Sheet 14

Create the directory sheet14 and make it your working directory.

Example 1. We use the two-dimensional Discrete Fourier Transform to compress an image. Go to my homepage and download the files compress.m and apollo11.mat to your working directory. Load the image file apollo11.mat and assign it to the variable `y`:

```
load apollo11

y = apollo11;
```

Now take the two-dimensional DFT:

```
ft = fft2(y);
```

Enter the compression rate, compress the DFT and then invert the DFT:

```
r = .65

ftc = compress(ft,r);

yc = ifft2(ftc);
```

Now plot both images to compare:

```
m = sprintf('Compressed %f Percent',100*r);

subplot(1,2,1)

image(yc);colormap(map);

title(m)

subplot(1,2,2)

image(y);colormap(map);

title('Original Image')
```

Try changing the compression rate, trying to maximize it while keeping a good image. Then once you've picked the optimal rate, plot each image indivdually, maximize and compare by switching between them. You need to replace the lines after the first subplot with

```
image(yc);colormap(map);

title(m)

figure

image(y);colormap(map);

title('Original Image')
```

You might see you should adjust your optimal compression!

Example 2: Now we compress using wavelets. Load the file and assign it to `y` as before. In the command window type `size(y)` to see how big the matrix is and use the dimensions to decide the highest possible level. For example, if the size is 550×800, then use the smallest dimension 550 to find the closest power of 2. You can see $2^9 = 512$ is the closest, so we'd use 9 as the highest possible level. Enter this as n and name the family of wavelets to use

```
n = 9;
fam = 'db1';
```

Perform the wavelet decomposition and enter the desired compression rate:

```
[C,S] = wavedec2(y,n,fam);
r = .9;
```

Compress and reconstruct:

```
Cc = compress(C,r);
yc = waverec2(Cc,S,fam);
```

Now display both images side by side:

```
m = sprintf('Compressed %f Percent',100*r);
subplot(1,2,1)
image(yc);colormap(map);
title(m)
subplot(1,2,2)
image(y);colormap(map);
title('Original Image')
```

Again, as in the previous example, strive for the best compression rate and then compare the images as you did in the last part of the example.

What happens is you use a lower level?

Example 3: Download the file noisycrab2.mat to your working directory from my homepage. Load it and assign it to the variable `ynoisy`:

```
load noisycrab2
ynoisy = noisycrab2;
```

Now we filter using the two-dimensional Discrete Fourier Transform. First, Fourier transform:

```
fy = fft2(ynoisy);
```

Enter the number of terms in the Fourier series to retain (the same number will be used both horizontally and vertically)

```
k = you choose;
```

Now filter the image by throwing out the small frequencies and reconstruct via the inverse Fourier transform:

```
[rows cols] = size(fy);
filtfy = [fy(1:k,1:k) zeros(k,cols-2*k) fy(1:k,cols-(k-1):cols)];
filtfy(k+1:rows-k,1:cols) = zeros(rows-2*k,cols);
a = fy(rows-(k-1):rows,1:k);
b = zeros(k,cols-2*k);
c = fy(rows-(k-1):rows,cols-(k-1):cols);
filtfy(rows-(k-1):rows,1:cols) = [a b c];
filtery = real(ifft2(filtfy));
```

To see how you did, plot the true image and the filtered image side by side as follows. First save the true image crabmosaic.mat from my homepage.

```
load crabmosaic
y=crabmosaic;
subplot(1,2,1)
image(y);colormap(map);
title('Original Image--No Noise')
subplot(1,2,2)
image(filtery); colormap(map);
title('Filtered Image')
```

Change the value of k to see if you can improve the filtered image. Add a figure statement and plot the true image and the noisy image side by side. Maximize both plots and switch back and forth to compare.

Example 4: The Fourier method we used didn't work too well in filtering out the noise. Now we try to use wavelets. Load the noisy image as before (along with the true image). Assign them to `ynoisy` and `y`, respectively. Choose the level to which you wish to decompose (let's say $n = 2$), choose the wavelet family and decompose:

```
n = 2;
fam = 'db1';
[C,S] = wavedec2(y,n,fam);
```

You will be throwing out the D, V and H components for levels 1 through 2, hence using the level 2 approximation as the filtered image.

Now construct the approximation (i.e., filtered image)

```
    An = wrcoef2('a',C,S,fam,n);
```

Then plot it side by side with the original image to see how you did.

```
    subplot(1,2,1)
    image(y);colormap(map); title('Original Image--No Noise')
    subplot(1,2,2)
    image(wcodemat(An,255));
    title('Filtered Image')
```

Try the levels $n = 1$ and $n = 3$ and try the wavelet families `'db2'` and `'bior3.7'`.