The main objective of this project is to first learn how to conduct convolution in the frequency domain, and then compare the performance of doing convolution in the spatial vs. frequency domain.

1 Background

One of the benefits of frequency-domain analysis is that convolution can be performed in the frequency domain by multiplication, which is a much simpler calculation as compared to convolution. Suppose in the time domain, we need to calculate

\[ y[n] = x[n] * h[n]. \]

Also suppose \( Y(e^{j\omega}) \), \( X(e^{j\omega}) \), and \( H(e^{j\omega}) \) are the Fourier transforms of \( y[n] \), \( x[n] \), and \( h[n] \), respectively. Then in the frequency domain, we’ll have

\[ Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}) \]

Now if we perform the inverse Fourier transform (ift), we’ll get

\[ y[n] = ift(Y(e^{j\omega})) \]

When actually implementing the above procedure, there’re two caveats that you need to keep in mind. The first is that we know usually the length of \( h[n] \) is shorter than that of \( x[n] \). The length of the Fourier transform is the same as that of the corresponding time-domain signal. This would create a problem in multiplication, as that multiplication is element-wise multiplication, requiring \( X(e^{j\omega}) \) and \( H(e^{j\omega}) \) to have the same dimension. That is, \( x[n] \) and \( h[n] \) should have the same dimension. So what do we do? We pad \( h[n] \) with zeros to make it the same length as \( x[n] \)!

The second caveat is about the boundary effect. If we would like \( y[n] \) obtained using the frequency-domain method to look exactly like the one from time-domain convolution, then we have to take care of aliasing occurred at the boundary of the
signal (or images). To do so, we need to pad both $x[n]$ and $h[n]$ to the length of $p + q - 1$, where $p$ is the length of $x[n]$ and $q$ is the length of $h[n]$.

In order to make the Fourier transform and the inverse Fourier transform as computationally efficient as possible, we usually pad $x[n]$ and $h[n]$ to the length of the smallest power of 2 that is larger than $p + q - 1$. Since we are using images in this project, in the following, I use image as an example to show the detailed procedure.

Step 1: Padding. Suppose the size of the image is $256 \times 256$, then we should pad it for $512 \times 512$.

```matlab
[m, n] = size(x);  % get the dimension of image x
xpad = zeros(512, 512);  % allocate space for the padded x
xpad(1:m, 1:n) = x;  % copy x to the upper-left corner of xpad

% we need to do the same for h.
% I’ll leave it to you to figure out how
```

Step 2: Fourier transform.

```matlab
X = fft2(xpad);  % fast Fourier Transform of images

% again, do the same for hpad
```

Step 3: Multiplication

```matlab
Y = X .* H  % element-wise multiplication
```

Step 4: Inverse transformation

```matlab
ypad = ifft2(Y);
```

Step 5: cut the upper-left corner which is where your convolution image resides.

```matlab
y = ypad(1:m, 1:n);
```

2 Tasks

1. (30 pts) Write a function `function [y] = conv2f(x, h)` where `conv2f` represents the implementation of convolution between $x$ and $h$ in the frequency domain. This function should implement the procedure described in Sec. 1. Within the function, plot a $2 \times 2$ figure of $X$, $H$, $Y$, and $y$. Note that since $X$, $H$, and $Y$ will all be matrices of complex numbers, when you plot it, use `mag()` to only plot the magnitude. Also since the DC component is
usually a large number, you probably will only see a bright spot at the center
of the image, which overshadows the rest of the image. What do you do
about it? This is a bonus point (+10). Use the image and kernel in Task 2.

2. (40 pts) Write a Matlab code testConv2f1.m using both spatial domain
method (use the MATLAB function conv2()) and frequency domain method
(use the conv2f function you just wrote) to calculate the convolution be-
tween the cameraman image http://web.eecs.utk.edu/~hqi/ece315/
project/cameraman.pgm and convolution kernel $h$ (i.e.,
a 2-D impulse response)

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Compare the results both visually and by reporting the execution time. Use
the MATLAB function tic and tac. Plot a 2×1 figure to compare visually
the result from time-domain convolution and frequency-domain convolution.
Also report the execution time as caption of the figure. Indicate the kernel
size in the caption too.

3. Write a Matlab code, testConv2f2.m. Generate convolution kernels sim-
ilar as Task 2 but of different sizes and compare the performance. Try at least
4 different sizes, e.g., $7 \times 7$, $25 \times 25$, $49 \times 49$, etc. Plot two figures. Use
Matlab function figure(). The first figure is $4 \times 2$ with visual comparison
of the convolution results from the spatial and frequency domains using the
four kernels of different sizes. In the caption, specify the kernel size. In the
second figure, plot a bar chart (or whichever ways that serve the purpose)
comparing the different run times from the two methods when different ker-
nel sizes are used.

4. Write a Matlab code, testAcorr.m that generate the autocorrelation im-
age of cameraman.pgm in both spatial and frequency domains and com-
pare the execution time. Autocorrelation is like a convolution with itself
where the kernel is the image itself. Generate a $2 \times 1$ plot with results from
time-domain and frequency domain. Again, in the caption, provide the run-
times.

3 What to Submit?

• a five-page report:
– Page 1: The $2 \times 2$ plot specified in Task 1
– Page 2: The $2 \times 1$ plot specified in Task 2
– Pages 3 and 4: The two figures specified in Task 3.
– Page 5: The $2 \times 1$ figure specified in Task 4.

• the source code: a tar file that includes the four .m files.