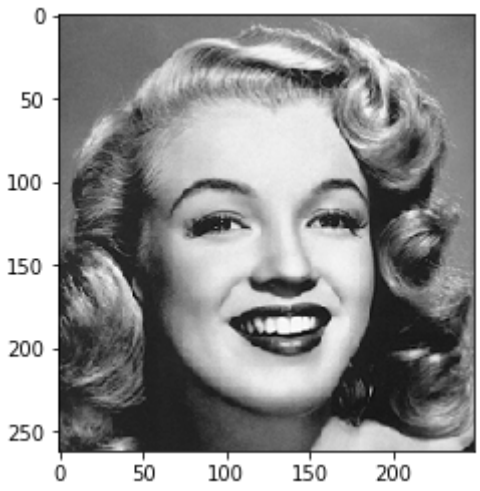
In [90]:
```
from PIL import Image
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = Image.open('/Users/ceciliazhang/Desktop/einstein.png').convert('L'
)
img = np.array(img)
print(img.dtype, img.shape)
img = img.astype(np.float32) / 255.

img2 = Image.open('/Users/ceciliazhang/Desktop/marilyn.png').convert('L'
)
img2 = np.array(img2)
img2 = img2.astype(np.float32) / 255.

plt.imshow(img2, cmap='gray')
plt.show()
```

```
uint8 (262, 249)
```

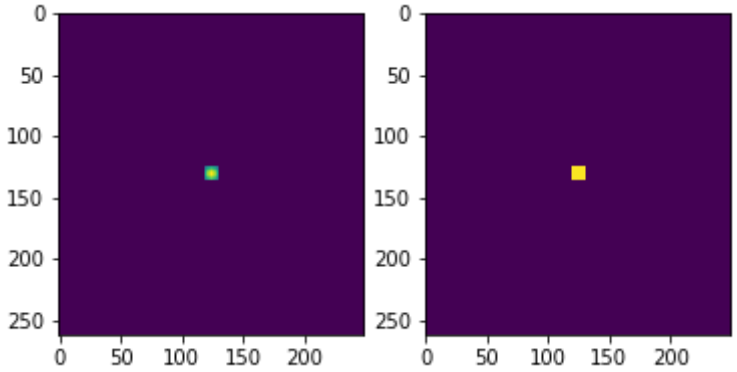


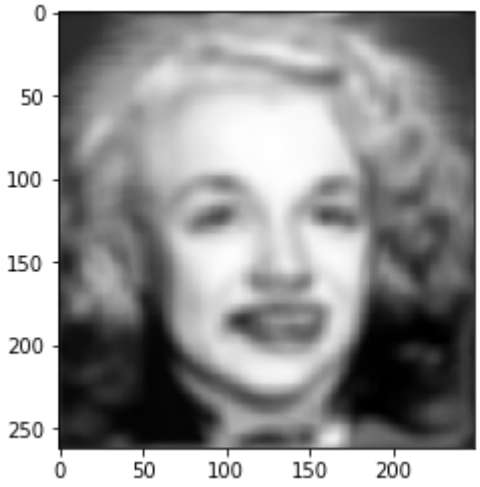
In [91]:
```
h, w = img.shape[:2]
print('image size: %d %d'%(h, w))
x_edge = abs(img[:, 1:] - img[:, :w-1])
y_edge = abs(img[1:, :] - img[:h-1, :])
```

```
image size: 262 249
```

Test on fft2

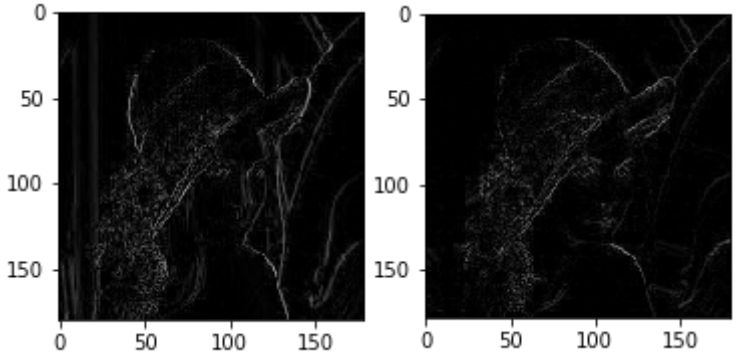
In [98]:
```
# Create 2D Gaussian Filter
h, w = img.shape[:2]

kernel_low = np.zeros((h, w))
x, y = np.meshgrid(np.linspace(-1,1,11), np.linspace(-1,1,11))
d = np.sqrt(x*x+y*y)
sigma, mu = 1.0, 0.0
g = np.exp(-( (d-mu)**2 / ( 2.0 * sigma**2 )))
g /= g.sum()
kernel_low[h//2-5:h//2+6, w//2-5:w//2+6] = g

g_low = 1 - g
g_low /= g_low.sum()
kernel_high[h//2-5:h//2+6, w//2-5:w//2+6] = g_low


plt.subplot(1,2,1)
plt.imshow(kernel_low)
plt.subplot(1,2,2)
plt.imshow(kernel_high)
plt.show()

# Apply FFT2
img_fft = np.fft.fft2(img)
kernel_high_fft = np.fft.fft2(kernel_high)
img1_rec = np.fft.fftshift(np.fft.ifft2(img_fft * kernel_high_fft))

img2_fft = np.fft.fft2(img2)
kernel_low_fft = np.fft.fft2(kernel_low)
img2_rec = np.fft.fftshift(np.fft.ifft2(img2_fft * kernel_low_fft))
plt.imshow(abs(img1_rec) + abs(img2_rec), cmap='gray')
plt.show()
```





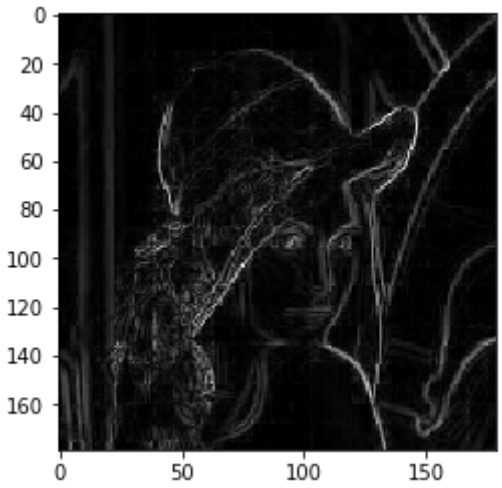
In [14]:
```
print(x_edge.shape)
print(y_edge.shape)
import matplotlib.pyplot as plt
plt.subplot(1,2,1)
plt.imshow(np.uint8(x_edge * 255), cmap='gray')
plt.subplot(1,2,2)
plt.imshow(np.uint8(y_edge * 255), cmap='gray')
plt.show()
```

```
(180, 179, 3)
(179, 180, 3)
```

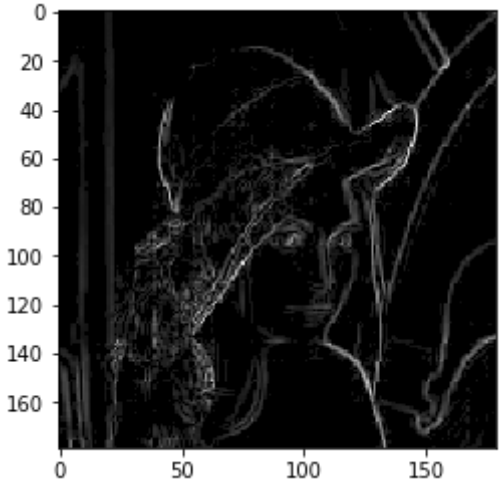


In [38]:
```
x_edge = x_edge[:-1, :]
y_edge = y_edge[:, :-1]

gradient = np.sqrt((x_edge * x_edge) + (y_edge * y_edge))

print(gradient.max(), gradient.min(), gradient.mean())

plt.imshow(np.uint8(gradient * 255), cmap='gray')
plt.show()
```

```
0.6519365 0.0 0.04650748
```



In [41]:
```
t = 0.05
gradient_cp = gradient.copy()
gradient_cp[gradient_cp < t] = 0
plt.imshow(np.uint8(gradient_cp * 255), cmap='gray')
plt.show()
```



In [46]:
```
import cv2
img_edge = cv2.Canny(np.uint8(255 * img), 100, 200)

plt.imshow(img_edge, cmap='gray')
plt.show()
```