Fourier Solutions

Public Shift Theorem

$$F_k = \sum_{n=0}^{N-1} f_n \bar{W}^{nk} \qquad k = 0, \ldots, N-1$$

Let $g_n = f_{n-s}$

<u>VERSION I</u>

$$G_k = \sum_{n=0}^{N-1} g_n \bar{W}^{nk}$$

$$= \sum_{n=0}^{N-1} f_{n-s} \bar{W}^{nk} \times \bar{W}^{sk} \bar{W}^{-sk}$$

$$= \bar{W}^{sk} \sum_{n=0}^{N-1} f_{n-s} \bar{W}^{k(n-s)}$$

Change of variables: let $m = n-s$

$$= \bar{W}^{sk} \sum_{m=-s}^{N-1-s} f_m \bar{W}^{mk}$$

$$= \bar{W}^{sk} \underbrace{\sum_{m=0}^{N-1} f_m \bar{W}^{mk}} \qquad \text{since } f \text{ \& } W \text{ are } N\text{-periodic}$$

$$= \bar{W}^{sk} F_k \qquad k = 0, \ldots, N-1$$

$\blacksquare$

<u>VERSION II</u>

$$G_k = \sum_{n=0}^{N-1} f_{n-s} \bar{W}^{nk}$$

Change of vars: $m = n-s \Rightarrow n = m+s$

$$G_k = \sum_{m=-s}^{N-1-s} f_m \bar{W}^{(m+s)k}$$

$$= \bar{W}^{sk} \underbrace{\sum_{m=0}^{N-1} f_m \bar{W}^{mk}} \qquad \text{since } f \text{ \& } W \text{ are } N\text{-periodic}$$

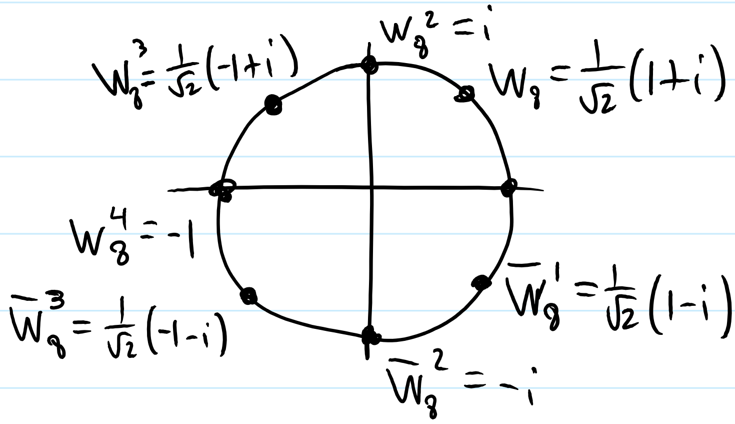
$$= \bar{W}^{sk} F_k \qquad k = 0, \ldots, N-1$$

$$= \bar{W}^{sk} \quad F_k \qquad k = 0, \dots, N-1$$

Public FFT v2 Solution

$$f = [-1, 2, 4, 3, 1, 3, 4, 2]^T$$

$$\begin{bmatrix} -1 \\ 2 \\ 4 \\ 3 \\ 1 \\ 3 \\ 4 \\ 2 \end{bmatrix} \rightarrow \begin{array}{l} -1+1 \\ 2+3 \\ 4+4 \\ 3+2 \\ (-1-1)\overline{W}_8^0 \\ (2-3)\overline{W}_8^1 \\ (4-4)\overline{W}_8^2 \\ (3-2)\overline{W}_8^3 \end{array} \qquad \begin{bmatrix} 0 \\ 5 \\ 8 \\ 5 \end{bmatrix} \quad \begin{bmatrix} -2 \\ \frac{1}{\sqrt{2}}(i-1) \\ 0 \\ \frac{1}{\sqrt{2}}(-1-i) \end{bmatrix}$$



$$\begin{bmatrix} 0 \\ 5 \\ 8 \\ 5 \end{bmatrix} \rightarrow \begin{array}{l} 0+8 \\ 5+5 \\ (0-8)\overline{W}_4^0 \\ (5-5)\overline{W}_4^1 \end{array} \qquad \begin{bmatrix} 8 \\ 10 \end{bmatrix} \rightarrow \begin{array}{l} 8+10 \\ (8-10)\overline{W}_2^0 \end{array} \quad \begin{array}{l} [18] \\ [-2] \end{array}$$

$$\begin{bmatrix} -8 \\ 0 \end{bmatrix} \rightarrow \begin{array}{l} -8+0 \\ (-8-0)\overline{W}_2^0 \end{array} \quad \begin{array}{l} [-8] \\ [-8] \end{array}$$

$$\begin{bmatrix} -2 \\ \frac{1}{\sqrt{2}}(i-1) \\ 0 \\ \frac{1}{\sqrt{2}}(-1-i) \end{bmatrix} \rightarrow \begin{array}{l} -2+0 \\ \frac{1}{\sqrt{2}}(i-1) + \frac{1}{\sqrt{2}}(-1-i) = -\frac{2}{\sqrt{2}} \\ (-2-0)\overline{W}_4^0 \\ \left(\frac{1}{\sqrt{2}}(i-\cancel{1}) + \frac{1}{\sqrt{2}}(\cancel{1}+i)\right)\overline{W}_4^1 = \frac{2i}{\sqrt{2}}(-i) = \frac{2}{\sqrt{2}} \end{array} \qquad \begin{bmatrix} -2 \\ -\frac{2}{\sqrt{2}} \end{bmatrix} \quad \begin{bmatrix} -2 \\ \frac{2}{\sqrt{2}} \end{bmatrix}$$

$$\begin{bmatrix} -2 \\ -\frac{2}{\sqrt{2}} \end{bmatrix} \rightarrow \begin{array}{l} \left(-2-\frac{2}{\sqrt{2}}\right) \\ \left(-2+\frac{2}{\sqrt{2}}\right)\overline{W}_2^0 \end{array} \qquad \begin{array}{l} \left[-2-\frac{2}{\sqrt{2}}\right] \\ \left[-2+\frac{2}{\sqrt{2}}\right] \end{array}$$

$$\begin{bmatrix} -2 \\ \frac{2}{\sqrt{2}} \end{bmatrix} \rightarrow \begin{array}{l} \left(-2+\frac{2}{\sqrt{2}}\right) \\ \left(-2-\frac{2}{\sqrt{2}}\right) \end{array} \qquad \begin{array}{l} \left[-2+\frac{2}{\sqrt{2}}\right] \\ \left[-2-\frac{2}{\sqrt{2}}\right] \end{array}$$

Recombine

$$[18]\to\begin{bmatrix}18\\-2\end{bmatrix}\to\begin{bmatrix}18\\-8\\-2\\-8\end{bmatrix}\to\begin{bmatrix}18\\-2-\frac{2}{\sqrt2}\\-8\\-2+\frac{2}{\sqrt2}\\-2\\-2+\frac{2}{\sqrt2}\\-8\\-2-\frac{2}{\sqrt2}\end{bmatrix}=\begin{bmatrix}18\\-3.414\\-8\\-0.586\\-2\\-0.586\\-8\\-3.414\end{bmatrix}$$

$$[-2]\to$$

$$[-8]\to\begin{bmatrix}-8\\-8\end{bmatrix}\to$$

$$[-8]\to$$

$$\left[-2-\frac{2}{\sqrt2}\right]\to\begin{bmatrix}-2-\frac{2}{\sqrt2}\\-2+\frac{2}{\sqrt2}\end{bmatrix}\to\begin{bmatrix}-2-\frac{2}{\sqrt2}\\-2+\frac{2}{\sqrt2}\\-2+\frac{2}{\sqrt2}\\-2-\frac{2}{\sqrt2}\end{bmatrix}\to$$

$$\left[-2+\frac{2}{\sqrt2}\right]\to$$

$$\left[-2+\frac{2}{\sqrt2}\right]\to\begin{bmatrix}-2+\frac{2}{\sqrt2}\\-2-\frac{2}{\sqrt2}\end{bmatrix}\to$$

$$\left[-2-\frac{2}{\sqrt2}\right]\to$$

Public myDCT Solutions

```
function Fdct = myDCT(f)

    % Perform an even extension of f
    fe = EvenExtension(f);

    % 2D-DFT yields purely real Fourier coefficients
    Fe = real( fft2(fe) );

    % We don't need to keep them all because of symmetry
    % (recall that the input is real-valued).
    % We only need to keep the same array size as the input.
    Fdct = IEvenExtension(Fe);
```

```
function f = myIDCT(Fdct)

    % Even extension of Fdct to restore symmetric Fourier coefficients
    Fe = EvenExtension(Fdct);

    % 2D-IDFT to get even-extended f
    f = ifft2(Fe);

    % Extract only one non-redundant part, the same size as the input.
    f = IEvenExtension(f);
```

Public JPEG Solutions

4(a) function G = myJPEGCompress(f, T, D)

```
    [h,w] = size(f);  % returns the width and height of f

  % Process tiles and blocks, using loops
  wtiles = 1:T:(w-mod(w,T));
  htiles = 1:T:(h-mod(h,T));

  G = zeros(length(htiles)*D, length(wtiles)*D);

  C = 1;
  for c = wtiles

     R = 1;
     for r = htiles
        % Extract a TxT tile from the image.
        ftile = f(r:(r+T-1),c:(c+T-1));

        % Perform the DCT on the tile
        F = myDCT(ftile);

        % Extract a DxD block of coefficients
        Fblock = F(1:D,1:D);

        % Store the block of coefficients in the output G
        G(R:(R+D-1),C:(C+D-1)) = Fblock;

        R = R + D;
     end
     C = C + D;
  end
```

4(b) function [f] = myJPEGDecompress(G, T, D)

```
  n_hblocks = size(G,1)/D;
```

```
    n_wblocks = size(G,2)/D;

    f = zeros(T*n_hblocks, T*n_wblocks);

    wblocks = 1:T:size(f,2);
    hblocks = 1:T:size(f,1);

    % Process tiles and blocks, including loops
    C = 1;
    for c = wblocks

        R = 1;
        for r = hblocks

            % Embed DCT coefs in a TxT array
            Fsub = zeros(T,T);
            Fsub(1:D,1:D) = G(R:(R+D-1),C:(C+D-1));

            % Call IDCT
            fe = myIDCT(Fsub);

            % Copy tile into image
            f(r:(r+T-1),c:(c+T-1)) = fe;

            R = R + D;
        end
        C = C + D;
    end
```

4(c)

```
% image_compression.m

f = imread('../house.jpg');
f = double(f(:,:,1));

T = 25;

%% Display the original, and the uncompressed
```

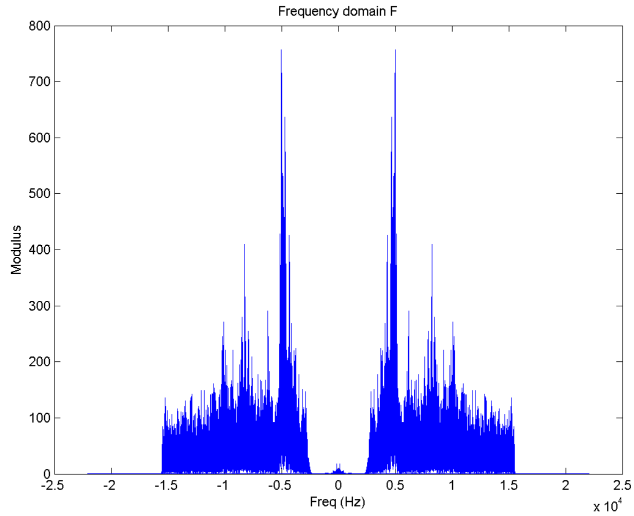
```
figure(1);

D = 11;  % compression ratio of 5:1
G = myJPEGCompress(f, T, D);
fc = myJPEGDecompress(G, T, D);
subplot(2,2,1);  % top-left in a 2x2 grid of images
imshow(fc, [0 255]);
title(['Compression Ratio = ' num2str(round(numel(f)/numel(G))) ':1']);

D = 8;  % compression ratio of 10:1
G = myJPEGCompress(f, T, D);
fc = myJPEGDecompress(G, T, D);
subplot(2,2,2);  % top-right
imshow(fc, [0 255]);
title(['Compression Ratio = ' num2str(round(numel(f)/numel(G))) ':1']);

D = 5;  % compression ratio of 25:1
G = myJPEGCompress(f, T, D);
fc = myJPEGDecompress(G, T, D);
subplot(2,2,3);  % bottom-left
imshow(fc, [0 255]);
title(['Compression Ratio = ' num2str(round(numel(f)/numel(G))) ':1']);

D = 3;  % compression ratio of 70:1
G = myJPEGCompress(f, T, D);
fc = myJPEGDecompress(G, T, D);
subplot(2,2,4);  % bottom-right
imshow(fc, [0 255]);
title(['Compression Ratio = ' num2str(round(numel(f)/numel(G))) ':1']);
```


Compression Ratio = 5:1
Compression Ratio = 10:1
Compression Ratio = 25:1
Compression Ratio = 69:1

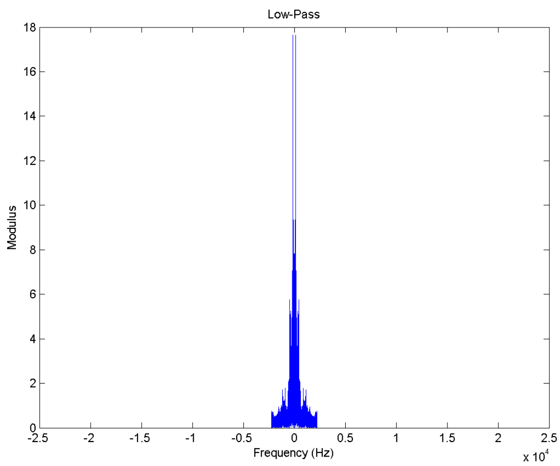
Public Filter Recording Solution

```
%% filter_audio.m

%Load some audio
blah = importdata('recording.wav');
f = blah.data;       % Rename the variables to our liking
Omega = blah.fs;     % Sampling rate (samples per second, or Hz)
clear blah;          % We don't need this anymore

% Derive some useful variables
N = length(f);            % Number of samples
L = N / Omega;            % Clip duration in seconds
t = ( 0:(N-1) )' * L/N;   % Time axis labels
omega = ifftshift( (-N/2):(N/2 - 1) )' / L;

%omega = fftshift( (-(N-1)/2 - 1):((N-1)/2 - 1) )' / L;  % Frequency axis labels

% Plot time domain signal f
figure(1);
plot(f); title('Sound');
xlabel('Time (seconds)');
ylabel('Amplitude');


%% Listen to the sound

% If you are running Matlab locally, you can play the sound using
%   sound(f, Omega);
% You can also save a wav file and download and listen to it.
audiowrite('sound_file.wav', real(f), Omega);


%% === YOUR CODE HERE ===

%% (a)

% Take DFT of audio signal
```

```
F = fft(f);  % DFT of f

% Plot modulus of frequency domain, abs(F)
figure(2);
plot(omega, abs(F));
title('Frequency domain F');
xlabel('Freq (Hz)');
ylabel('Modulus');
print('a_DFT_of_f.png', '-dpng');
```

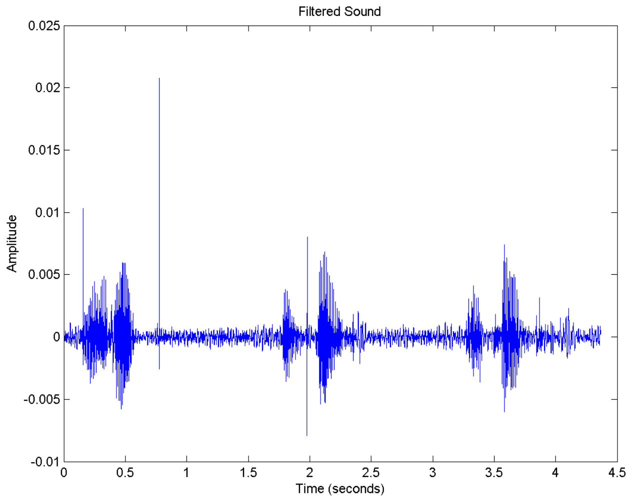



```
%% (b)

% Choose a frequency threshold
tau = 2250;  % Threshold frequenc

% Extract the high frequencies (above tau)
% Apply the threshold
mask = abs(omega)<tau;
F_lowpass = F .* mask;

% Plot the filtered Fourier coefficients
figure(3);
plot(omega, abs(F_lowpass));
title('Low-Pass');
xlabel('Frequency (Hz)');
ylabel('Modulus');
print('b_DFT_of_filtered_f.png', '-dpng');
```




```
%% (c)

%% Reconstruct the low-pass and high-pass signals
f_lowpass = ifft(F_lowpass);

sound(10*f_lowpass, Omega);
%audiowrite('sound_file.wav', real(f), Omega);
```

```
%audiowrite('sound_file.wav', real(f), Omega);

% Plot the filtered sound
figure(4);
plot(t, f_lowpass);
title('Filtered Sound');
xlabel('Time (seconds)');
ylabel('Amplitude');
print('c_filtered_f.png', '-dpng');
```




```
%% (d)
% Mr. Carver said, "Rober? He's dead. I killed him."
```