Andrew Cornelio

1) a) We can model

$$h(t) = \begin{cases} 1/N & t \in \{0, \dots, N-1\} \\ 0 & \text{o.w.} \end{cases}$$

b) $$H(f) = \sum_{t=-\infty}^{\infty} h(t) e^{-i2\pi t f}$$

$$= \frac{1}{N} \sum_{t=0}^{N-1} e^{-i2\pi t f}$$

$$= \frac{1}{N} \left(\frac{1 - e^{-i2\pi N f}}{1 - e^{-i2\pi f}} \right)$$

$$= \frac{1}{N} e^{i\pi f} \left(\frac{1 - e^{-i\pi 2Nf}}{e^{i\pi f} - e^{-i\pi f}} \right)$$

$$= \frac{1}{N} e^{-i\pi(N-1)f} \left(\frac{e^{i\pi N f} - e^{-i\pi N f}}{e^{i\pi f} - e^{-i\pi f}} \right)$$

$$= \frac{1}{N} e^{-i\pi(N-1)f} \frac{\sin(\pi N f)}{\sin(\pi f)}$$

The division of sins results in $H(0)$ being undefined. However,

$$\lim_{f \to 0} H(f) = \frac{1}{N}$$

c) See code section. It is a band pass due to repeated low magnitude regions

2)
$$\frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{n=0}^{N-1} s[n]e^{-i2\pi kn/N}\right]e^{i2\pi km/N}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} s[n]\sum_{k=0}^{N-1} e^{-i2\pi k(n-m)/N}$$

Note we are still multiplying every term by every other term, so switching order of summation is valid:

Now note that

$$e^{-i2\pi k(n-m)/N} = \begin{cases} 1 & \text{if } n=m \\ e^{-2\pi i k l/N} & \text{o.w.} \end{cases}$$

for some $l \neq 0$

We also must observe that the sum is just over the N roots of unity so:

$$\sum_{k=0}^{N-1} e^{-i2\pi kl/N} = \sum_{k=1}^{N} \omega_N^k = \frac{\omega_N^{\,k}-1}{\omega_N - 1} = \frac{1-1}{\omega_N-1} = 0$$

So, our original sum just becomes

$$\frac{1}{N}\sum_{n=0}^{N-1} s[m] = s[m]$$

3) a) Note that

$$S[k] = \sum_{n=0}^{N-1} s[n] W_N^{-nk}$$

So, we see that in D, n will increase horizontally, and k will increase downward.

$$D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix}$$

where $\omega = e^{2\pi i/4}$

b) We can see from this that $D^4 = I$, where I is the identity matrix. This also implies $D^3 = D^{-1}$.

c) We find the eigenvalues:

$$Dv = \lambda v$$

$$D^4 v = \lambda^4 v$$

$$(D^4 - \lambda^4 I)v = 0$$

$$(D^4 - \lambda^4 D^4)v = 0$$

$$\rightarrow \operatorname{Det}(D^4 - \lambda^4 D^4) = 0$$

$$\lambda^4 = 1$$

$$\lambda = \pm 1, \pm i$$

4) $$S[k_1, k_2] = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} \sin\left(2\pi \frac{l_1 n_1}{M} + 2\pi \frac{l_2 n_2}{N}\right) W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

$$= \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} \frac{e^{2\pi i \frac{l_1 n_1}{M} + 2\pi i \frac{l_2 n_2}{N}} - e^{-2\pi i \frac{l_1 n_1}{M} - 2\pi i \frac{l_2 n_2}{N}}}{2i} W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

$$= \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} \frac{W_M^{l_1 n_1} W_N^{l_2 n_2} - W_M^{-l_1 n_1} W_N^{-l_2 n_2}}{2i} W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

$$= \frac{1}{2i}\left[\sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} W_M^{(l_1-k_1)n_1} W_N^{(l_2-k_2)n_2} - \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} W_M^{-(l_1+k_1)n_1} W_N^{-(l_2+k_2)n_2}\right]$$

$$= \frac{1}{2i}\left[\sum_{n_1=0}^{M-1} W_M^{(l_1-k_1)n_1} \delta(l_2-k_2) - \sum_{n_1=0}^{M-1} W_M^{-(l_1+k_1)n_1} \delta(l_2+k_2)\right]$$

$$= \frac{1}{2i}\left[\delta(l_1-k_1)\delta(l_2-k_2) - \delta(l_1+k_1)\delta(l_2+k_2)\right]$$

$$= \frac{i}{2}\left[\delta(l_1+k_1, l_2+k_2) - \delta(l_1-k_1, l_2-k_2)\right]$$

$$= \frac{i}{2}\left[\delta(k_1+l_1, k_2+l_2) - \delta(k_1-l_1, k_2-l_2)\right]$$

5) We can use the fact that the DFT of a product is the convolution of the DFT of the signals. The two signals are: $s[n_1, n_2]$ and $f(n_1, n_2)$, where

$$f(n_1, n_2) = (-1)^{n_1+n_2}$$

We know

$$s \longrightarrow S$$

We can calculate the DFT of f:

$$F(k_1,k_2) = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} (-1)^{n_1+n_2} W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

Note that we can model the function

$$f(n_1,n_2) = (-1)^{n_1+n_2}$$

As a cosine function

$$f(n_1,n_2) = \cos\left(2\pi \frac{l_1 n_1}{M} + 2\pi \frac{l_2 n_2}{N}\right)$$

Note that in our case $l_1 = \pm\frac{M}{2}$, and $l_2 = \pm\frac{N}{2}$. We already derived the DFT for sin of those arguments. The derivation for cosine is nearly identical, up to a negative sign and an i, so it will not be shown. Hence:

$$f(n_1,n_2) \longrightarrow \frac{1}{2}\left(\delta[k_1+l_1, k_2+l_2] + \delta[k_1-l_1, k_2-l_2]\right)$$

Note f is invariant under sign change of l_1 or l_2. Hence we can take

$$f(n_1,n_2) \longrightarrow \delta\left[k_1 - \frac{M}{2}, k_2 - \frac{N}{2}\right]$$

Now we must find the convolution

$$G = S * \delta\left[k_1 - \frac{M}{2}, k_2 - \frac{N}{2}\right]$$

But by using the properties of the δ function, we get:

$$G = \delta[k_1 - \frac{M}{2}, k_2 - \frac{N}{2}]$$

6 a) We can calculate the DFT

$$L(k_1,k_2) = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} l[n_1, n_2] W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

$$= \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{N-1} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_L(n_1 - mM, n_2 - nN) W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

Let's change the boundaries of n_1 and n_2 to make the summation easier:

$$= \sum_{n_1=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} h_L(n_1 - mM, n_2 - nN) W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

Now, since we only encounter the portion of $l[n_1, n_2]$ centered at the origin, we can get rid of the inner summations:

$$\sum_{n_1=-\frac{M}{2}}^{\frac{M}{2}-1} \sum_{n_2=-\frac{N}{2}}^{\frac{N}{2}-1} h_L(n_1, n_2) W_M^{-k_1 n_1} W_N^{-k_2 n_2}$$

$$= -4 + W_M^{k_1} + W_M^{-k_1} + W_N^{k_2} + W_N^{-k_2}$$

Note that the complex exponentials can be rewritten as cosines.

$$= -4 + 2\cos\left(\frac{2\pi k_1}{M}\right) + 2\cos\left(\frac{2\pi k_2}{N}\right)$$

b) Refer to contour plot at in the code section. It is a band pass since it amplifies frequencies in 4 regions of each quadrant and reduces them in others. This assumes the frequency is not greater than 128.

Cornelio, Andrew

May 1, 2020

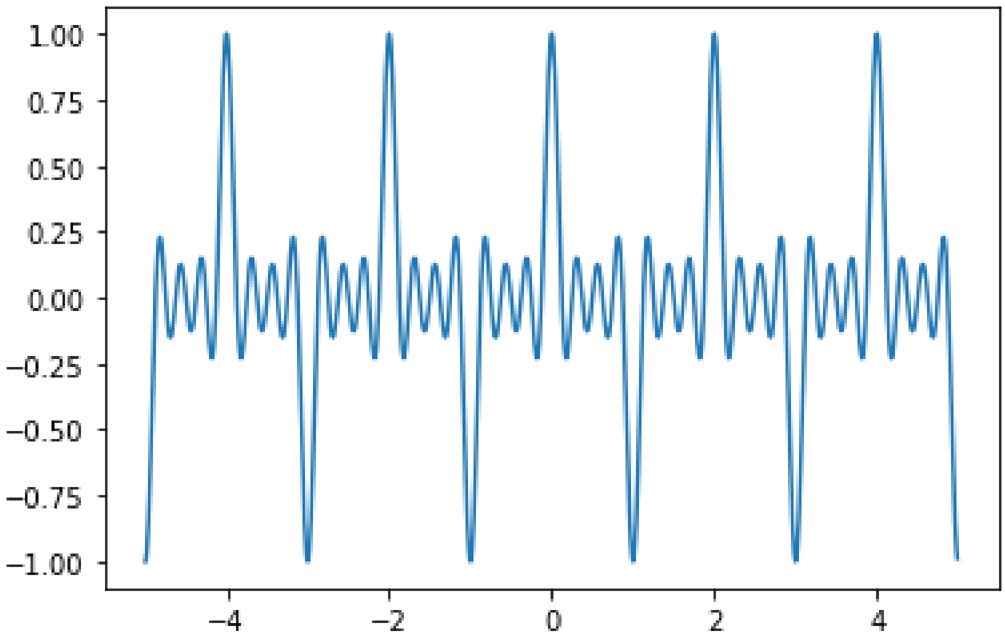
[1]:
```python
import cv2
import numpy as np
import math
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.patches import Circle
from scipy import ndimage, misc, signal
from scipy.ndimage.interpolation import shift
import argparse
import itertools
from PIL import Image
```

[2]:
```python
image_chest = cv2.imread('image-chest-xrays.png', 0).astype(np.int64);
image_boy = cv2.imread('image-Dante.png', 0).astype(np.int64);
```

0.1 1.c

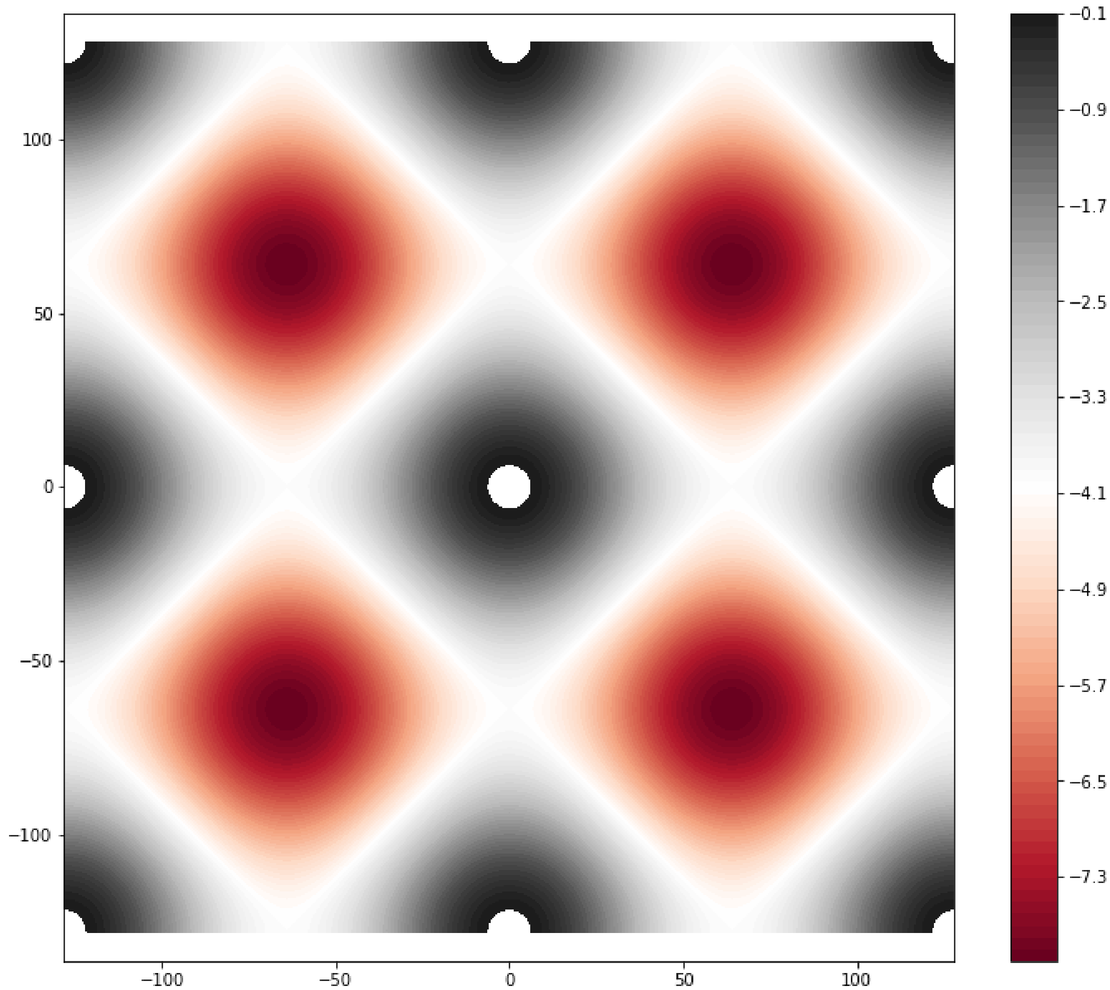
[3]:
```python
x = np.arange(-5, 5, 0.01)
S = 1/8 * np.sin(8*np.pi*x)/np.sin(np.pi*x)
plt.plot(x,S)
```

[3]:
```
[<matplotlib.lines.Line2D at 0x7fa44e2fb950>]
```



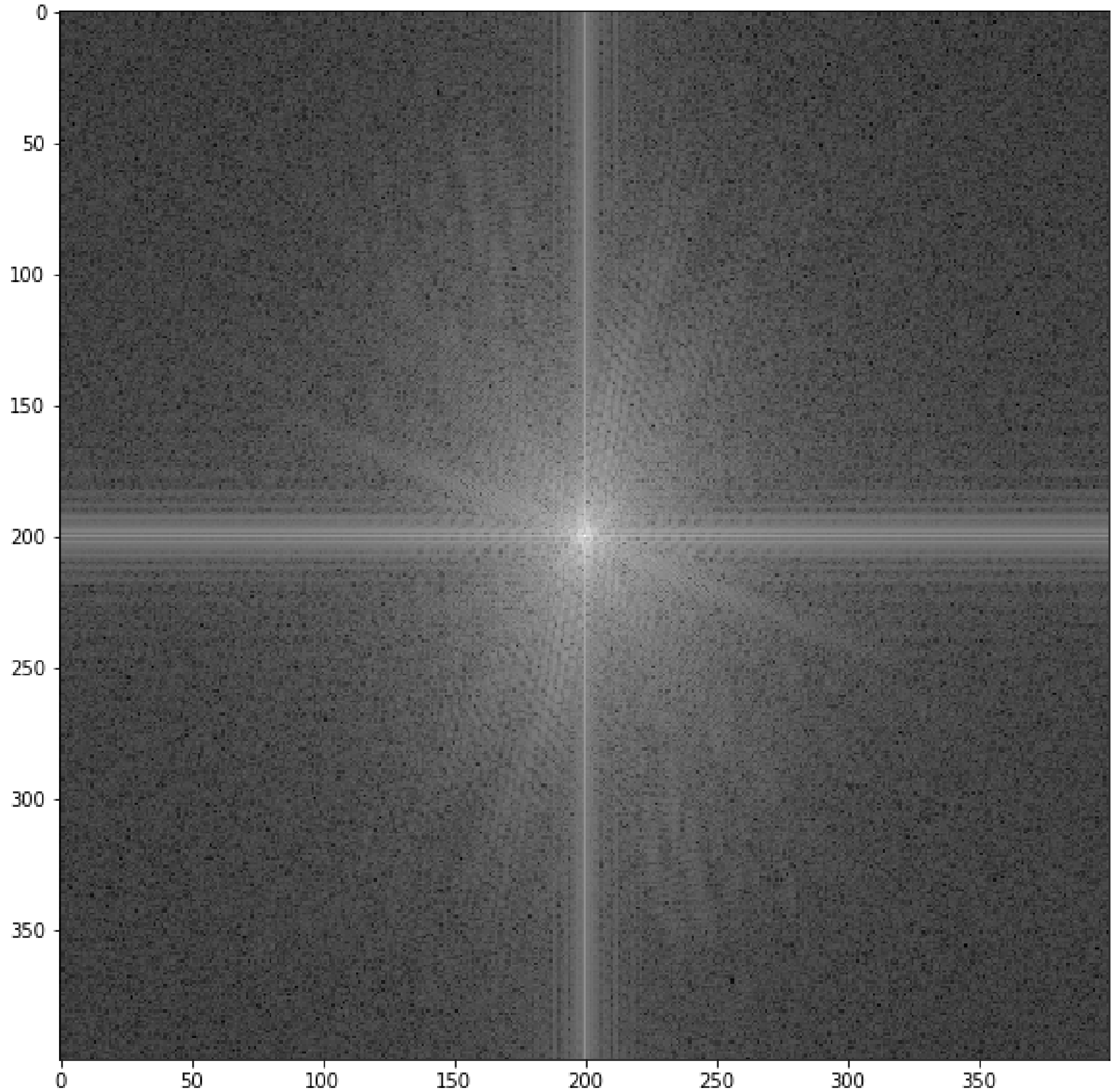
0.2 6.b

[4]:
```
x = np.arange(-128, 128, 0.1)
y = np.arange(-128, 128, 0.1)
X, Y = np.meshgrid(x, y)
Z = -4 + 2*np.cos(2*np.pi*X/128) + 2*np.cos(2*np.pi*Y/128)
levels = np.arange(-8, 0, 0.1)

fig = matplotlib.pyplot.gcf()
fig.set_size_inches(12, 10.5)
plt.axis('equal')
plt.contourf(X, Y, Z, levels, cmap='RdGy')
plt.colorbar();
```



0.3 7

[5]:
```
G = np.fft.fft2(image_boy)
AG1 = np.log(1+abs(G));
MaxVal = np.amax(AG1);
AG2 = (255*(AG1/MaxVal)).astype(np.uint8)
SAG2 = np.fft.fftshift(AG2);

fig,ax = plt.subplots(1)
ax.imshow(SAG2, cmap='gray')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(10, 10)
```



[6]:
```
x = np.arange(0, 400)
y = np.arange(0, 400)

cx = 200.
cy = 200.
r = 50.

# The two lines below could be merged, but I stored the mask
# for code clarity.
mask = (x[np.newaxis,:]-cx)**2 + (y[:,np.newaxis]-cy)**2 < r**2
U = np.fft.fftshift(G) * mask
U_shift = np.fft.fftshift(U)
```

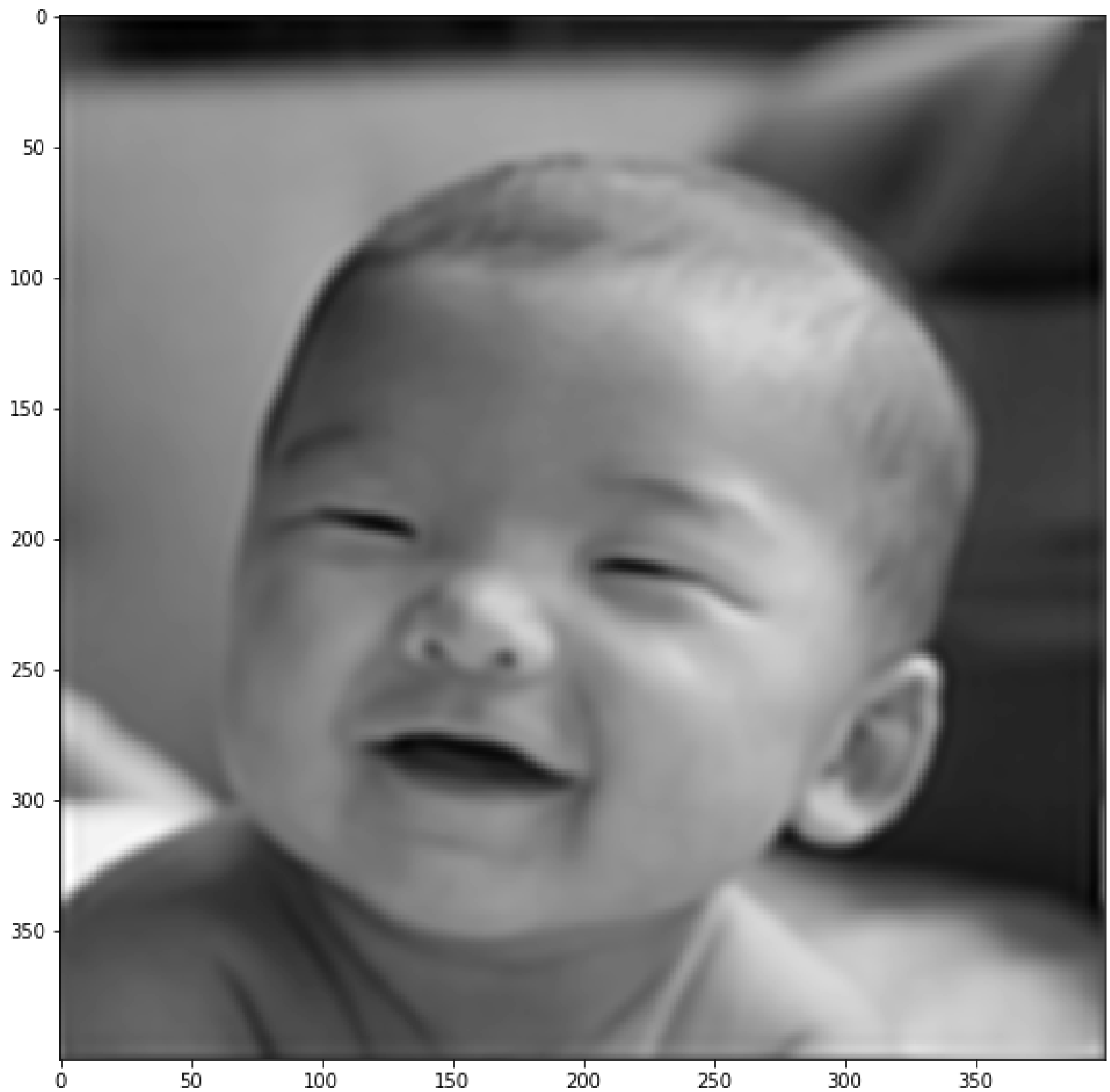
```
g = np.fft.ifft2(U_shift).astype(np.uint8)

fig,ax = plt.subplots(1)
ax.imshow(g, cmap='gray')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(10, 10)
```

```
/home/andrew/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:14:
ComplexWarning: Casting complex values to real discards the imaginary part
```

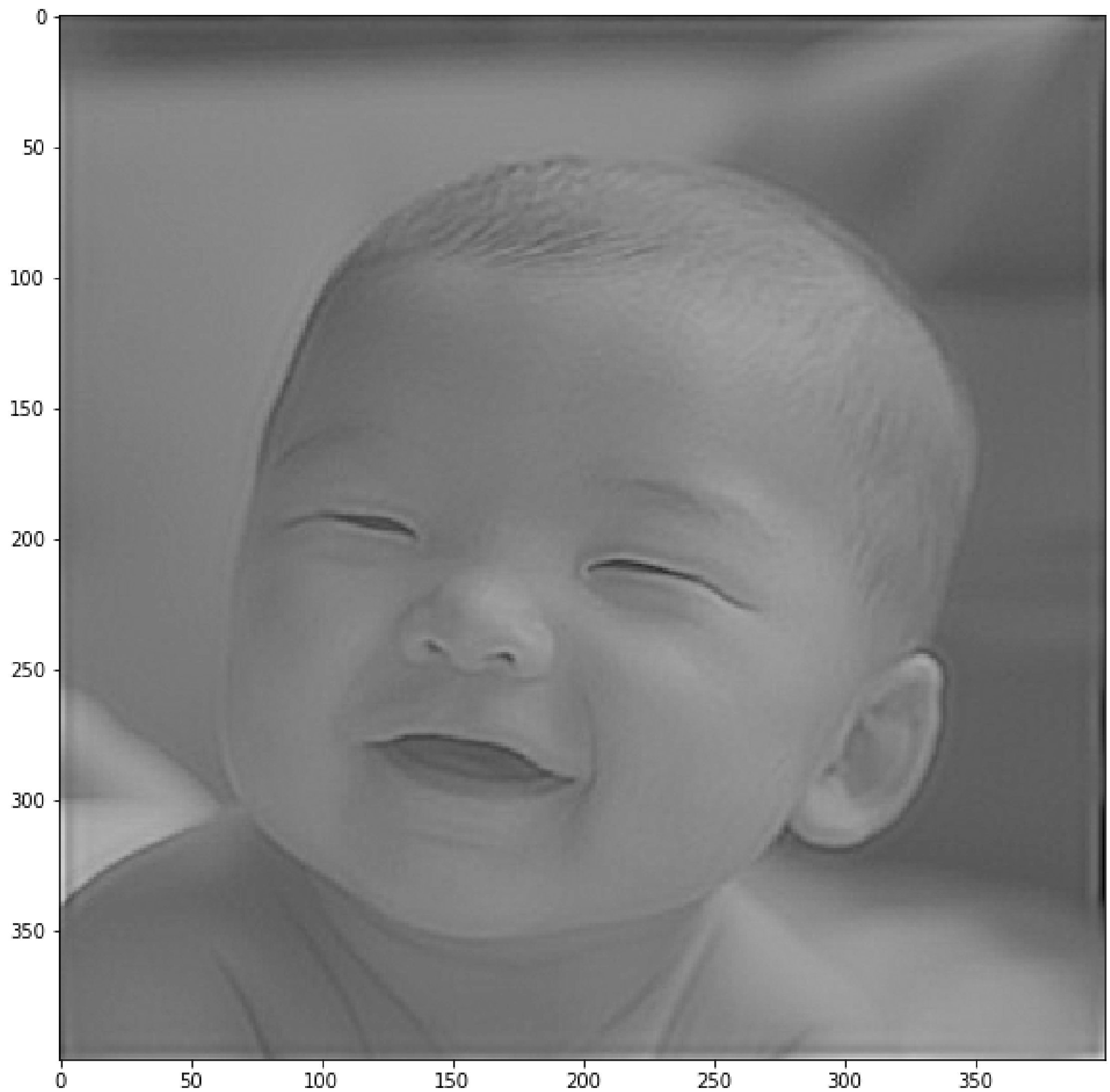
[7]:
```
x = np.arange(0, 400)
y = np.arange(0, 400)
```

```python
cx = 200.
cy = 200.
r = 50.
n = 4

# The two lines below could be merged, but I stored the mask
# for code clarity.
f = ((x[np.newaxis,:]-cx)**2 + (y[:,np.newaxis]-cy)**2)/(r**2)
B = 1 / (1 + f**n)
U = np.fft.fftshift(G) * B
U_shift = np.fft.fftshift(U)

u = np.fft.ifft2(U_shift).astype(np.uint8)

fig,ax = plt.subplots(1)
ax.imshow(u, cmap='gray')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(10, 10)
```

```
/home/andrew/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:16:
ComplexWarning: Casting complex values to real discards the imaginary part
  app.launch_new_instance()
```



[8]:
```
image_boy_sharp = image_boy + 3*(image_boy - u)
fig,ax = plt.subplots(1)
ax.imshow(image_boy_sharp, cmap='gray')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(10, 10)
```



[9]:
```
def calc_TV(image):
    image = image.astype(np.uint8)
    Kernel_hxS = 1/8 * np.array([[ 1, 2, 1],
                                 [ 0, 0, 0],
                                 [-1, -2, -1]])


    Kernel_hyS = np.transpose(Kernel_hxS)

    J_x = cv2.filter2D(image, -1, Kernel_hxS)
    J_y = cv2.filter2D(image, -1, Kernel_hyS)

    J = np.sqrt(J_x*J_x + J_y*J_y).astype("uint8")
```

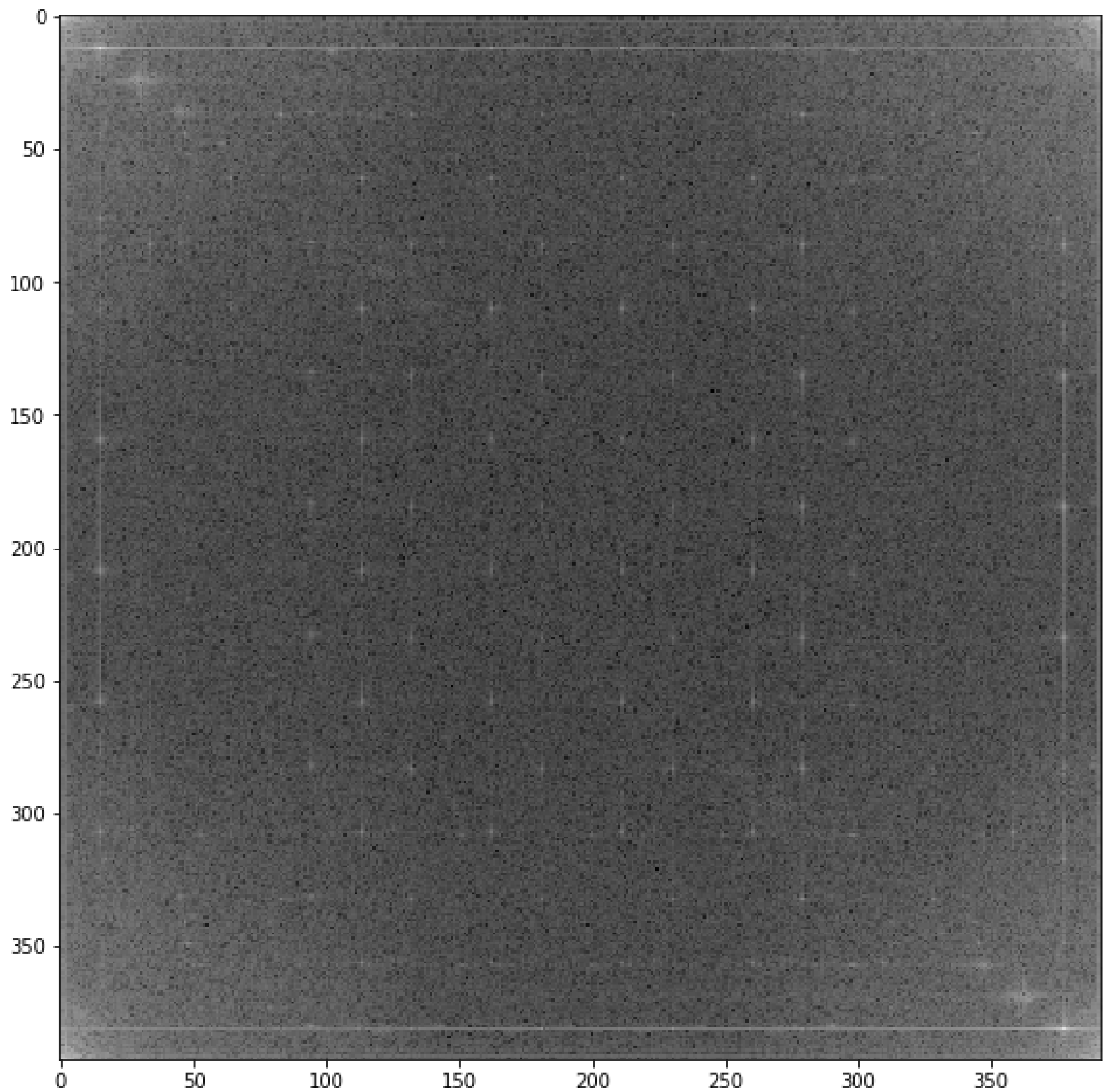
```
    return 1/J.size * np.sum(J)
```

[10]:
```
print('original image tv = ', calc_TV(image_boy))
print('sharpened image tv = ', calc_TV(image_boy_sharp))
```

```
original image tv =  1.17480625
sharpened image tv =  2.06943125
```

0.4 8

[11]:
```
G = np.fft.fft2(image_chest)
AG1 = np.log(1+abs(G));
MaxVal = np.amax(AG1);
AG2 = (255*(AG1/MaxVal)).astype(np.uint8)

fig,ax = plt.subplots(1)
ax.imshow(AG2, cmap='gray')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(10, 10)
```



[12]:
```
x = np.arange(0, 392)
y = np.arange(0, 393)


mask1 = (x[np.newaxis,:]-12)**2 + (y[:,np.newaxis]-10)**2 > 5**2
mask2 = (x[np.newaxis,:]-380)**2 + (y[:,np.newaxis]-380)**2 > 5**2

u = np.fft.ifft2(mask2 * mask1 * G).astype(np.uint8)
fig,ax = plt.subplots(1)
ax.imshow(u, cmap='gray')
fig = matplotlib.pyplot.gcf()
fig.set_size_inches(10, 10)
```

```
/home/andrew/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:8:
ComplexWarning: Casting complex values to real discards the imaginary part
```