Signal Processing

in MatLab

- What is the Fourier Transform? What is the difference between the frequency domain versus spatial domain?
- Examples of signal processing:
  - 1D filtering of a song segment by lowpass/highpass and highpass emphasis Butterworth filters.
  - 2D filtering of an image by lowpass/highpass and highpass emphasis Butterworth filters.

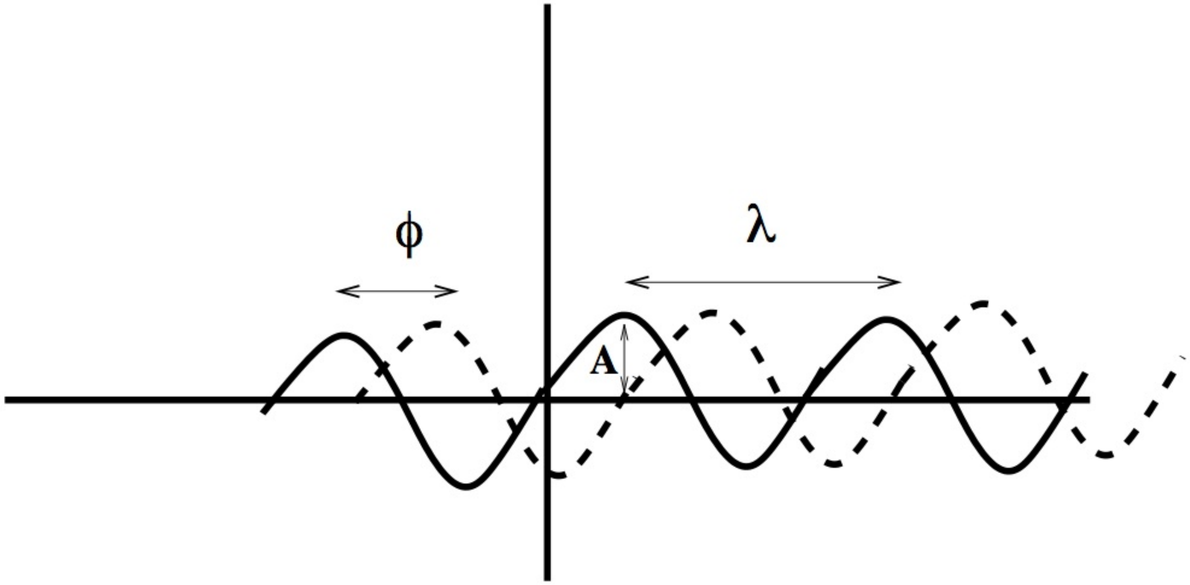
Fourier Analysis

- A 1D signal can be represented by a collection of sin functions with different frequencies, amplitudes and phases.
- A 2D image can be represented by a collection of sinusoidal surfaces with different frequencies, amplitudes and phases.
- We change from the spatial domain to the frequency domain:

$$\begin{array}{ccc} X & \Rightarrow & u \\ X, Y & \Rightarrow & u, v \end{array}$$

  where X and Y are the spatial domain coordinates and u and v are the frequency coordinates.

The Definition of a 1D Sinusoid



A 1D Sinusoid in terms of its amplitude, A, wavelength, λ, and its phase, ϕ.

- Consider a 1D sinusoid (see the above figure) given by:

$$X(t) = A\sin(ut + \phi),$$

  where A is the **amplitude**, u is the **frequency** and ϕ is the **phase**.

- λ is the **wavelength**:

$$\lambda = \frac{1}{f} = \frac{2\pi}{u},$$

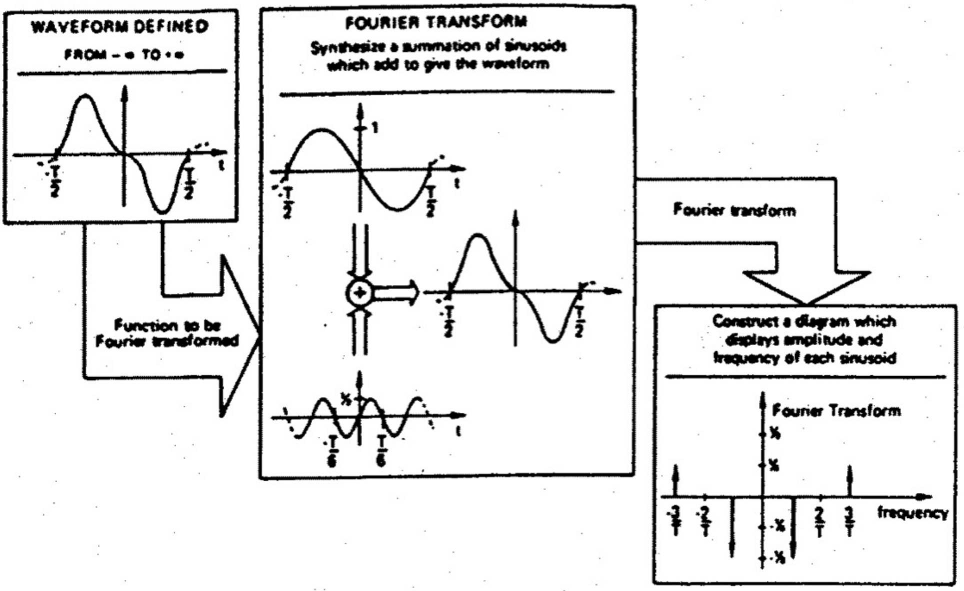
  where $u = 2\pi f$ is the frequency in cycles/radians and f is the frequency in cycles.

1D Fourier Transform

- **An amazing fact:** Any arbitrary 1D signal can be decomposed into a number of sinusoids specified by their frequency, amplitude and phase. If we add all these sinusoids together we get back the original arbitrary signal!!! So now we can look at the frequency, amplitude and phase information in 1D signals and manipulate them as we like.
- **Forward Transform:** compute all frequency, amplitude and phase from the signal.
- **Intermediate Step:** process the frequency, amplitude and phase information (for example, remove or modify all information having a certain frequency).

- **Backward Transform:** reconstruct the modified signal from the modified frequency, amplitude and phase information.

- Consider an arbitrary signal as shown in the figure below:



Two sinusoids and the their sum.

- This signal is composed of two simple sinusoids:

  $$-A\sin\left(\frac{1}{T}t\right)$$

  and

  $$\frac{A}{3}\sin\left(\frac{3}{T}t\right).$$

  $\lambda = T$ and $f = \frac{1}{T}$ for the 1^{st} sinusoid and $\lambda = \frac{T}{3}$ and $f = \frac{3}{T}$ for the 2^{nd} sinusoid.

- For general arbitrary signals, we need to be concerned with the sampling rate and the fact that the signal is finite. Nevertheless, we can usually represent the signal and recover its frequency information adequately.

The Fourier Transform Pair

- The Fourier Transform identifies the different frequencies and amplitudes of sinusoids, which together, combine to form an arbitrary waveform (note that we use $f = \frac{u}{2\pi}$ as frequency in the equations below).

- Mathematically, this relationship is stated as:

$$H(f) = \int_{-\infty}^{\infty} h(x) e^{-i2\pi f x} dx.$$

$H(f)$ is the Fourier Transform of $h(x)$. The inverse Fourier Transform is given by:

$$h(x) = \int_{-\infty}^{\infty} H(f) e^{i2\pi f x} df.$$

Complex Numbers (Review)

- The Fourier Transform computes n complex numbers for a input signal $f(x)$ of length n. Complex numbers are indicated by i. Complex number $a + ib$ has **real** part a and an **imaginary** part b. i is defined as $\sqrt{-1}$.
- Addition: $(a + ib) + (c + id) = (a + c) + i(b + d)$.
- Subtraction: $(a + ib) - (c + id) = (a - c) + i(b - d)$.
- Multiplication: $(a + ib) \times (c + id) = (ac - bd) + i(bc + ad)$.
- Division: $\frac{(a+ib)}{(c+id)} = \frac{(a+ib)(c-id)}{(c+id)(c-id)} = \frac{(ac+bd)+i(bc-ad)}{c^2+d^2}$.
- Complex conjugate: $(a + ib) \times (a - ib) = (a^2 + b^2)$, a real number.

The Discrete Fourier Transform Pair

- Without derivation, here is the forward discrete Fourier Transform:

$$F(u) = \frac{1}{n} \sum_{x=0}^{n-1} f(x) e^{-i2\pi ux/n}$$

for $u = 0, 1, 2, ..., n-1$, and the inverse discrete Fourier Transform:

$$f(x) = \sum_{u=0}^{n-1} F(u) e^{i2\pi ux/n}$$

for $x = 0, 1, 2, ..., n-1$.

Units of Image Frequencies

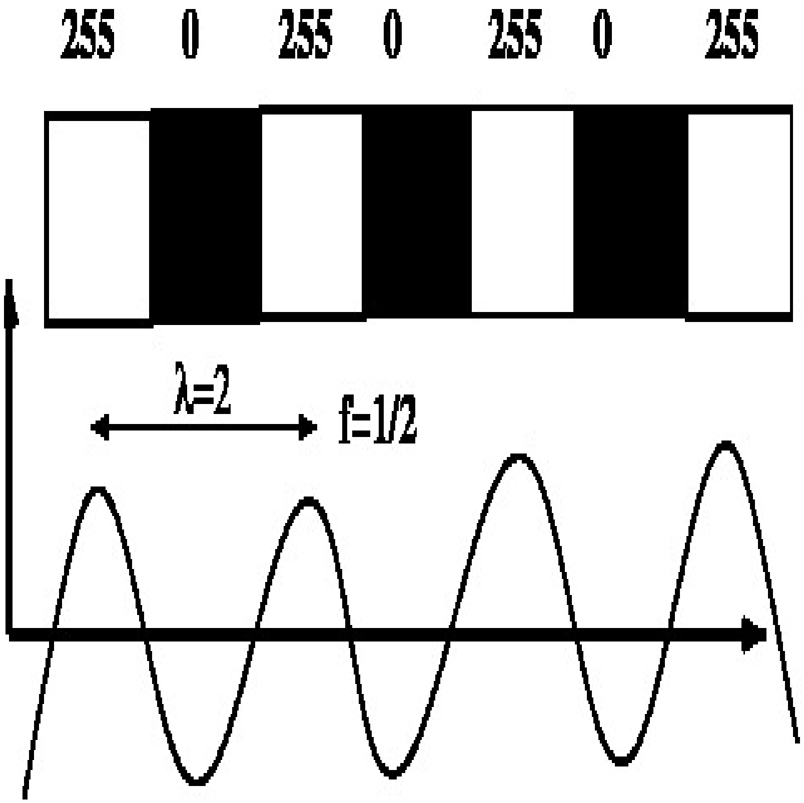
- What are the units of frequency for an image?

$$(x, y) \Rightarrow \text{“}pixels''$$

$$(u, v) \Rightarrow \text{pixels}^{-1} \text{ or } \text{cycles/pixel}$$

  The unit of frequency is the inverse of distance in the image. By the sampling theorem: because each pixel is 1 away from its closest neighbours the highest representable frequency is $\frac{1}{2}$ cycles/pixel.

- A digitized image of alternating black and white pixels (a checkerboard in 2D) has the highest frequency of $\frac{1}{2}$, i.e. all images have $-0.5 \leq u, v \leq 0.5$. The figure below shows a 1D image (signal) with highest frequency.



A 1D image (signal) with maximum frequency of $\frac{1}{2}$.

Interpretation of a Fourier Transform Computation

- For n input signal $f(x)$, the Fourier Transform computes n complex numbers of the form $a + ib$.

- It does not compute the frequencies (equivalently the wavelengths) but we know that the centered frequencies are `u=-1/2:1/n:1/2-1/n` (frequency 0 is at the middle).

- The amplitude (magnitude) of $a + ib$ can be computed as $\sqrt{a^2 + b^2}$.

- The phase angle for $a + ib$ can be computed as as $\phi = \mathtt{tan}^{-1}\left(\frac{b}{a}\right)$.

The 1D Fourier Transform in MatLab

- In Matlab, we use the function `fft` to compute the 1D Fourier Transform. `fft` actually stands for "**F**ast **F**ourier **T**ransform" as it is possible [via a binary tree] to compute the Fourier Transform in $O(nlog(n))$ time, rather than $O(n^2)$ time (required for the nested loop in the discrete Fourier Transform formula). If the time for a Fourier Transform for $n = 1000$ takes 1,000,000 time units to execute ($O(n^2)$), the Fast Fourier Transform takes 6,908 time units ($O(nlog(n))$)!!! This is a phenomenal speedup.

- The `fft` MatLab function returns the Fourier Transform result with the response for frequency 0 at the extreme left. It is often convenient to have the Fourier Transform results with frequency 0 is the middle. We call this **centered**. We can center the `fft` results using `fftshift` so that the Fourier Transform responses are ordered by the n frequencies:

$$-1/2, -1/2+1/n, -1/2+2/n ... -1/n, 0, 1/n, 2/n, ... 1/2-2/n, 1/2-1/n$$

`fftshift` basically swaps the 1^{st} and 2^{nd} parts of the `fft` results. If the signal is make up of two equal sized parts A and B then `fftshift` swaps A md B.

- We can compute the inverse Fourier Transform using `ifft`. We can shift these results using `ifftshift` to undo the forward shifting by `fftshift`. `fftshift` and `ifftshift` are the same when n is even but different when n is odd.

1D Fourier Transform Example

- Consider the Fourier Transform example (in **L21_1D_fft_example.m**):

```
% n=8 input signal
inputSignal=[7 6 5 4 6 8 10 9];
time=1:1/10:8;

% fftshift is necessary to center the frequencies
fftSignal=fftshift(fft(inputSignal));
fftAmplitude=abs(fftSignal);
fftPhase=atan(imag(fftSignal)./real(fftSignal));
fftFrequency=-1/2:1/8:1/2-1/8;
% compute the inverse Fourier Transform
inverseSignal=ifft(ifftshift(fftSignal));

% print spatial and frequency information
fprintf(’inputSignal: ’);
fprintf(’%d ’,inputSignal);
fprintf(’\n’);
fprintf(’fftFrequency: ’);
fprintf(’%8.4f’,fftFrequency);
fprintf(’\n’);
fprintf(’fftAmplitude: ’);
fprintf(’%8.4f ’,fftAmplitude);
fprintf(’\n’);
fprintf('fftPhase: ');
fprintf('%8.4f ',fftPhase);
fprintf('\n');
fprintf('fftSignal: ');
disp(fftSignal);
fprintf('\n');
fprintf('inverseSignal: ');
disp(inverseSignal);
fprintf('\n');
```

produces the following output:

```
inputSignal: 7  6  5  4  6  8  10  9
fftFrequency:-0.5000 -0.3750 -0.2500 -0.1250 0.0000 0.1250 0.2500 0.3750
fftAmplitude: 1.0000 1.1224 2.2361 10.4279 55.0000 10.4279 2.2361 1.1224
fftPhase:  0.0000 -0.0448 -0.4636 -1.2668 0.0000 1.2668 0.4636 0.0448
fftSignal: 1.0000+0.0000i -1.1213+0.0503i -2.0000+1.0000i  3.1213-9.9497i
          55.0000+0.0000i  3.1213+9.9497i -2.0000-1.0000i -1.1213-0.0503i
inverseSignal: 7.0000 6.0000 5.0000 4.0000 6.0000 8.0000 10.0000 9.0000
```

- We can see the following:

- There are 8 frequencies from -0.5 to 0.3750 in steps of 0.1250 (which is=1/8).
- The amplitudes show that the largest values (55.0000 and 10.4279) are at or around frequency zero. Most of the “power” of a signal is in its low frequencies.
- The phase show the shift in the sinusoids for the 8 Fourier Transform responses.
- The responses of the Fourier Transform are (generally) complex numbers.
- The recovered signal from the inverse Fourier Transform is exactly the input signal (the complex parts are zero or so small that MatLab does not print them)

Filtering the "Long and Winding Road" Sound Segment

- Recall the use of MatLab to listen to a segment of the left and right channels as well as the combined stereo channel of the "Long and Winding Road" via **L10_long_winding_road1.m**. **L10_long_winding_road2.m** played the same segment at 50% and 150% speeds, as well as backwards at normal speed (so called devil music).

- Here we regenerate the segment after applying a Butterworth lowpass filter, a Butterworth highpass filter and a Butterworth high frequency emphasis filter.

- A **lowpass** filter is a filter that lets low frequencies pass (relatively) unchanged while attenuating (reducing) high frequencies. A lowpassed signal should have its high frequencies significantly reduces (there might correspond to noise or subtle sound effects).

- A **highpass** filter is a filter that lets high frequencies pass (relatively) unchanged while attenuating (reducing) low frequencies. A highpassed signal should have its high frequencies relatively unchanged but its low frequencies significantly attenuated. Since most of the signal's power is low frequency information highpass filtering will effectively destroy the original signal.

- A **highpass emphasis** filter is a filter that lets the low frequencies pass unchanged but emphasizes or scales up the high frequency information.

1D Butterworth Filters

- The Butterworth lowpass filter can be given as:

$$H_{low}(u) = \frac{1}{\left[1 + \left(\frac{D(u)}{D_0}^{2n}\right)\right]}.$$

  Note that this is a frequency centered filter, so the Fourier Transform of the input signal must also be centered. $D(u)$ is $\sqrt{u^2}$, which is just the absolute value of u, $|u|$.

- The Butterworth highpass filter can be given as:

$$H_{high}(u) = \frac{1}{\left[1 + \left(\frac{D_0}{D(u)}^{2n}\right)\right]}$$

- The Butterworth highpass emphasis filter can be written as:

$$H_{emphasis}(u) = \text{emphasis\_factor} + \frac{1}{\left[1 + \left(\frac{D_0}{D(u)}^{2n}\right)\right]}$$

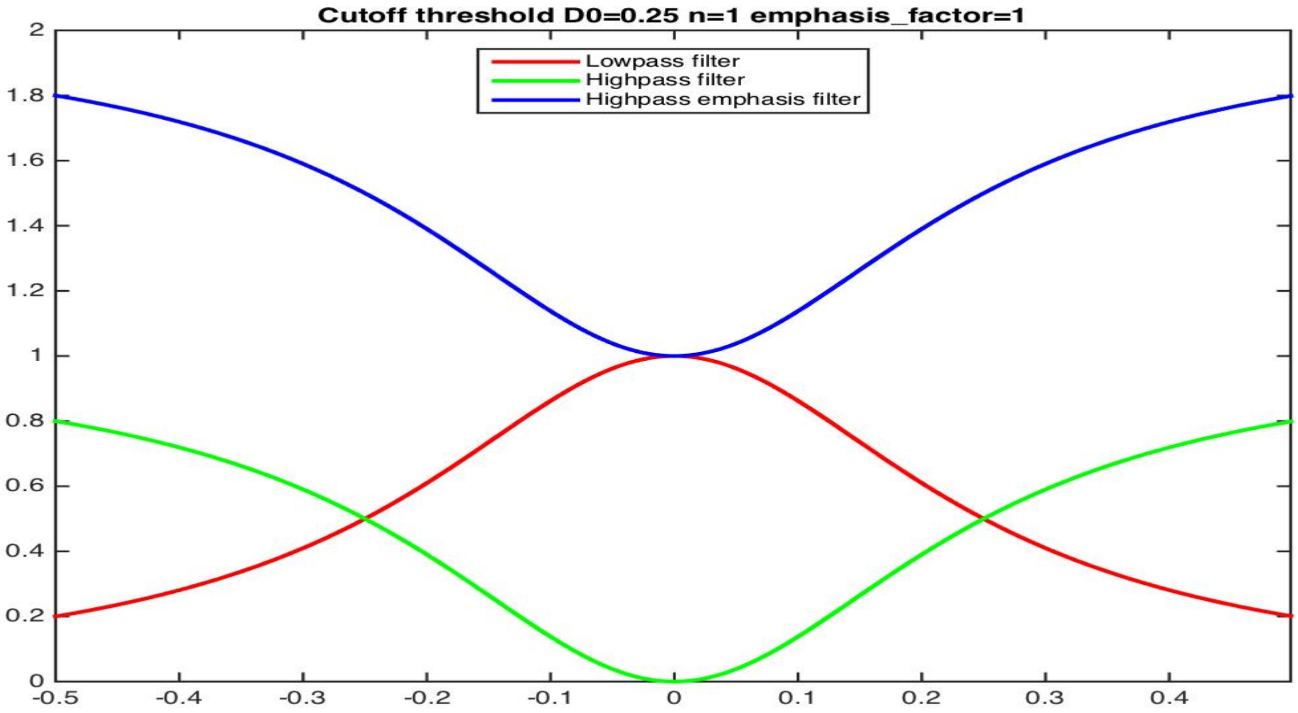
If $\text{emphasis\_factor} = 0.0$ this is simply the highpass filter. If $\mathbf{emphasis\_factor = 1.0}$ the low frequencies are mostly left alone but higher frequency can be scaled by up to 2.0.

- We can plot these 3 filters using the MatLab code (in **L21butterworth_plot.m** given below:

```
% Set parameters for lowpass, highpass and
% highpass emphasis filtering
% frequencies vary from -1/2 to 1/2-1/number_of_samples

number_of_samples=512;
u=-1/2:1/number_of_samples:1/2-1/number_of_samples;
D0=0.25;
```

```
n=1; % 1st order Butterworth filter
emphasis_factor=1.0; % highpass emphasis filtering

% lowpass filter
H_lowpass=L21butterworth_lowpass1D(D0,n,u);
% highpass filter
H_highpass=L21butterworth_highpass1D(D0,n,u);
% highpass emphasis filter
H_highpass_emphasis=L21butterworth_highpass_emphasis1D(D0,...
                                  n,u,emphasis_factor);

plot(u,H_lowpass,’linewidth’,2.0,’color’,’red’)
hold on
plot(u,H_highpass,’linewidth’,2.0,’color’,’green’)
hold on
plot(u,H_highpass_emphasis,’linewidth’,2.0,’color’,’blue’)
hold off
axis([u(1) u(number_of_samples) 0.0 2.0]);
title([’Cutoff threshold D0=’ num2str(D0) ’ n=’ num2str(n) ...
       ’ emphasis\_factor=’ num2str(emphasis_factor)]);
legend(’Lowpass filter’,’Highpass filter’,...
       ’Highpass emphasis filter’,’location’,’north’);
print filter_plot1D.jpg -djpeg
```

- The figure below shows the plot of the lowpass, highpass and highpass emphasis Butterworth filters for $D0 = 0.5$, $n = 1$ and $\mathrm{emphasis\_factor} = 1.0$ for the 512 frequencies $-1/2 : 1/512 : 1/2 - 1/512$ generated by **L21butterworth_plot.m**.



Plots of the lowpass Butterworth filter (red), the highpass Butterworth filter (green) and the highpass emphasis Butterworth filter (blue). The lowpass/highpass filters have their maximum/minimum values for frequency 0.0. The highpass emphasis filter has its minimum value at frequency 0.0 and maximum values at frequencies, -1/2 and 1/2-1/512.

- Lastly, we show the MatLab code for the three Butterworth filters in **L21butterworth_lowpass1D.m**, **L21butterworth_highpass1D.m** and **L21butterworth_highpass_emphasis1D**:

```
function [H]=L21butterworth_lowpass1D(D0,n,u)
% H is centered as u is centered
% D(u) is abs(u)
n=2*n;
H=1./(1+(abs(u)./D0).^n);
end
```

```
function [H]=L21butterworth_highpass1D(D0,n,u)
% H is centered as u is centered
% D(u) is abs(u)
n=2*n;
H=1./(1+(D0./abs(u)).^n);
end
```

```
function [H]=L21butterworth_highpass_emphasis1D(D0,n,u,emphasis_factor)
% H is centered as u is centered
% D(u) is abs(u)
n=2*n;
H=emphasis_factor+1./(1+(D0./abs(u)).^n);
end
```

Butterworth Filters & “Long Winding Road” Song

- Consider the following MatLab code (in **L21long_and_winding_road3.m**).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Play highpass, lowpass and highpass emphasis
% filtered stereo segment of the "Long Winding Road"
% This segment was first used in Lecture 10
% We perform filtering in the frequency domain
% using the Fast Fourier Transform (fft) with
% frequency centering (fftshift)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all
% fs=44100 samples/second is used
% 606171 digital samples - takes time to read!
[road,fs]=audioread(’road.wav’);
fprintf(’road is a %d row by %d column array\n’,...
         size(road,1),size(road,2));

% The left and right channel signals are
% the two columns of the road array
left=road(:,1);
right=road(:,2);

% time of the clip
number_of_samples=numel(left);
```

```
% time to pause for a sound playback
pause_time=(1/44100)*number_of_samples*1.05;
% 606171 samples
number_of_samples=length(left);
time=(1/44100)*number_of_samples;
t=linspace(0,time,number_of_samples);

% Make sure your speakers are turned on
fprintf('playing original stereo sound track\n');
% Play stereo audio signal at sample rate fs
sound(road,fs)
% Wait for the left signal to play
pause(pause_time);

% Set parameters for lowpass, highpass and
% highpass emphasis filtering
% frequencies vary from -1/2 to 1/2-1/number_of_samples
u=-1/2:1/number_of_samples:1/2-1/number_of_samples;
D0_low=0.15;
D0_high=0.25;
n=1; % 1st order Butterworth filter
emphasis_factor=1.0; % highpass emphasis filtering

% apply Fourier Transform to left/right signals
% save so we can use the same result three times, once
% for lowpass filtering, once for highpass filtering
% and once for highpass emphasis filtering
% transpose to make column vector
org_fftLeft=fftshift(fft(left))’;
org_fftRight=fftshift(fft(right))’;

% lowpass filter
H_lowpass=L21butterworth_lowpass1D(D0_low,n,u);
% filter
fftLeft=org_fftLeft.*H_lowpass;
fftRight=org_fftRight.*H_lowpass;
% inverse filter
ifftLeft_low=ifft(ifftshift(fftLeft));
ifftRight_low=ifft(ifftshift(fftRight));
lowStereo=[ifftLeft_low;ifftRight_low];
% Play lowpass stereo audio signal at sample rate fs
fprintf(’playing lowpass stereo sound track\n’);
% lowStereo could have small imaginary parts due
% to roundoff error: ignore these by using the
% real part only
soundsc(real(lowStereo),fs)
% Wait for the lowpass stereo signal to play
pause(pause_time);

% highpass filter filter
H_highpass=L21butterworth_highpass1D(D0_high,n,u);
% filter
fftLeft=org_fftLeft.*H_highpass;
fftRight=org_fftRight.*H_highpass;
ifftLeft_high=ifft(ifftshift(fftLeft));
```

```
ifftRight_high=ifft(ifftshift(fftRight));
highStereo=[ifftLeft_high;ifftRight_high];
% Play highpass stereo audio signal at sample rate fs
fprintf(’playing highpass stereo sound track\n’);
% highStereo could have small imaginary parts due
% $to roundoff error: ignore these by using the
% real part only
soundsc(real(highStereo),fs)
% Wait for the highpass stereo signal to play
pause(pause_time);

% highpass emphasis filter
H_highpass_emphasis=...
   L21butterworth_highpass_emphasis1D(D0_high,n,u,emphasis_factor);
fftLeft=org_fftLeft.*H_highpass_emphasis;
fftRight=org_fftRight.*H_highpass_emphasis;
% Play stereo audio signal at sample rate fs
ifftLeft_emphasis=ifft(ifftshift(fftLeft));
ifftRight_emphasis=ifft(ifftshift(fftRight));
highEmphasisStereo=[ifftLeft_emphasis;ifftRight_emphasis];
% Play highpass emphasis stereo audio signal at sample rate fs
fprintf(’playing highpass emphasis stereo sound track\n’);
% highEmphasisStereo could have small imaginary parts due
% to roundoff error: ignore these by using the
% real part only
soundsc(real(highEmphasisStereo),fs)
% Wait for the highpass emphasis left signal to play
pause(pause_time);
```

```
% keep the graphs to a reasonable size
number_of_samples=1000;
time=(1/44100)*number_of_samples;
% time value for x dimension of plot
t=linspace(0,time,number_of_samples);

% original signal
% plot left signal against t
figure
plot(t,left(1:number_of_samples))
hold on
plot(t,right(1:number_of_samples))
xlabel(’time (sec)’);
ylabel(’signal strength’)
legend(’left channel’,’right channel’);
title({[num2str(number_of_samples)...
        ’ left and right channel original digital’];...
       [’samples of Long and Winding Road Song’]});
hold off
print original_long_and_winding_road_signal.jpg -djpeg

% lowpass signal
% plot left signal against t
figure
plot(t,abs(ifftLeft_low(1:number_of_samples)))
hold on
plot(t,abs(ifftRight_low(1:number_of_samples)))
```

```
xlabel(’time (sec)’);
ylabel(’signal strength’)
legend(’left channel’,’right channel’);
title({[num2str(number_of_samples)...
        ’ left and right channel lowpass digital’];...
       [’samples of Long and Winding Road Song’]});
hold off
print lowpass_long_and_winding_road_signal.jpg -djpeg

% highpass signal
% plot left signal against t
figure
plot(t,abs(ifftLeft_high(1:number_of_samples)))
hold on
plot(t,abs(ifftRight_high(1:number_of_samples)))
xlabel(’time (sec)’);
ylabel(’signal strength’)
legend(’left channel’,’right channel’);
title({[num2str(number_of_samples)...
        ’ left and right channel highpass digital’];...
       [’samples of Long and Winding Road Song’]});
hold off
print highpass_long_and_winding_road_signal.jpg -djpeg

% highpass emphasis signal
% plot left signal against t
figure
plot(t,abs(ifftLeft_emphasis(1:number_of_samples)))
```

```
hold on
plot(t,abs(ifftRight_emphasis(1:number_of_samples)))
xlabel('time (sec)');
ylabel('signal strength')
legend('left channel','right channel');
title({[num2str(number_of_samples)...
        ' left and right channel highpass emphasis digital'];...
       ['samples of Long and Winding Road Song']});
hold off
print highpass_emphasis_long_and_winding_road_signal.jpg -djpeg
```

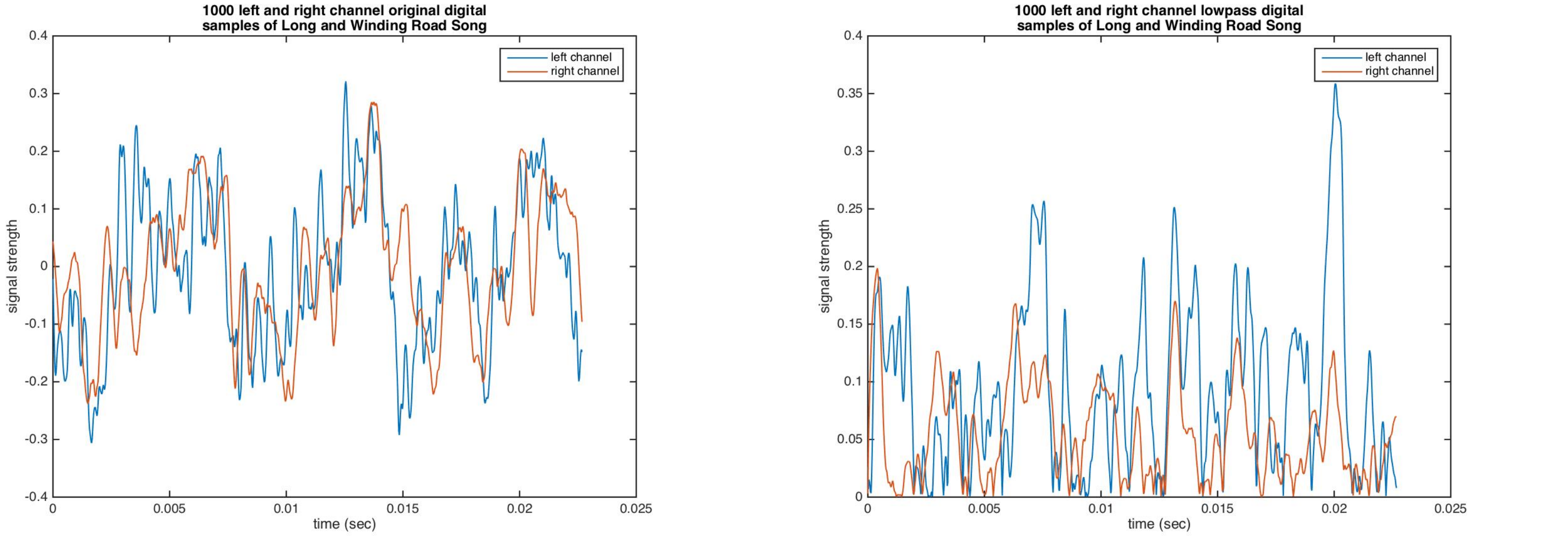
- Some comments:
  - The lowpass sound segment has the general sound of the song but is a bit “blurred” out. Higher frequency information has been attenuated and the singer’s words and subtle sounds are attenuated.
  - The highpass sound segment has the words and subtle sounding a bit clearer but the general overall sounds associated with low frequency information have been seriously attenuated. The song does

not sound right now!

  - The highpass emphasis sound segment has the low frequency information preserved but with the high frequency information emphasized by a factor up to 2. The word and noise have been emphasized. Maybe this result is better than both the lowpass and highpass results but also better than the original segment!
  - All three filtered results seem to have a "skipping" sound at 1 second intervals. The reason for this in currently unknown.

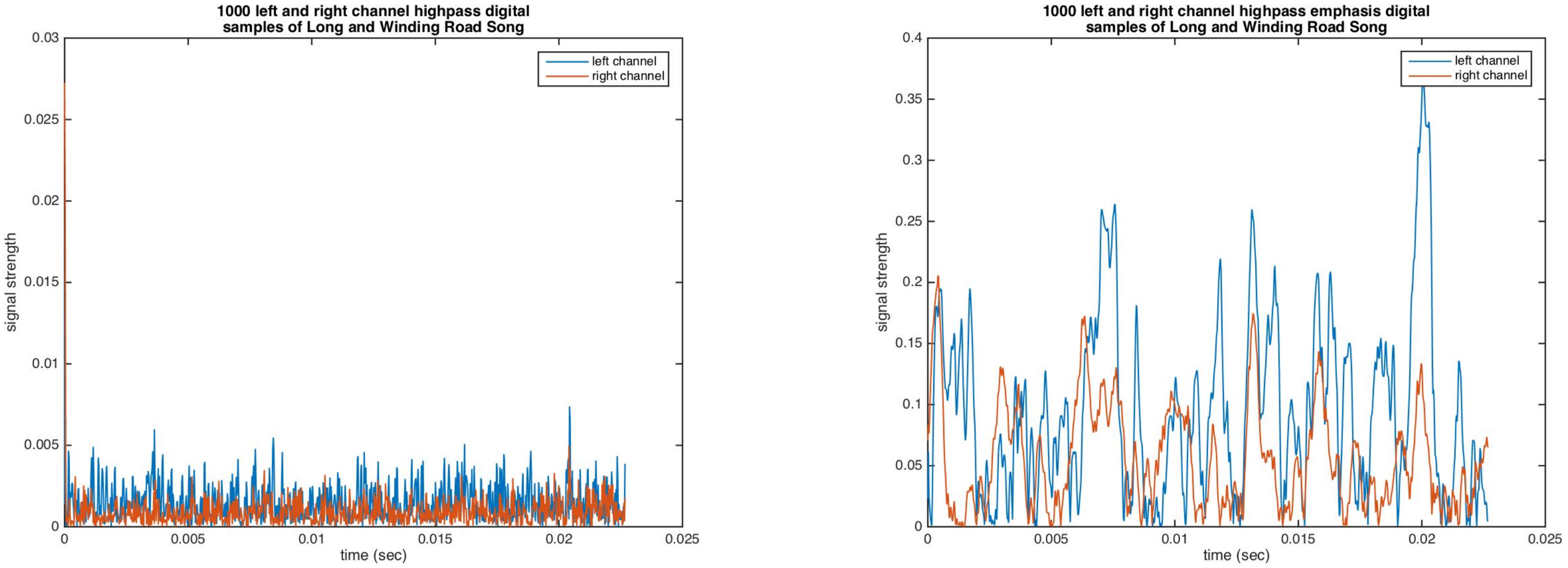
- We show the frequency responses for first 1000 samples of the signal for the left and right channels for the original signal, the lowpass signal, the highpass signal and the highpass emphasis signals.

- The lowpass signal is some what smoothed out with respect to the original signal.
- The highpass signal has the low frequency information removed (so most of the original signal is missing).
- The highpass emphasis is similar to the original signal.



(a) (b)

Left and right sound channels for (a) the original signal and (b) the lowpass signal for 1000 samples.



(a) (b)

Left and right sound channels for (a) the highpass signal and (b) the highpass emphasis signal for 1000 samples.

2D Fourier Transform

- The Fourier Transform can be performed in 2D. A natural 2D signal is a 2D image. Things work as in the 1D case except now we have two frequency variables, (u, v), such that for a $N \times M$ image $u = -1/2 : 1/N : 1/2 - 1/N$ and $v = -1/2 : 1/M : 1/2 - 1/M$. Just as in the 1D case, we have a Fourier Transform pair to go forward from the spatial domain (x, y) into frequency space (u, v) and to go backwards from the frequency domain ((u, v) to the spatial domain (x, y).

- We can think of the 2D Fourier Transform as decomposing a $n \times m$ 2D arbitrary signal (for example, an image) as a set of 2D sinusoial surfaces described by equations:

$$A(u,v)\sin(2\pi(u+v)+\phi(u,v))$$

where $A(u,v)$ is the amplitude of $F(u,v)$ $\left[\sqrt{Re^2+Im^2}\right]$ and $\phi(u,v)$ is the phase of $F(u,v)$ $\left[\tan^{-1}\left(\frac{Im}{Re}\right)\right]$, where Re and Im are the real and imaginary of the complex number computed for $F(u,v)$.

The 2D Discrete Fourier Transform

- The forward Fourier Transform can be discretized as:

$$F(u,v) = \sum_{x=0}^{n-1}\sum_{y=0}^{l-1} f(x,y)e^{-i2\pi\frac{ux+vy}{n}}\Delta x\Delta y,$$

where $u = 0, 1, 2, ..., n-1$ and $v = 0, 1, 2, ..., l-1$, n and l are integer powers of 2 and $\Delta x = \frac{1}{n}$ and $\Delta y = 1$.

- The inverse Fourier Transform can be discretized as:

$$f(x,y) = \sum_{u=0}^{l-1}\sum_{v=0}^{n-1} F(u,v)e^{i2\pi\frac{ux+vy}{n}}\Delta u\Delta v,$$

where $x = 0, 1, 2, ..., n-1$ and $y = 0, 1, 2, ..., l-1$, n and l are integer powers of 2 again and $\Delta u = 1$ and $\Delta v = \frac{1}{l}$.

The 2D Fast Fourier Tranform and MatLab

- MatLab has forward and inverse Fast Fourier Transform functions, `fft2` and `ifft2`.

- As in the 1D case, it is often advantageous to have the Fourier Transform results centered (frequency (0,0) is in the middle of the array). This is because it is very natural to have filters mathematically specified as being centered. We use `fftshift` and `ifftshift` to do this. If we view the image as 4 rectanglar boxes labelled left to right, top to bottom as A, B, C and D, the `fftshift` just swaps A and C and B and D. `ifftshift` does the same thing if both dimensions are even (but something different if one or more of the two dimensions is odd).

Illustration of the 2D FourierTransform

- The table below shows a 11×11 image:

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Original A

A simple 11×11 image (see Digital Picture Processing, Volume 1, 2^{nd} Edition, page 22 of Rosenfeld and Kak).

- MatLab code to set up the image is given below and is in **L21_11by11_array_fft2_example.m**. The real and imaginary parts of the Fourier Transform and their amplitudes and phase information are printed.

```
% Set up 11 by 11 image
A=zeros(11,11,’single’);
A(5,5)=1.0; A(5,6)=1.0; A(5,7)=1.0;
A(6,5)=1.0; A(6,7)=1.0;
A(7,5)=1.0; A(7,6)=1.0; A(7,7)=1.0;

% print out the data as 11 by 11 rows using
% vectorized fprintf
fprintf(’Original A\n’);
for i=1:11
fprintf(’ %2d & ’,A(i,:));
fprintf(’\n’);
end % for i
fprintf(’\n\n’);

% B is now a 11 by 11 array of complex numbers
B=fftshift(fft2(A));

% Print out the real parts of B
% The output has &’s for latex
fprintf(’\nReal values of B\n’);
for i=1:11
fprintf(’%5.2f & ’,real(B(i,:)))
fprintf(’\n’);
end % for i
fprintf(’\n\n’);

% Print out the imaginary parts of B
% The output has &’s for latex
fprintf(’\nImaginary values of B\n’);
for i=1:11
fprintf(’%5.2f & ’,imag(B(i,:)))
fprintf(’\n’);
end % for i
fprintf(’\n\n’);

% Print out the amplitude values of B
% The output has \&’s for latex
fprintf(’\nAmplitude values of B\n’);
for i=1:11
amplitude=squeeze(sqrt(real(B(i,:).ˆ2)+imag(B(i,:).ˆ2)));
fprintf(’%5.2f & ’,amplitude);
fprintf(’\n’);
end % for i
fprintf(’\n\n’);

% Print out the phase values of B
% The output has &’s for latex
fprintf(’\nPhase values of B\n’);
for i=1:11
phase=squeeze(atan(imag(B(i,:))./real(B(i,:))));
fprintf(’%5.2f & ’,phase);
fprintf(’\n’);
end % for i
fprintf(’\n\n’);
```

u/v	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	0.00	-0.30	0.98	-1.50	1.37	-0.56	-0.56	1.37	-1.50	0.98	-0.30
-4	-0.30	0.45	-0.73	0.78	-0.37	-0.43	1.24	-1.69	1.58	-1.07	0.53
-3	0.98	-0.73	0.40	-0.11	-0.09	0.21	-0.33	0.51	-0.75	0.97	-1.07
-2	-1.50	0.78	-0.11	0.09	-0.82	1.80	-2.31	1.89	-0.65	-0.75	1.58
-1	1.37	-0.37	-0.09	-0.82	2.84	-4.83	5.51	-4.33	1.89	0.51	-1.69
0	-0.56	-0.43	0.21	1.80	-4.83	7.22	-7.52	5.51	-2.31	-0.33	1.24
1	-0.56	1.24	-0.33	-2.31	5.51	-7.52	7.22	-4.83	1.80	0.21	-0.43
2	1.37	-1.69	0.51	1.89	-4.33	5.51	-4.83	2.84	-0.82	-0.09	-0.37
3	-1.50	1.58	-0.75	-0.65	1.89	-2.31	1.80	-0.82	0.09	-0.11	0.78
4	0.98	-1.07	0.97	-0.75	0.51	-0.33	0.21	-0.09	-0.11	0.40	-0.73
5	-0.30	0.53	-1.07	1.58	-1.69	1.24	-0.43	-0.37	0.78	-0.73	0.45

Real values of FT

The real FT values for the shifted image in the above figure (see Digital Picture Processing, Volume 1, 2^{nd} Edition, page 23 of Rosenfeld and Kak).

u/v	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	0.00	-0.09	0.63	-1.73	3.01	-3.88	3.88	-3.01	1.73	-0.63	0.09
-4	-0.09	0.29	-0.84	1.71	-2.55	2.96	-2.72	1.95	-1.01	0.31	0.00
-3	0.63	-0.84	0.88	-0.77	0.60	-0.46	0.38	-0.33	0.22	0.00	-0.31
-2	-1.73	1.71	-0.77	-0.64	1.79	-2.08	1.49	-0.55	-0.00	-0.22	1.01
-1	3.01	-2.55	0.60	1.79	-3.28	3.10	-1.62	0.00	0.55	0.33	-1.95
0	-3.88	2.96	-0.46	-2.08	3.10	-2.12	0.00	1.62	-1.49	-0.38	2.72
1	3.88	-2.72	0.38	1.49	-1.62	-0.00	2.12	-3.10	2.08	0.46	-2.96
2	-3.01	1.95	-0.33	-0.55	-0.00	1.62	-3.10	3.28	-1.79	-0.60	2.55
3	1.73	-1.01	0.22	0.00	0.55	-1.49	2.08	-1.79	0.64	0.77	-1.71
4	-0.63	0.31	-0.00	-0.22	0.33	-0.38	0.46	-0.60	0.77	-0.88	0.84
5	0.09	-0.00	-0.31	1.01	-1.95	2.72	-2.96	2.55	-1.71	0.84	-0.29

Imaginary values of FT

The imaginary FT values for the shifted image in the above figure (see Digital Picture Processing, Volume 1, 2^{nd} Edition, page 23 of Rosenfeld and Kak).

u/v	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	0.18	0.82	1.53	0.85	0.00	0.00	0.00	0.00	0.00	0.39	0.16
-4	0.82	0.83	0.39	0.00	0.00	0.00	0.00	0.00	0.67	0.90	0.84
-3	1.53	0.39	0.00	0.00	0.00	0.00	0.00	0.17	0.49	1.44	1.91
-2	0.85	0.00	0.00	0.00	0.00	0.00	2.14	2.35	0.36	1.80	2.47
-1	0.00	0.00	0.00	0.00	0.00	3.86	6.20	4.60	1.06	1.69	1.10
0	0.00	0.00	0.00	0.00	3.86	8.00	8.28	5.17	1.06	0.61	0.00
1	0.00	0.00	0.00	2.14	6.20	8.28	7.13	3.60	0.29	0.00	0.00
2	0.00	0.00	0.17	2.35	4.60	5.17	3.60	0.75	0.00	0.00	0.00
3	0.00	0.67	0.49	0.36	1.06	1.06	0.29	0.00	0.00	0.00	0.00
4	0.39	0.90	1.44	1.80	1.69	0.61	0.00	0.00	0.00	0.00	0.00
5	0.16	0.84	1.91	2.47	1.10	0.00	0.00	0.00	0.00	0.00	0.09

Amplitude values of FT

The amplitude values of the FT responses for the shifted image for the above 11×11 image.

u/v	-5	-4	-3	-2	-1	0	1	2	3	4	5
-5	0.29	0.57	0.86	1.14	1.43	-1.43	-1.14	-0.86	-0.57	-0.29	0.00
-4	0.57	0.86	1.14	1.43	-1.43	-1.14	-0.86	-0.57	-0.29	0.00	0.29
-3	0.86	1.14	1.43	-1.43	-1.14	-0.86	-0.57	-0.29	-0.00	0.29	0.57
-2	1.14	1.43	-1.43	-1.14	-0.86	-0.57	-0.29	0.00	0.29	0.57	0.86
-1	1.43	-1.43	-1.14	-0.86	-0.57	-0.29	0.00	0.29	0.57	0.86	1.14
0	-1.43	-1.14	-0.86	-0.57	-0.29	0.00	0.29	0.57	0.86	1.14	1.43
1	-1.14	-0.86	-0.57	-0.29	-0.00	0.29	0.57	0.86	1.14	1.43	-1.43
2	-0.86	-0.57	-0.29	-0.00	0.29	0.57	0.86	1.14	1.43	-1.43	-1.14
3	-0.57	-0.29	0.00	0.29	0.57	0.86	1.14	1.43	-1.43	-1.14	-0.86
4	-0.29	-0.00	0.29	0.57	0.86	1.14	1.43	-1.43	-1.14	-0.86	-0.57
5	-0.00	0.29	0.57	0.86	1.14	1.43	-1.43	-1.14	-0.86	-0.57	-0.29

Phase values of FT

The phase values of the FT responses for the shifted image for the above 11×11 image.

Display of 2D Fourier Transform Spectra as an image.

- Since many image spectra decrease rapidly as a function of frequency, high-frequency terms tend to be obscured when displayed in image form. Thus it is common to display $\log(1 + |F(u,v)|)$ instead of $|F(u,v)|$.

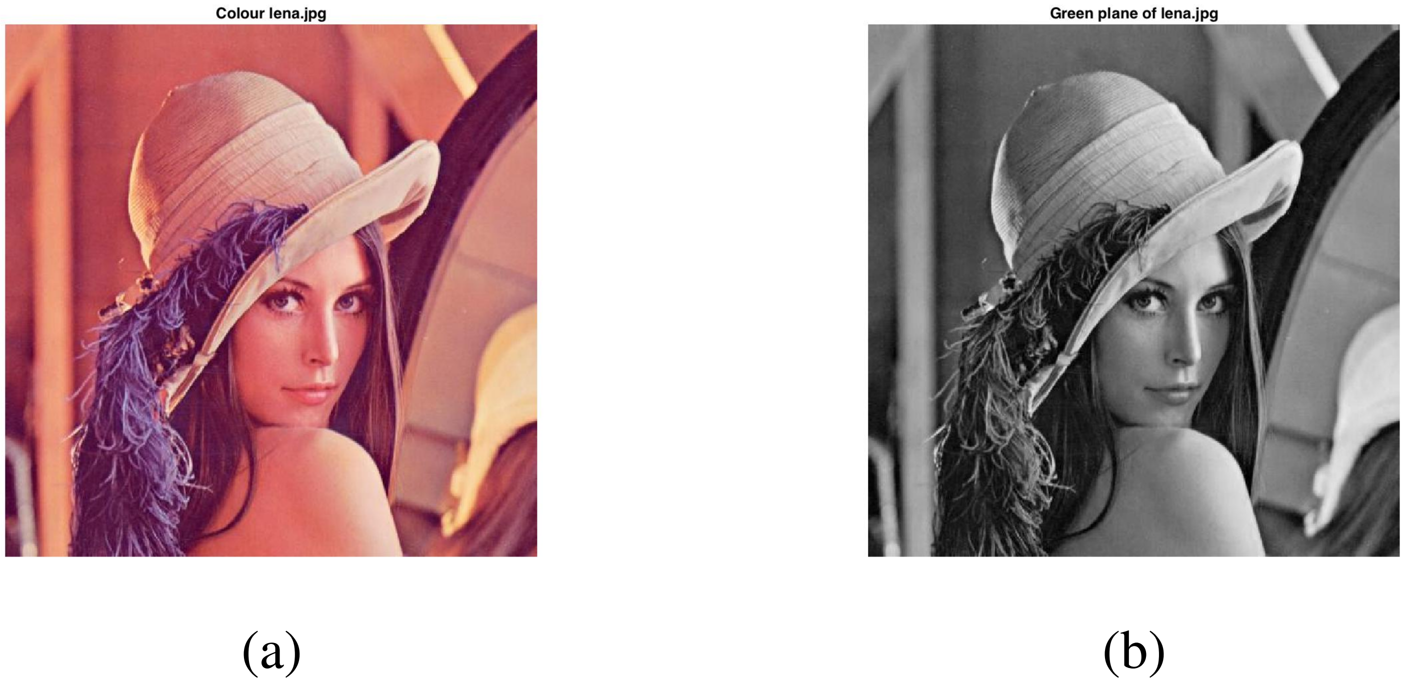
- Consider the MatLab code in **L21log_spectra.m**. This code shows how to do the computation: computes the green image as the 2^{nd} plane of the colour lena.jpg image.

```
I=imread(’lena.jpg’);
figure
imshow(I,[]);
title(’Colour lena.jpg’);
print colour_lena.jpg -djpeg

greenI=squeeze(I(:,:,2));
```

```
figure
imshow(greenI,[]);
title('Green plane of lena.jpg');
print green_lena.jpg -djpeg

FTuncentered=fft2(greenI);
FTcentered=fftshift(FTuncentered);

values=abs(FTcentered);
logValues=log(1.0+values);
fprintf('Minimum FT value: %10.6f\n',min(values(:)));
fprintf('Maximum FT value: %10.6f\n',max(values(:)));
fprintf('Minimum FT log value: %10.6f\n',min(logValues(:)));
fprintf('Maximum FT log value: %10.6f\n',max(logValues(:)));

figure
imshow(abs(FTuncentered),[]);
title('Uncentered fft2');
print uncenteredFTlena.jpg -djpegclose all
I=imread('lena.jpg');
figure
imshow(I,[]);
title('Colour lena.jpg');
print colour_lena.jpg -djpeg

greenI=squeeze(I(:,:,2));
figure
imshow(greenI,[]);
```

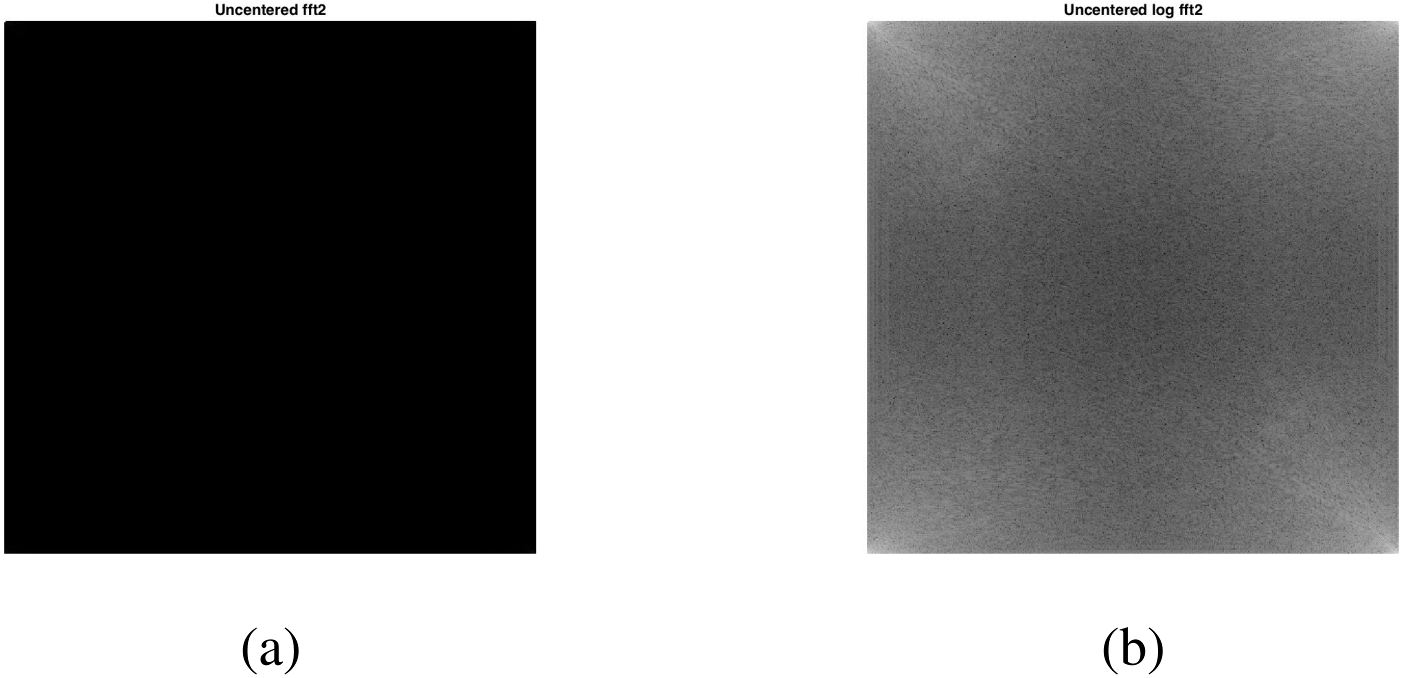
```
title(’Green plane of lena.jpg’);
print green_lena.jpg -djpeg

FTuncentered=fft2(greenI);
FTcentered=fftshift(FTuncentered);

values=abs(FTcentered);
logValues=log(1.0+values);
fprintf(’Minimum FT value: %10.6f\n’,min(values(:)));
fprintf(’Maximum FT value: %10.6f\n’,max(values(:)));
fprintf(’Minimum FT log value: %10.6f\n’,min(logValues(:)));
fprintf(’Maximum FT log value: %10.6f\n’,max(logValues(:)));

figure
imshow(abs(FTuncentered),[]);
title(’Uncentered fft2’);
print uncenteredFTlena.jpg -djpeg

figure
imshow(abs(FTcentered),[]);
title(’Centered fft2’);
print centeredFTlena.jpg -djpeg

figure
imshow(log(1+abs(FTuncentered)),[]);
title(’Uncentered log spectra of fft2’);
print uncenteredFTloglena.jpg -djpeg
```

```
figure
imshow(log(1+abs(FTcentered)),[]);
title('Centered fft2');
print centeredFTloglena.jpg -djpeg
```



(a) The colour lena image and (b) the green plane as a grayvalue image.

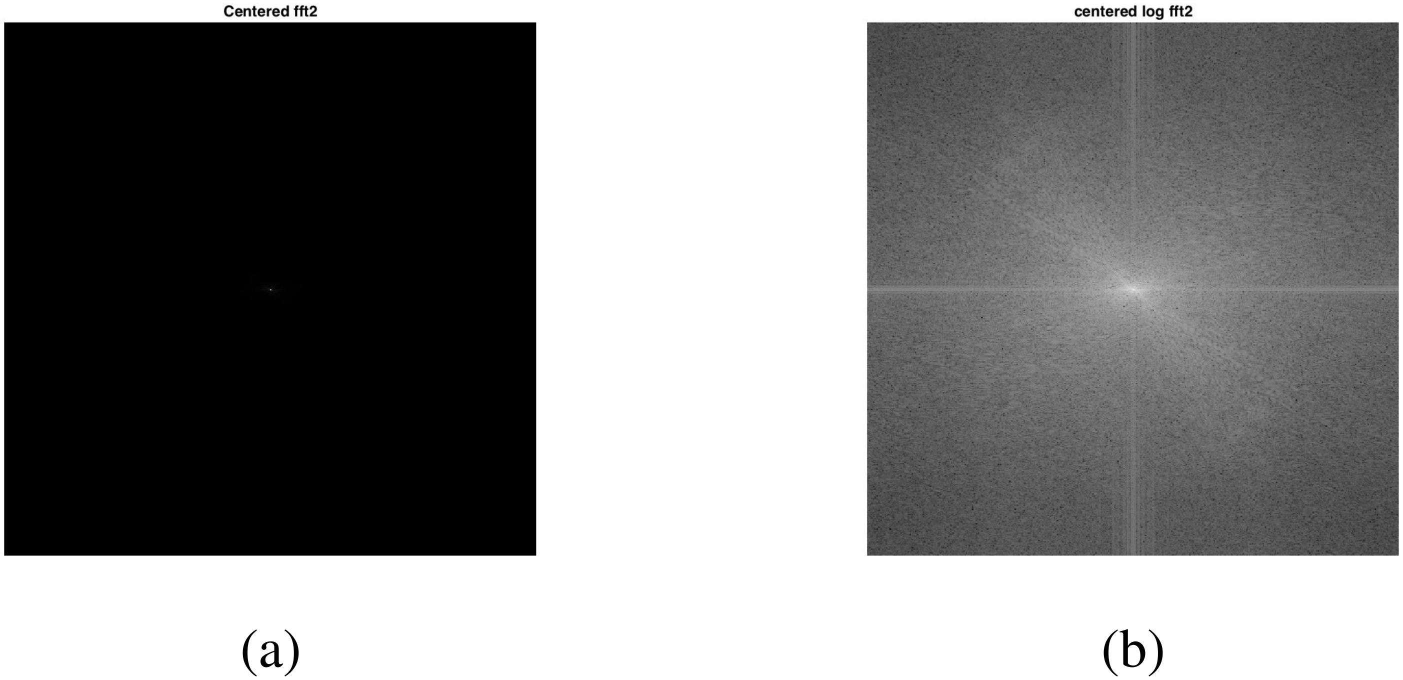
- The next figure $uncentered|F(u,v)|$ and $\log(1+|F(u,v)|)$ for the green lena image.



(a) (b)

The uncentered FT spectra and the uncentered log spectra images of the green lena image.

- The next figure $centered|F(u,v)|$ and $\log(1+|F(u,v)|)$ for the green lena image.



(a) (b)

The centered FT spectra and the centered log spectra images of the green lena image. Look carefully and you will see a small white dot at te center in (a). Obviously, (b) gives the viewer the most useful information in this and the images in the previous figure.

- The minimum and maximum values for the FT speactra are 2.14 and 25959445.00 which is scaled to be between 0 and 255 to make a gray-value image. Naturally, the spread between the min and max values is not a good linear fit to these grayvalues

- The minimum and maximum values for the FT log spectra are 1.14 and 17.07 which can be rescaled to fit between 0 and 255 easily.

- Its is not possible to see the maximum white pixels in the 4 corners in the FT spectra, but we can see these maxima in the FT log spectra image (spread out in the 4 corners).

- We can see a small white dot in the centered FT spectra image. However, not only do we see a much larger white dot but also horizontal, vertical and diagonal lines. There are obviously horizontal, vertical and diagonal edges in the image.

2D Butterworth Filtering the Green Lena Image

- We lowpass, highpass and highpass emphasis filter the green lena image. In the next section we lowpass, highpass and highpass emphasis the colour lena image.

- We present the MatLab code to filter the green lena image below (in **L21green_butterworth2D.m**):

```
I=imread('lena.jpg');
figure
imshow(I,[]);
title('Colour lena.jpg');
print colour_lena.jpg -djpeg

greenI=squeeze(I(:,:,2));
figure
imshow(greenI,[]);
title('Green plane of lena.jpg');
print green_lena.jpg -djpeg
```

```
FTcentered=fftshift(fft2(greenI));

% set parameters for lowpass, highpass and
% highpass emphasis filtering
% frequencies vary from -1/2 to 1/2-1/dimension
dim1=size(greenI,1);
dim2=size(greenI,2);
u=-1/2:1/dim1:1/2-1/dim1;
v=-1/2:1/dim2:1/2-1/dim2;
D0_low=0.15;
D0_high=0.25;
n=1; % 1st order butterworth filter
emphasis_factor=1.0; % highpass emphasis filtering

Hlow=L21butterworth_lowpass2D(D0_low,n,u,v);
Hhigh=L21butterworth_highpass2D(D0_high,n,u,v);
Hemphasis=L21butterworth_highpass_emphasis2D(D0_high,...
          n,u,v,emphasis_factor);

lowpassfft=Hlow.*FTcentered;
highpassfft=Hhigh.*FTcentered;
emphasisfft=Hemphasis.*FTcentered;

% No need for ifftshift here (why?)
Igreen_low=abs(ifft2(lowpassfft));
Igreen_high=abs(ifft2(highpassfft));
Igreen_emphasis=abs(ifft2(emphasisfft));
```

```
figure
% there many be some small imaginary values
% due to roundoff error in ifft
imshow(abs(Igreen_low),[]);
title('Lowpass filter green lena image');
print lowpass_butterworth_green_lena.jpg -djpeg

figure
imshow(abs(Igreen_high),[]);
title('Highpass filter green lena image');
print highpass_butterworth_green_lena.jpg -djpeg

figure
imshow(abs(Igreen_emphasis),[]);
title('Highpass emphasis filter green lena image');
print highpass_emphasis_butterworth_green_lena.jpg -djpeg

[U,V]=meshgrid(u,v);

figure
mesh(U,V,Hlow);
hold on
mesh(U,V,Hemphasis);
hold off
axis([-1/2 1/2-1/dim1 -1/2 1/2-1/dim2 0 2]);
alpha(0.5);
view(-37.5,30.0);
```

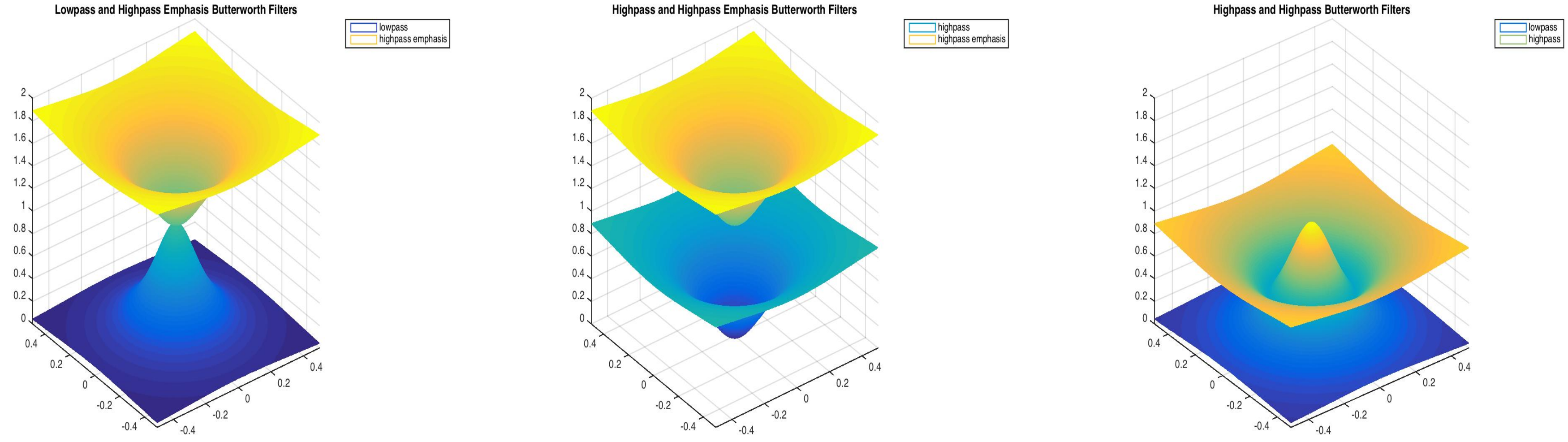
```
legend('lowpass','highpass emphasis');
title('Lowpass and Highpass Emphasis Butterworth Filters');
print Lowpass_Highpass_Emphasis_Butterworth_Filter.jpg -djpeg

figure
mesh(U,V,Hhigh);
hold on
mesh(U,V,Hemphasis);
hold off
axis([-1/2 1/2-1/dim1 -1/2 1/2-1/dim2 0 2]);
alpha(0.5);
view(-37.5,30.0);
legend('highpass','highpass emphasis');
title('Highpass and Highpass Emphasis Butterworth Filters');
print Highpass_Highpass_Emphasis_Butterworth_Filter.jpg -djpeg

figure
mesh(U,V,Hlow);
hold on
mesh(U,V,Hhigh);
hold off
axis([-1/2 1/2-1/dim1 -1/2 1/2-1/dim2 0 2]);
alpha(0.5);
view(-37.5,30.0);
legend('lowpass','highpass');
title('Highpass and Highpass Butterworth Filters');
print Highpass_Highpass_Butterworth_Filter.jpg -djpeg
```

- The MatLab code for the 2D Butterworth lowpass, highpass and highpass emphasis filters. They are very similar to the 1D filters and are in files **L21butterworth_lowpass2D.m**, **L21butterworth_lowpass2D.m** and **L21butterworth_lowpass2D.m**.

- Following this code we show the 3D mesh plots of these filters.

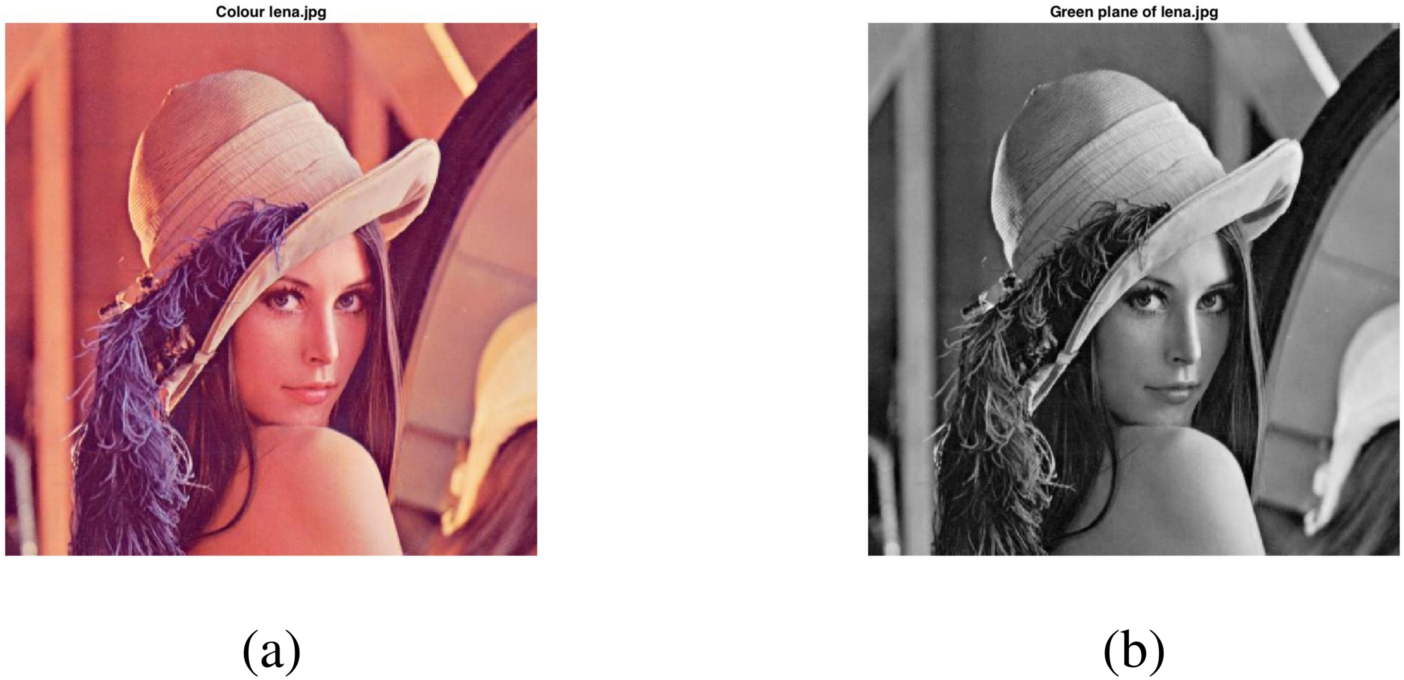
```
function [H]=L21butterworth_lowpass2D(D0,n,u,v)
% H is centered as u,v are centered
% D(u,v) is sqrt(u^2+v^2)
n=2*n;
[U,V]=meshgrid(u,v);
D=sqrt(U.^2+V.^2);
H=1./(1+(D./D0).^n);
end

function [H]=L21butterworth_highpass2D(D0,n,u,v)
% H is centered as u is centered
% D(u,v) is sqrt(u^2+v^2)
n=2*n;
[U,V]=meshgrid(u,v);
D=sqrt(U.^2+V.^2);
```

```
H=1./(1+(D0./D).^n);
end

function [H]=L21butterworth_highpass_emphasis2D(D0,n,u,v,emphasis_factor)
% H is centered as u is centered
% D(u,v) is sqrt(u^2+v^2)
n=2*n;
[U,V]=meshgrid(u,v);
D=sqrt(U.^2+V.^2);
H=emphasis_factor+1./(1+(D0./D).^n);
end
```

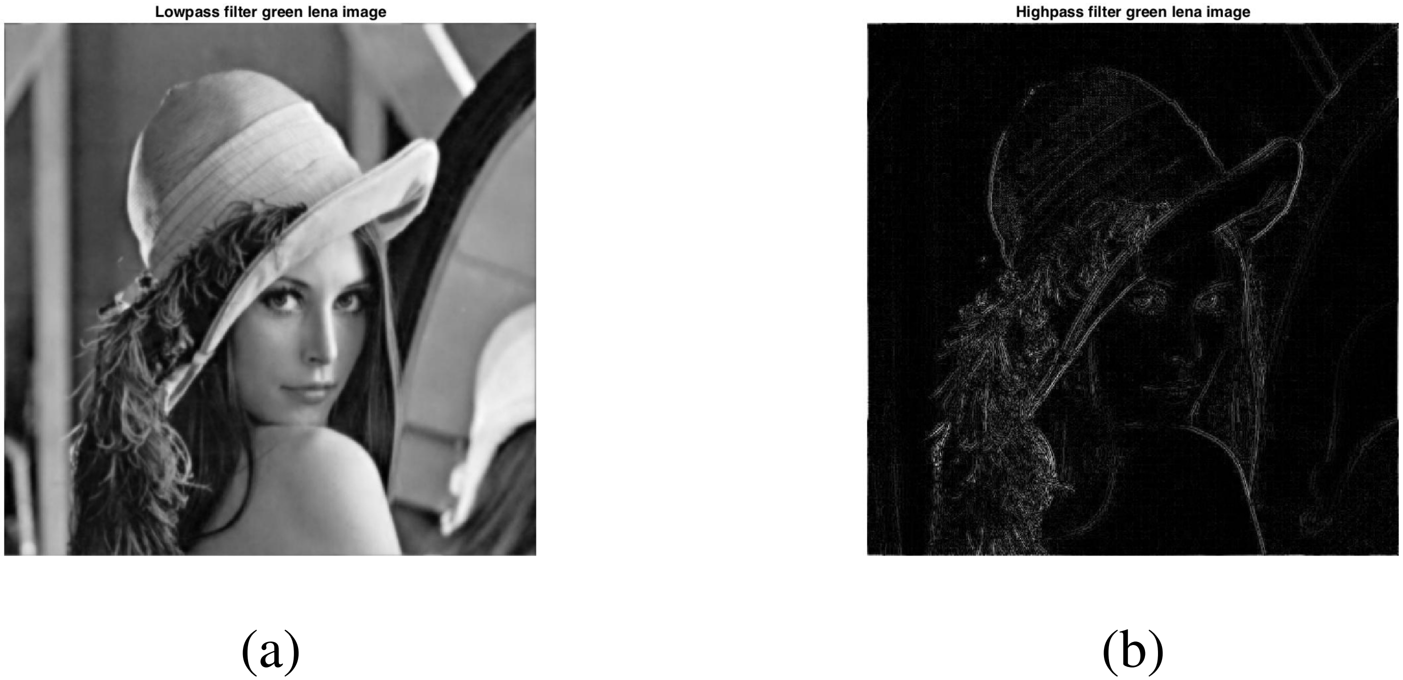


(a) (b) (c)

3D plots of (a) the lowpass Butterworth 2D filter and the highpass emphasis Butterworth 2D filter, (b) the highpass Butterworth 2D filter and the highpass emphasis Butterworth 2D filter and (c) the lowpass Butterworth 2D filter and the highpass Butterworth 2D filte. In (c) notice how the lowpass and high pass filters obscure each other as they overlap. The lowpass filter has a min of 0.0431 and a max of 1.0, the highpass filter has a min of 0 and a max of 0.8889 and the highpass mphasis filter has a min of 1 and a max of 1.889.



(a) The Colour lena image and (b) the green plane of the lena image.



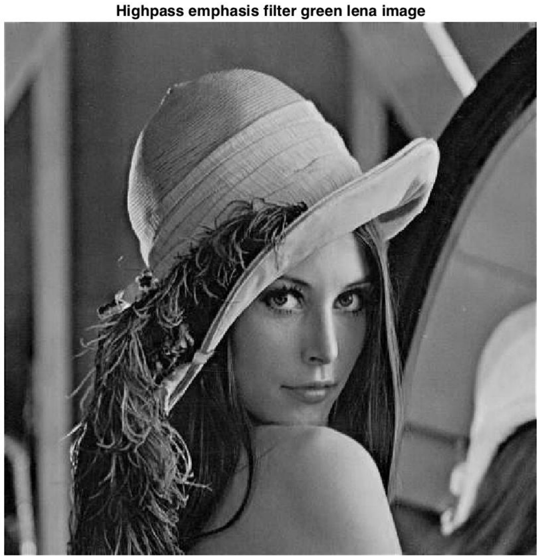
Lowpass filter green lena image

Highpass filter green lena image

(a)

(b)

(a) The lowpass green lena image and (b) the highpass green lena image. The highpass image looks somewhat like an edge image.



The highpass emphasis green lena image.

2D Butterworth Filtering the Colour Lena Image

- To filter the colout lena image, we compuute the YIQ representatiom of the colour image (NTSC standard) and filter the Y image while leaving the I and Q images alone. The MatLab code to do this is given below and in file **L21colour_butterworth2D.m**.

```
fRGB=imread('lena.jpg');
figure
imshow(fRGB,[]);
title('Colour lena.jpg');
print colour_lena.jpg -djpeg

fYIQ=rgb2ntsc(fRGB);

Y=squeeze(fYIQ(:,:,1));
figure
imshow(Y,[]);
title('Y lena image');
print Y_lena.jpg -djpeg

Yfft=fftshift(fft2(Y));
```

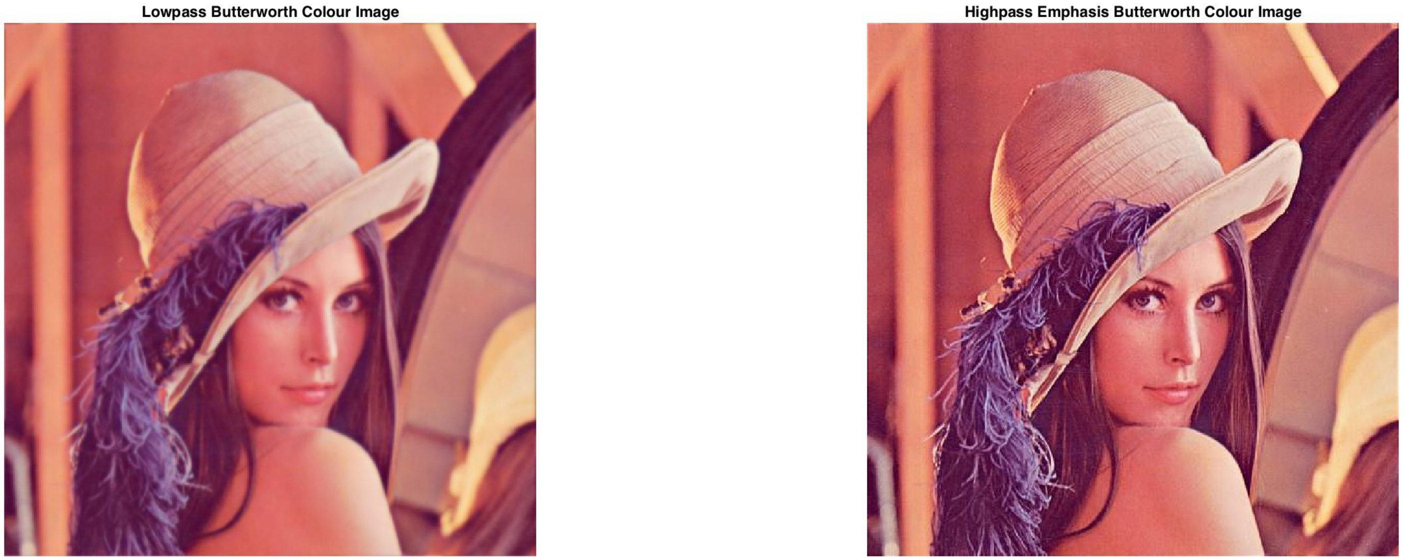
```
dim1=size(Y,1);
dim2=size(Y,2);
u=-1/2:1/dim1:1/2-1/dim1;
v=-1/2:1/dim2:1/2-1/dim2;
D0_low=0.15;
D0_high=0.25;
n=1; % 1st order butterworth filter
emphasis_factor=1.0; % highpass emphasis filtering

Hlow=L21butterworth_lowpass2D(D0_low,n,u,v);
Hhigh=L21butterworth_highpass2D(D0_high,n,u,v);
Hemphasis=L21butterworth_highpass_emphasis2D(D0_high,n,u,v,emphasis_factor);

lowpassfft=Hlow.*Yfft;
highpassfft=Hhigh.*Yfft;
emphasisfft=Hemphasis.*Yfft;

% replace Y image with lowpass image
% No need for ifftshift here (why?)
fYIQ(:,:,1)=abs((ifft2(lowpassfft)));
figure
fRGB=ntsc2rgb(fYIQ);
imshow(fRGB,[]);
title(’Lowpass Butterworth Colour Image’);
print lowpass_butterworth_colour_image.jpg -djpeg

figure
```

```
imshow(squeeze(fYIQ(:,:,1)),[]);
title(’lowpass Y image’);
print Y_lowpass_lena.jpg -djpeg

% replace Y image with highpass image
fYIQ(:,:,1)=abs((ifft2(highpassfft)));
figure
fRGB=ntsc2rgb(fYIQ);
imshow(fRGB,[]);
title(’Highpass Butterworth Colour Image’);
print highpass_butterworth_colour_image.jpg -djpeg

figure
imshow(squeeze(fYIQ(:,:,1)),[]);
title(’highpass Y image’);
print Y_highpass_lena.jpg -djpeg

% replace Y image with highpass emhasis image
fYIQ(:,:,1)=abs((ifft2(emphasisfft)));
figure
fRGB=ntsc2rgb(fYIQ);
imshow(fRGB,[]);
title(’Highpass Emphasis Butterworth Colour Image’);
print highpass_emphasis_butterworth_colour_image.jpg -djpeg

figure
imshow(squeeze(fYIQ(:,:,1)),[]);
title(’highpass Y image’);
```

```
print Y_highpass_emphasis_lena.jpg -djpeg
```

(a) (b)

(a) The Y lena image and (b) the lowpass colour colour image.


Lowpass Butterworth Colour Image
Highpass Emphasis Butterworth Colour Image

(a) (b)

(a) Butterworth highpass colour lena image and (b) Butterworth highpass emphasis lena image.