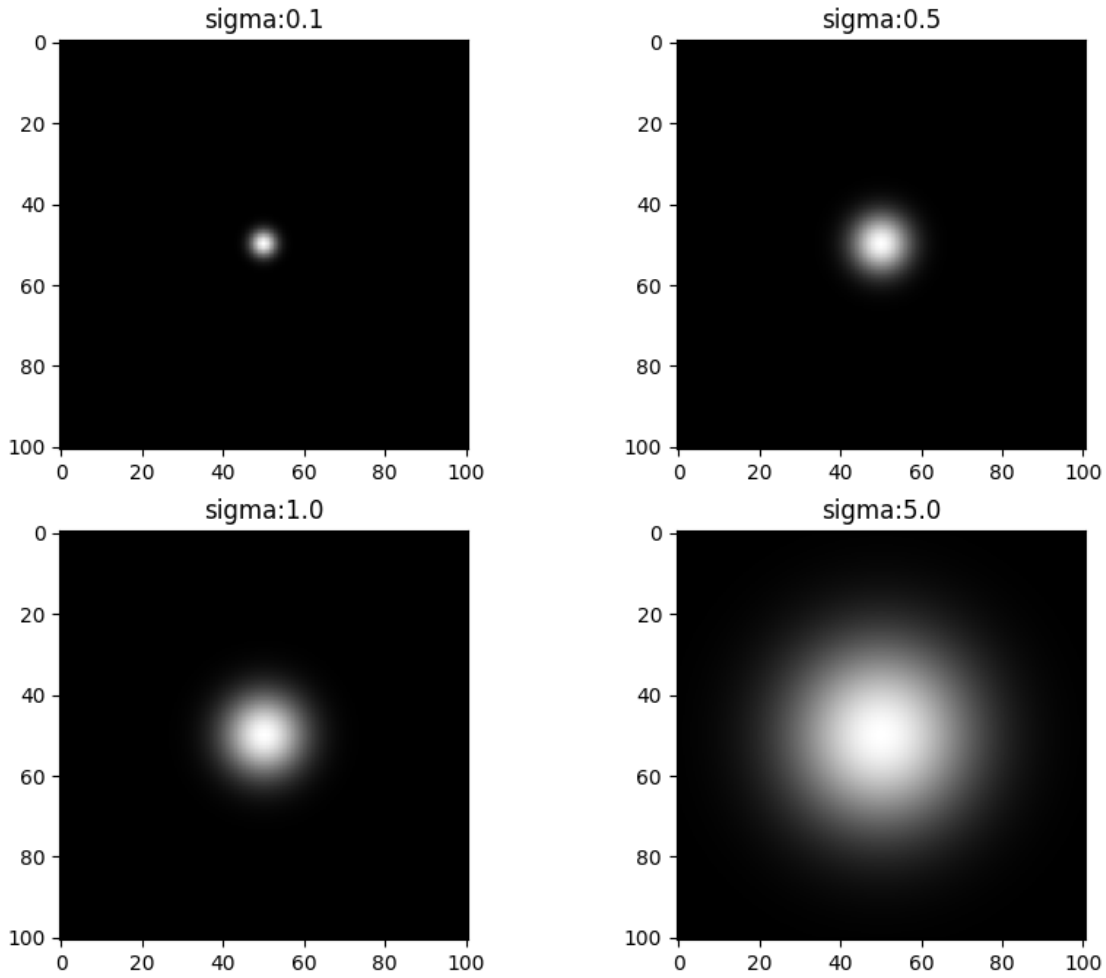
[153]:
```
for sigma in [0.1, 0.5, 1.0]:
  x = np.linspace(-5, 5, 201)
  y = 1 - np.exp(-x**2 / sigma)
  plt.plot(x, y, label = "".format(sigma))

plt.legend()
```

```
WARNING:matplotlib.legend:No artists with labels found to put in legend.  Note
that artists whose label start with an underscore are ignored when legend() is
called with no argument.
```

[153]:
```
<matplotlib.legend.Legend at 0x7bf7dba383d0>
```



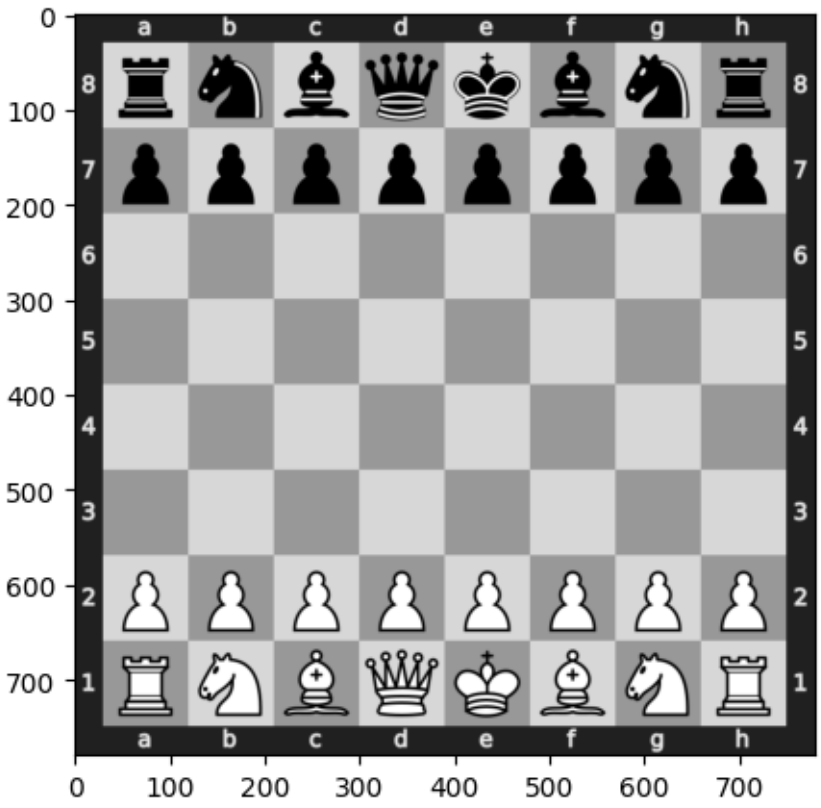
[154]:
```
# plot 3D function y = exp(-(x**2 + y**2) / sigma)
x = np.linspace(-5, 5, 101)
y = np.linspace(-5, 5, 101)

X, Y = np.meshgrid(x, y)

plt.figure(figsize=(10,8))
for i, sigma in enumerate([0.1, 0.5, 1.0, 5.0]):
  z = np.exp(-(X**2 + Y**2) / sigma)
  plt.subplot(2, 2, i+1)
  plt.imshow(z, cmap='gray')
  plt.title(f"sigma:{sigma}")

plt.show()
```



5 Low pass frequency by using Gaussian smooth function

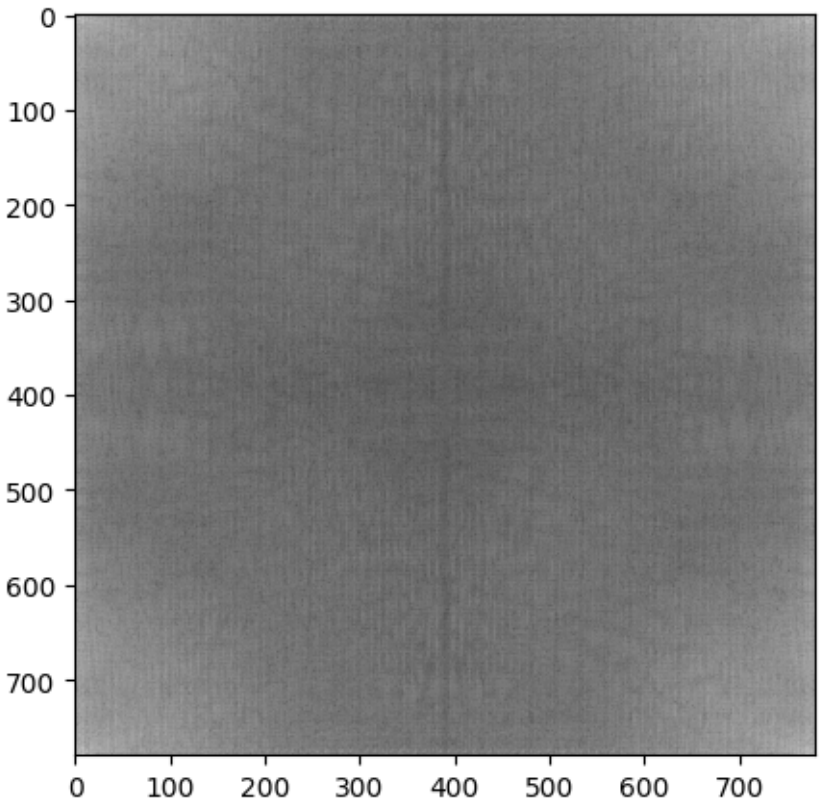
5.1 Read Image and Convert to gray image

[155]:
```
img = cv2.imread('chess_board.png')
# Convert to gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Show image
plt.figure(figsize=(5, 5))
plt.imshow(gray, cmap='gray')
plt.show()
```



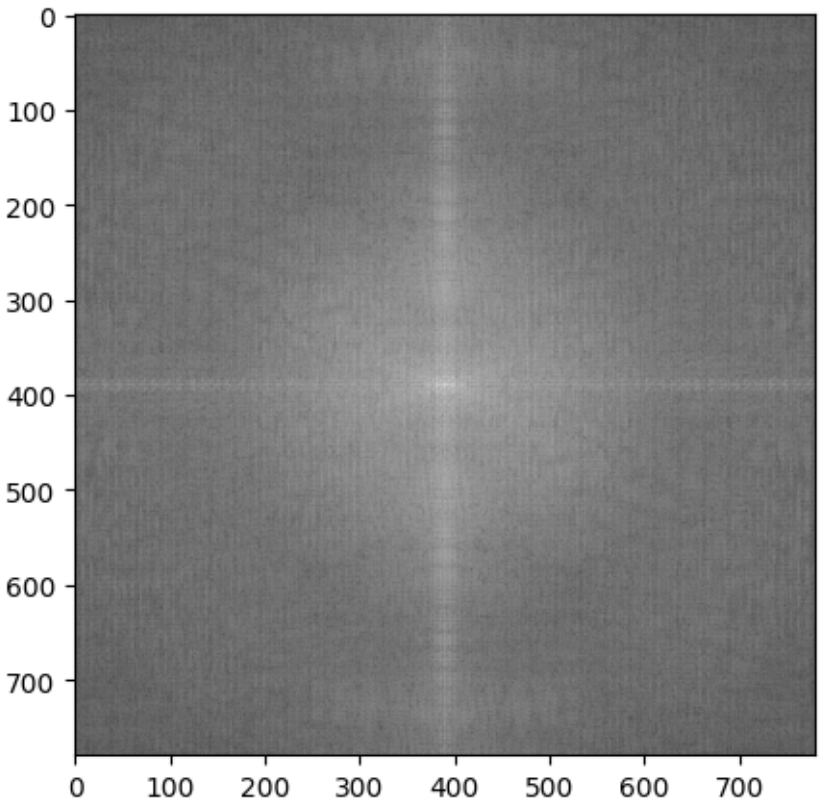
5.2 Fourier Transformation

[156]:
```
f = np.fft.fft2(gray)
plt.figure(figsize=(5, 5))
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```



5.3 Frequency Shift

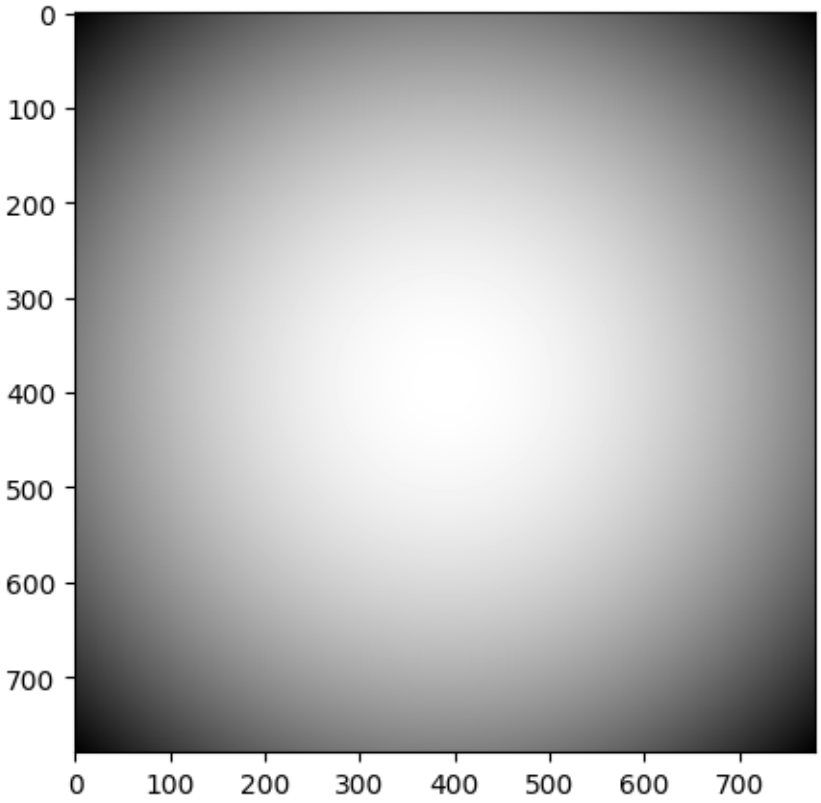
[157]:
```
shifted_f = np.fft.fftshift(f)
plt.figure(figsize=(5,5))
plt.imshow(np.log(np.abs(shifted_f)), cmap='gray')
plt.show()
```



5.4 Low pass frequency

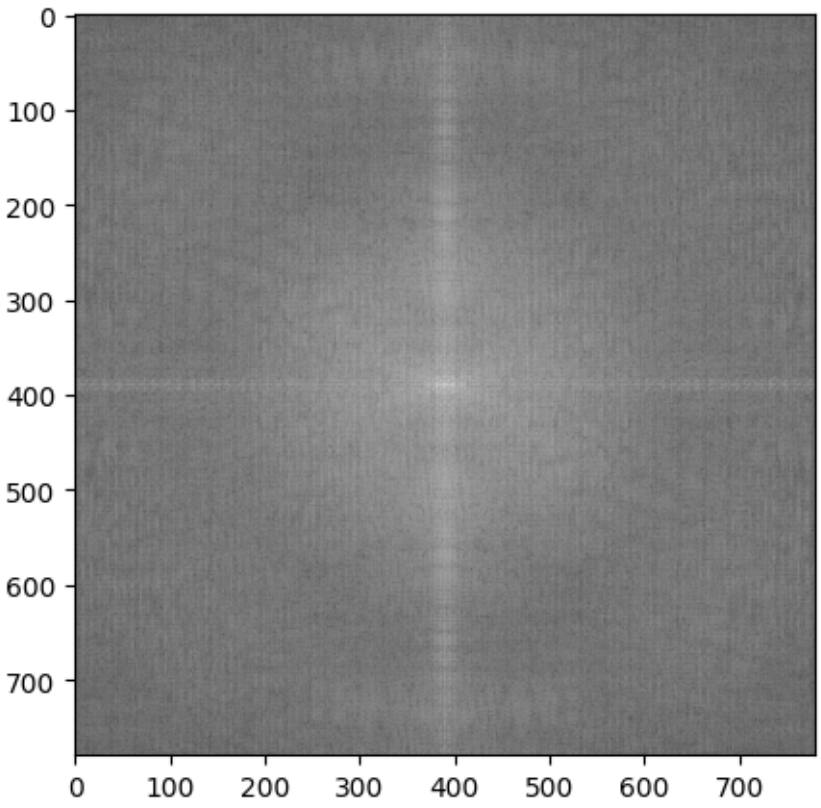
[158]:
```
low_pass = np.ones(shape=gray.shape)

# Lay tam buc anh
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2

# ban kinh
Do = 350

# Tinh khoang cach tu 1 diem den tam
def d(x, y):
  return np.sqrt((x-xc)**2 + (y-yc)**2)

for x in range(gray.shape[1]):
  for y in range(gray.shape[0]):
    low_pass[y,x] = np.exp(-d(x,y)**2 / (2*Do**2))

f = f * low_pass
```

```
plt.figure(figsize=(5,5))
plt.imshow(np.log(np.abs(low_pass)), cmap='gray')
plt.show()
```



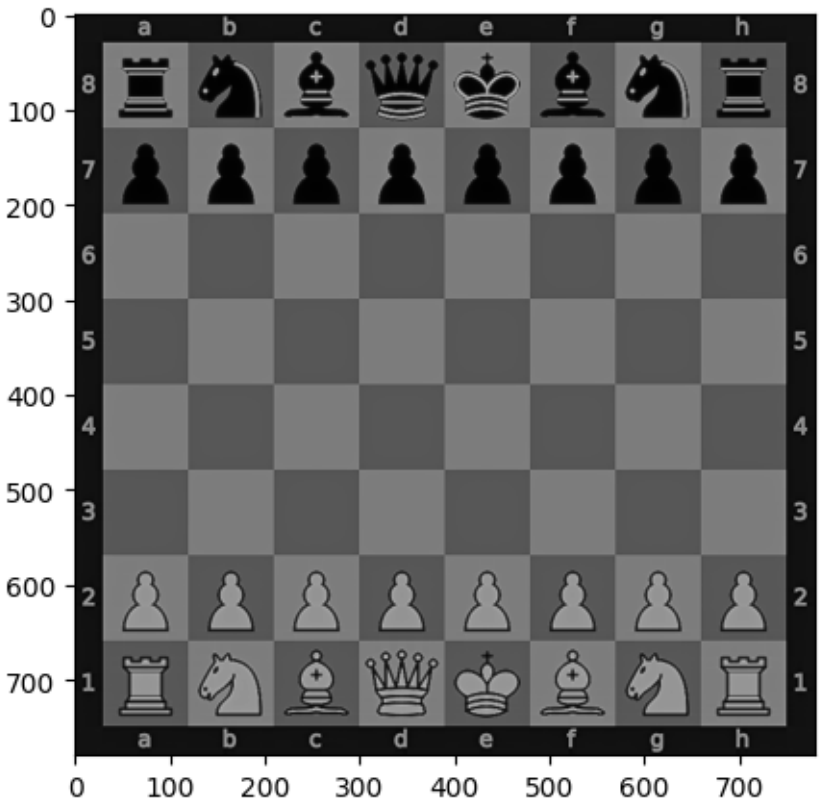
5.5 Invert Shift

[159]:
```
f = np.fft.ifftshift(f)
plt.figure(figsize=(5,5))
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```



5.6 Invert Fourier Transform

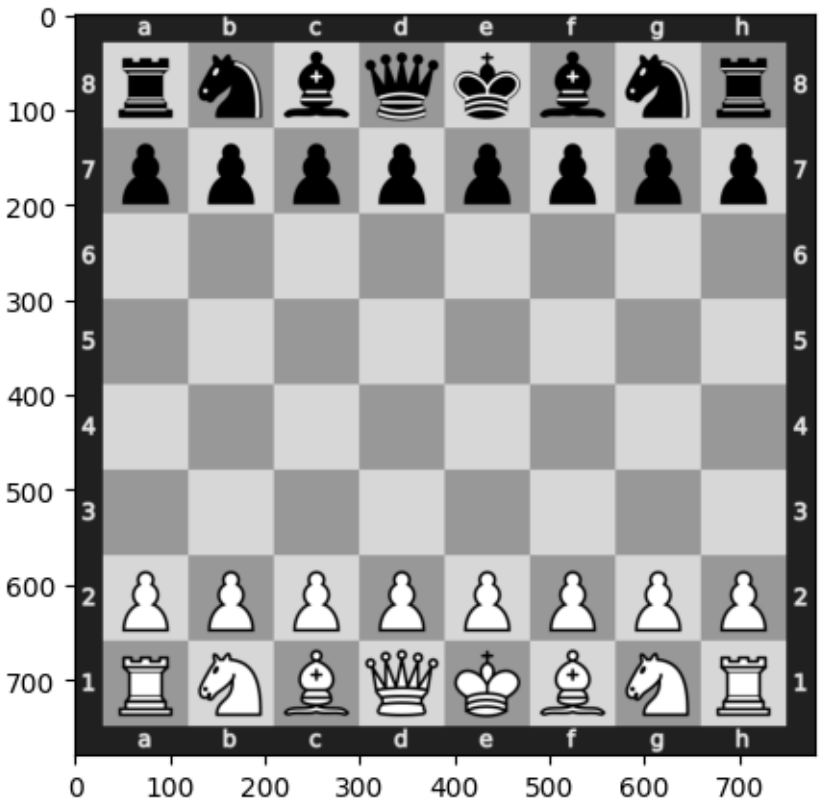
[160]:
```
new_img = np.abs(np.fft.ifft2(f))
plt.figure(figsize=(5,5))
plt.imshow(new_img, cmap='gray')
plt.show()
```



6 High pass frequency by using Gaussian smooth function

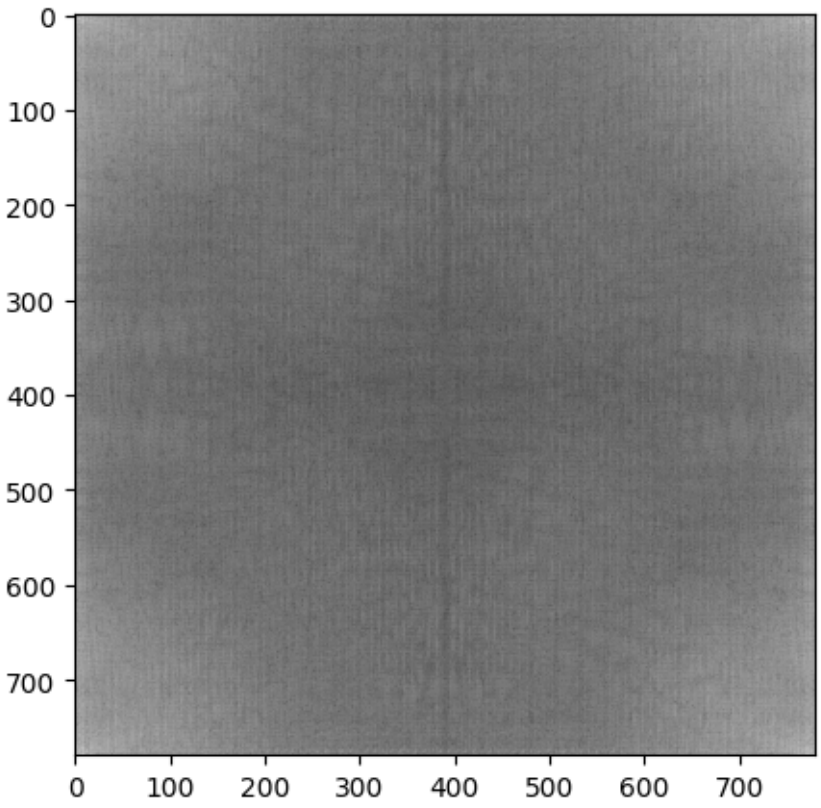
6.1 Read Image and Convert to gray image

[161]:
```
img = cv2.imread('chess_board.png')
# Convert to gray
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Show image
plt.figure(figsize=(5, 5))
plt.imshow(gray, cmap='gray')
plt.show()
```



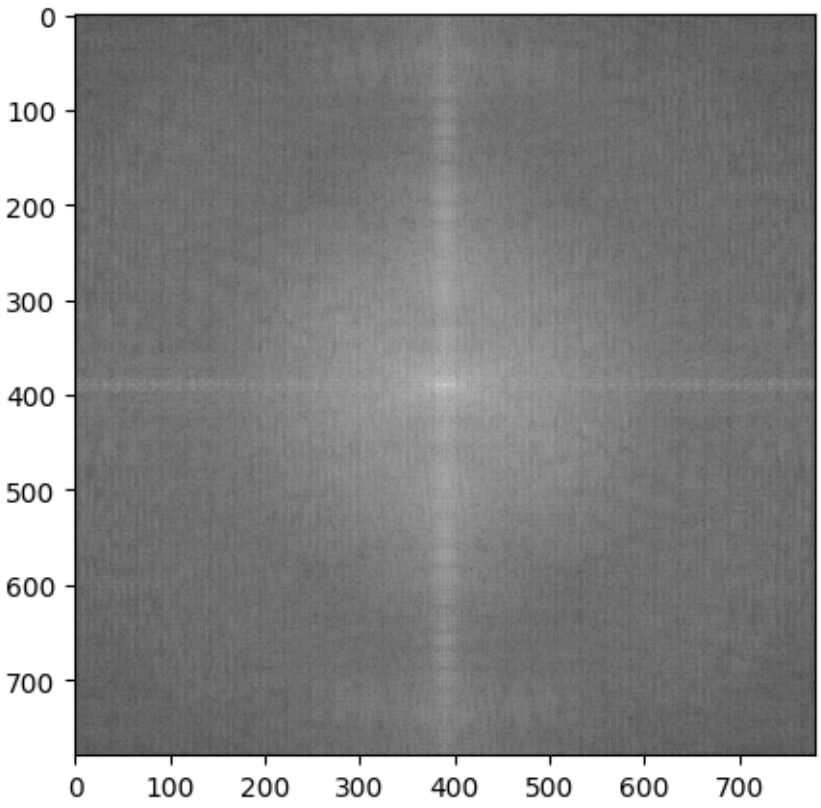
6.2 Fourier Transformation

[162]:
```
f = np.fft.fft2(gray)
plt.figure(figsize=(5, 5))
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```



6.3 Frequency shift

[163]:
```
shifted_f = np.fft.fftshift(f)
plt.figure(figsize=(5,5))
plt.imshow(np.log(np.abs(shifted_f)), cmap='gray')
plt.show()
```

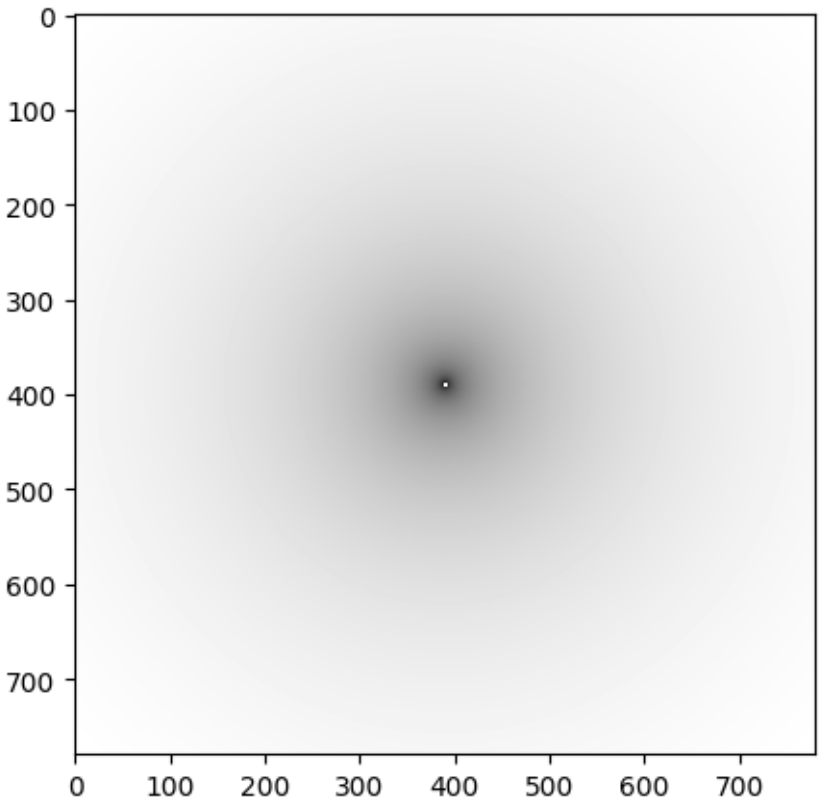


6.4 High pass frequency

[164]:
```
high_pass = np.ones(shape=gray.shape)

# Lay tam buc anh
xc = gray.shape[1] // 2
yc = gray.shape[0] // 2

# ban kinh
Do = 350

# Tinh khoang cach tu 1 diem den tam
def d(x, y):
  return np.sqrt((x-xc)**2 + (y-yc)**2)

for x in range(gray.shape[1]):
  for y in range(gray.shape[0]):
    high_pass[y,x] = 1 - np.exp(-d(x,y)**2 / (2*Do**2))

f = f * high_pass
```

```
plt.figure(figsize=(5,5))
plt.imshow(np.log(np.abs(high_pass)), cmap='gray')
plt.show()
```

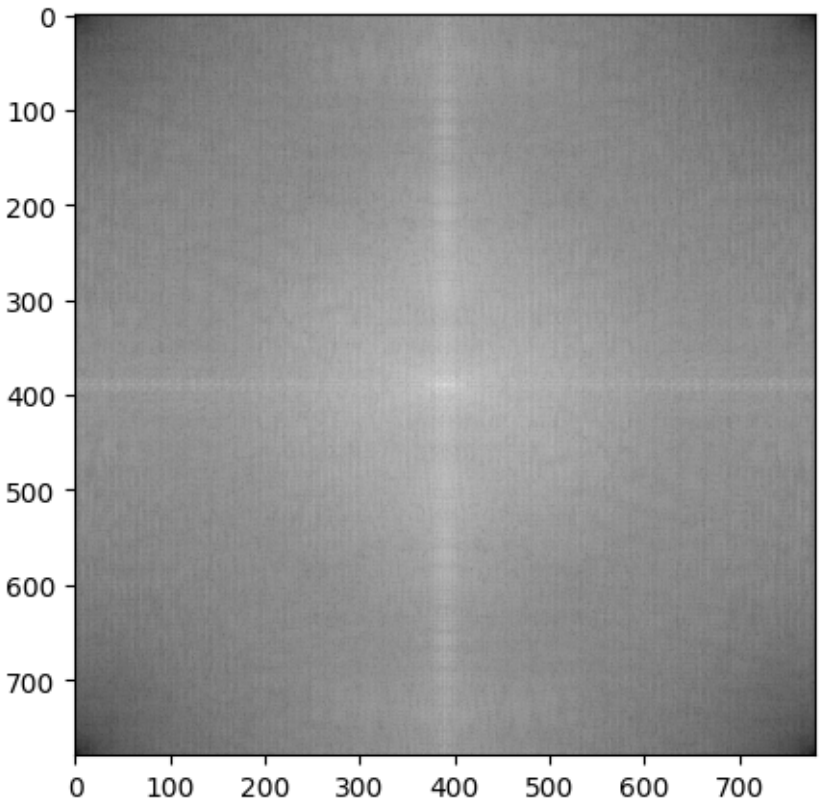
```
<ipython-input-164-f0c381786592>:21: RuntimeWarning: divide by zero encountered
in log
  plt.imshow(np.log(np.abs(high_pass)), cmap='gray')
```



6.5 Invert Shift

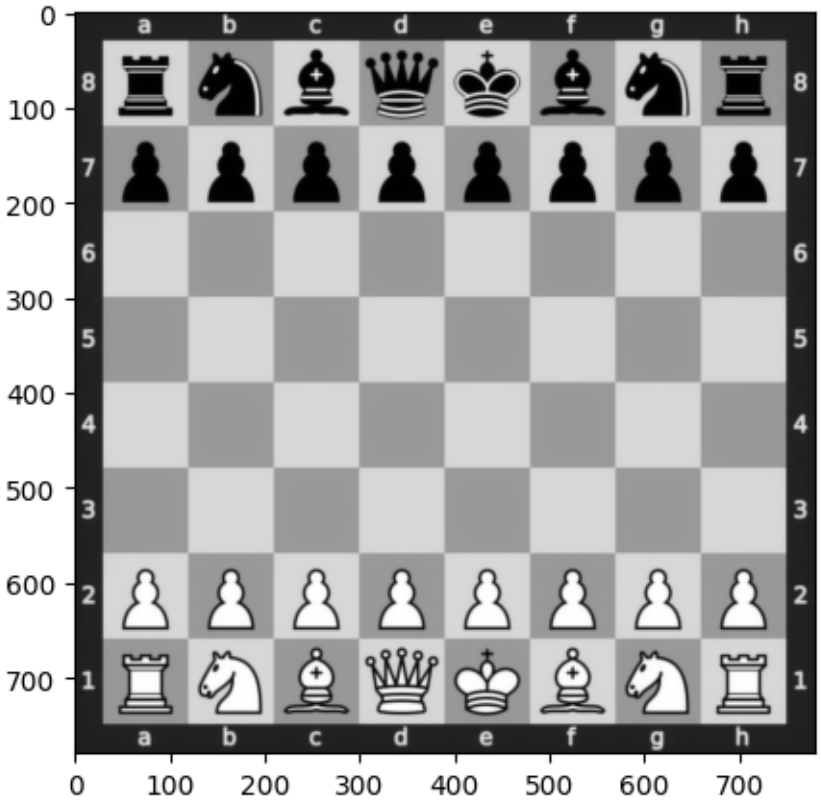
[165]:
```
f = np.fft.ifftshift(f)
plt.figure(figsize=(5,5))
plt.imshow(np.log(np.abs(f)), cmap='gray')
plt.show()
```

```
<ipython-input-165-ed82df64b3ce>:3: RuntimeWarning: divide by zero encountered
in log
  plt.imshow(np.log(np.abs(f)), cmap='gray')
```



6.6 Invert Fourier Transform

[166]:
```
new_img = np.abs(np.fft.ifft2(f))
plt.figure(figsize=(5,5))
plt.imshow(new_img, cmap='gray')
plt.show()
```


a b c d e f g h
8 7 6 5 4 3 2 1
0 100 200 300 400 500 600 700