Lab 1 – Basic MATLAB

1. Audio (use flute22.wav and music.mp3)

(a) Write a MATLAB function named `audio2bin` that reads signal samples from an audio file (such as .wav or .mp3) and writes the signal samples to a raw binary file.

```
function [x,fs] = audio2bin(infile,outfile)
if(nargin == 1)
    outfile = [infile(1:max(strfind(infile,'.'))),'bin'];
end
fprintf('Input  file = %s\n',infile);
fprintf('Output file = %s\n',outfile);

[x,fs]=audioread(infile);

fid = fopen(outfile,'wb'); % write binary
fwrite(fid,[1 size(x,2) size(x,1) fs 0],'int'); % audio header
x = x.'; % transpose
fwrite(fid,x(:),'float'); % vectorize then write data
fclose(fid);

return;
```

(b) Write a MATLAB function named `bin2audio` that reads signal samples from a raw binary file and writes the signal samples to an audio file (such as .wav or .mp3).

```
function [x,fs] = bin2audio(infile,outfile)
if(nargin == 1)
    outfile = [infile(1:max(strfind(infile,'.'))),'wav'];
end
fprintf('Input  file = %s\n',infile);
fprintf('Output file = %s\n',outfile);

fid = fopen(infile,'rb'); % read binary
ndim  = fread(fid,1,'int'); % read number of dimensions
nchan = fread(fid,1,'int'); % read number of channels
dim0  = fread(fid,1,'int'); % read first dimension
dim1  = fread(fid,1,'int'); % read second dimension
dim2  = fread(fid,1,'int'); % read third dimension
[x,cnt]=fread(fid,inf,'float'); % read data
fclose(fid);

x = reshape(x,nchan,dim0).'; % reshape audio
fs = dim1;

audiowrite(outfile,x,fs);

return;
```

(c) Use the `sound` or `soundsc` MATLAB commands to play an audio signal to the speaker.

flute22.wav

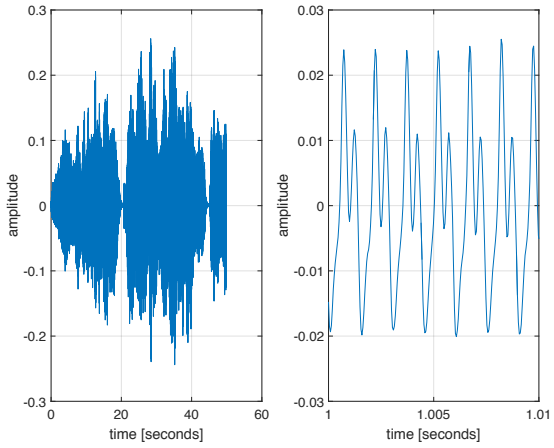
```
>> audio2bin('flute22.wav');
>> bin2audio('flute22.bin');
>> [x,fs]=audioread('flute22.wav');
>> sound(x,fs)
>> soundsc(x(1:5*fs,:),fs)
```

music.mp3

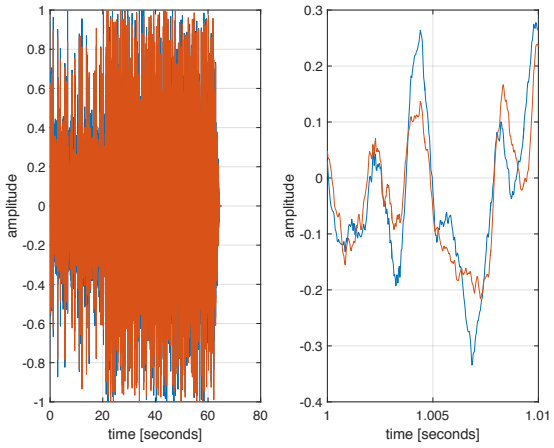
```
>> audio2bin('music.mp3');
>> bin2audio('music.bin');
>> [x,fs]=audioread('music.wav');
>> sound(x,fs)
>> soundsc(x(1:5*fs,:),fs)
```

(d) Make a subplot plot of the audio signal using a true time-scaled x-axis (set the x-axis limits to [1.0 1.01] seconds)

flute22.wav



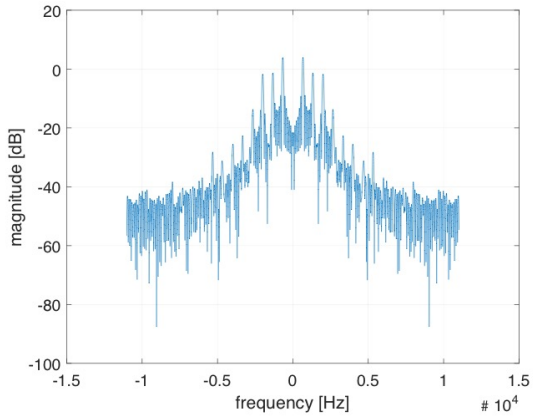
music.mp3




```
[x,sample_rate] = audioread('audiofile.ext'); % change audiofile.ext to audio file
time = [0:length(x)-1]/sample_rate ; % time [seconds]

subplot(1,2,1)
plot(time,x); % plot with accurately scaled time axis
xlabel('time [seconds]','FontSize' ,18);
ylabel('amplitude','FontSize',18);
set(gca,'FontSize',16)
grid on;

subplot(1,2,2);
plot(time,x); % plot with accurately scaled time axis
% adjust the view
t1 = 1.00; % seconds
t2 = 1.01; % seconds
xlim([t1 t2]); % view from t1 to t2 seconds
xlabel('time [seconds]','FontSize',18);
ylabel('amplitude','FontSize',18);
set(gca,'FontSize',16)
grid on;

orient landscape;
print -dpdf time_plot.pdf
```

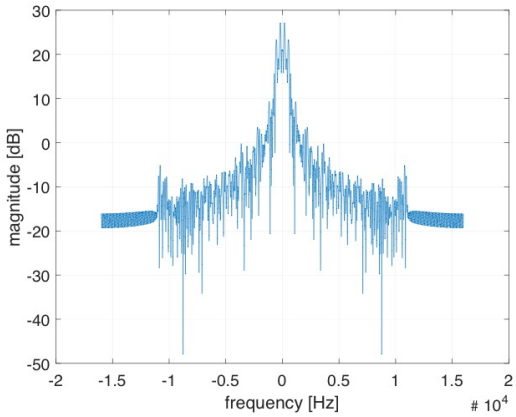
(e) Make a subplot plot of the spectrum (FFT) of the audio signal data from 1 to 1.01 seconds and use a true frequency-scaled x-axis (what is the frequency of the highest peak?)

flute22.wav



[frequency,MAX_magnitude] = [667.5 Hz,3.89 dB]

music.mp3

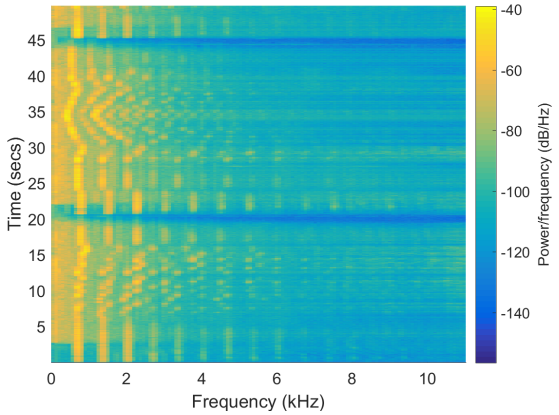


[frequency,MAX_magnitude] = [179.7 Hz,27.17 dB]

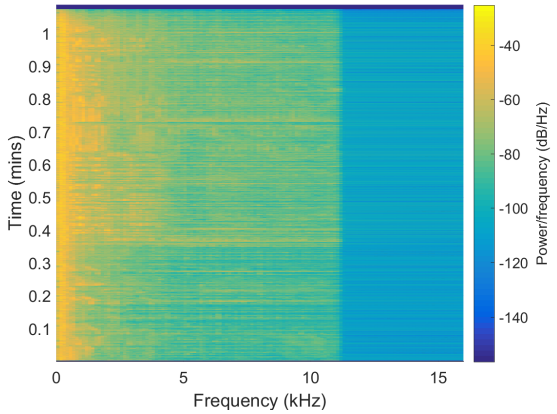
```
[x,sample_rate] = audioread('audiofile.ext'); % change audiofile.ext to audio file
t1 = 1.00; % seconds
t2 = 1.01; % seconds
i1 = round(t1*sample_rate); % convert time to index
i2 = round(t2*sample_rate); % convert time to index

nfft = 2^12; % FFT size
freq = ([0:nfft-1]/nfft-0.5)*sample_rate ; % frequency [Hz]
X = fft(x(i1:i2),nfft); % compute the discrete - Fourier transform
plot(freq,20*log10(abs(fftshift(X))));
% plot with accurately scaled frequency axis

xlabel('frequency [Hz]','FontSize',18);
ylabel('magnitude [dB]','FontSize',18);
set(gca,'FontSize',16)
grid on;
print -dpdf frequency_plot.pdf
```

(f) In a subplot show the spectrogram of the signal.

flute22.wav



music.mp3




```
[x,sample_rate] = audioread('music.mp3'); % change audiofile.ext to audio file

nfft = 2^8; % FFT size
overlap = round(0.8*nfft);
window = hamming(nfft);
y = (x(:,1)+x(:,2))/2; % use y if audiofile.ext is stereo
spectrogram(y,window,overlap,nfft,sample_rate); % x = mono, y = stereo

set(gca,'FontSize',16);
grid on;
print -dpdf spectrogram_plot.pdf
```

(g) Write an oscilloscope script to visualize the audio signal over a 10 ms window (handle graphics).

```
[x,fs] = audioread('music.mp3'); % change audiofile.ext to audio file
t = [0:length(x)-1]/fs; % time [seconds]
%plot(t,x); % this plots whole waveform, gives big picture, but cannot see detail

win_sec = 0.01; % window length [seconds]
win_sam = round(win_sec*fs); % window length [samples]
step_sec = 0.0002; % step length [seconds]
step_sam = round(step_sec*fs); % step length [samples]

han = plot(t(1:win_sam),x(1:win_sam)); drawnow;
ylim(0.1*[-1 1]);
for i=win_sam:step_sam:length (x)
ind = [i-win_sam+1:i];
set(han,'XData',t(ind),'YData',x(ind));
xlim(t(ind([1,end]))); drawnow ;
% pause; % if this runs too fast, include a pause
end
```

2. Image (use liftingbody.png and coloredchips.png)

(a) Write a MATLAB function named `image2bin` that reads signal samples from an image file and writes the signal samples to a raw binary file.

```
function x = image2bin(infile,outfile)
if(nargin == 1)
    outfile = [infile(1:max(strfind(infile,'.'))),'bin'];
end
fprintf('Input  file = %s\n',infile);
fprintf('Output file = %s\n',outfile);

x=imread(infile);

fid = fopen(outfile,'wb'); % write binary
fwrite(fid,[2 size(x,3) size(x,1) size(x,2) 0],'int'); % image header
x = permute(x,[2 1 3]); % permutate
x = reshape(x,[size(x,1)*size(x,2) size(x,3)]).'; % reshape & transpose
fwrite(fid,x(:),'float'); % vectorize then write data
fclose(fid);

return;
```

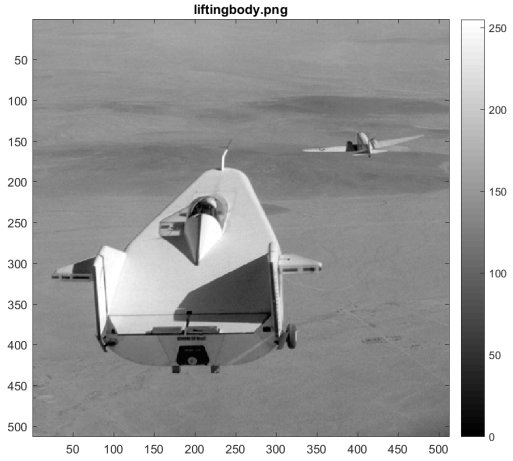
(b) Write a MATLAB function named `bin2image` that reads signal samples from a raw binary file and writes the signal samples to an image file.

```
function x = bin2image(infile,outfile)
if(nargin == 1)
    outfile = [infile(1:max(strfind(infile,'.'))),'png'];
end
fprintf('Input  file = %s\n',infile);
fprintf('Output file = %s\n',outfile);

fid = fopen(infile,'rb'); % read binary
ndim  = fread(fid,1,'int'); % read number of dimensions
nchan = fread(fid,1,'int'); % read number of channels
dim0  = fread(fid,1,'int'); % read first dimension
dim1  = fread(fid,1,'int'); % read second dimension
dim2  = fread(fid,1,'int'); % read third dimension
[x,cnt]=fread(fid,inf,'float'); % read data
fclose(fid);

if(nchan == 1) % grayscale image
    x = reshape(x,[dim1 dim0]).'; % reshape image
    x = double(x)/255; % convert to K percentage
end
if(nchan == 3) % color image
    R = x(1:3:end); G = x(2:3:end); B = x(3:3:end); % isolate channels
    R = reshape(R,[dim1 dim0]).'; % reshape RED channel
    G = reshape(G,[dim1 dim0]).'; % reshape GREEN channel
    B = reshape(B,[dim1 dim0]).'; % reshape BLUE channel
    x = cat(3,R,G,B); % concatenate along the third dimension
    x = double(x)/255; % convert to RGB percentages
end
```

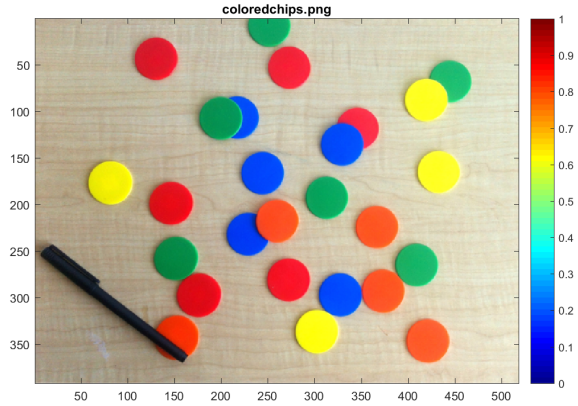
```
imwrite(x,outfile,'PNG');

return;
```

(c) Make a plot of the image (pixels should be square).

liftingbody.png



coloredchips.png




```
infile = 'imagefile.ext'; % change imagefile.ext to image file
x = imread(infile);
imagesc(x); % show scaled image
axis image; % make the pixels square
title(infile);
colormap jet; % gray = grayscale image, jet = color image
colorbar; % show the colorbar
print -dpng image_plot.png
```

3. Video (use xylophone.mp4)

(a) Write a MATLAB function named `video2bin` that reads signal samples from a video file and writes the signal samples to a raw binary file (only one video frame in memory at a time).

```
function x = video2bin(infile,outfile)
if(nargin == 1)
    outfile = [infile(1:max(strfind(infile,'.'))),'bin'];
end
fprintf('Input  file = %s\n',infile);
fprintf('Output file = %s\n',outfile);

obj=VideoReader(infile,'tag','myreader1');
x = read(obj); % read in all the frames

fid = fopen(outfile,'wb'); % write binary
fwrite(fid,[3 size(x,3) size(x,1) size(x,2) size(x,4)],'int'); % video header
nframes = obj.NumberOfFrames;
for i=1:nframes
    y = read(obj,i); % read video frame
    y = permute(y,[2 1 3]); % permutate
    y = reshape(y,[size(y,1)*size(y,2) size(y,3)]).'; % reshape & transpose
```

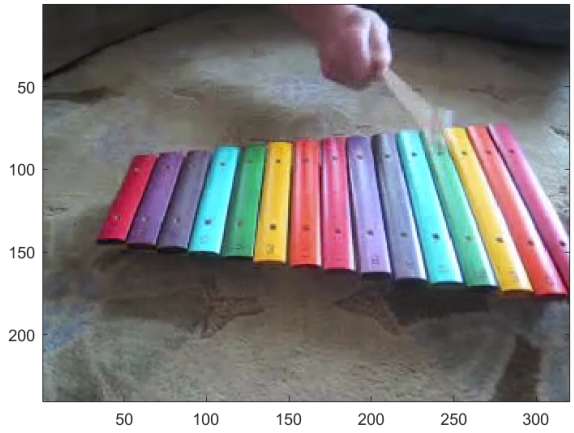
```
    fwrite(fid,y(:),'float'); % vectorize then write data
end
fclose(fid);

return;
```

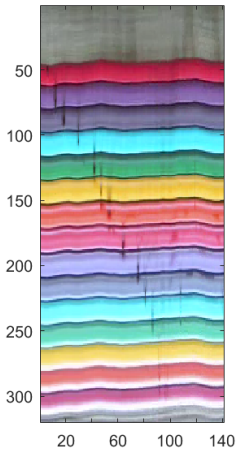
(b) Write a MATLAB function named `bin2video` that reads signal samples from a raw binary file and writes the signal samples to a video file (only one video frame in memory at a time).

```
function x = bin2video(infile,outfile)
if(nargin == 1)
    outfile = [infile(1:max(strfind(infile,'.'))),'mp4'];
end
fprintf('Input  file = %s\n',infile);
fprintf('Output file = %s\n',outfile);

fid = fopen(infile,'rb'); % read binary
ndim  = fread(fid,1,'int'); % read number of dimensions
nchan = fread(fid,1,'int'); % read number of channels
dim0  = fread(fid,1,'int'); % read first dimension
dim1  = fread(fid,1,'int'); % read second dimension
dim2  = fread(fid,1,'int'); % read third dimension
[x,cnt]=fread(fid,inf,'float'); % read data
fclose(fid);

obj = VideoWriter(outfile,'MPEG-4'); % create video file
open(obj); % open video file
j = 1; % beginning of frame
k = dim0*dim1*nchan-1; % data length of frame
for i=1:dim2
    y = x(j:j+k); % isolate image from video
    if(nchan == 1) % grayscale image
        y = reshape(y,[dim1 dim0]).'; % reshape image
        y = double(y)/255; % convert to K percentage
    end
    if(nchan == 3) % color image
        R = y(1:3:end); G = y(2:3:end); B = y(3:3:end); % isolate channels
        R = reshape(R,[dim1 dim0]).'; % reshape RED channel
        G = reshape(G,[dim1 dim0]).'; % reshape GREEN channel
        B = reshape(B,[dim1 dim0]).'; % reshape BLUE channel
        y = cat(3,R,G,B); % concatenate along the third dimension
        y = double(y)/255; % convert to RGB percentages
    end
    writeVideo(obj,y); % write the video frame
    j = j+k+1; % move to next frame
end
close(obj); % close video file

return;
```

(c) Write a movie player script to visualize the frames of video (handle graphics).

xylophone.mp4






```
% load the frames of a video one at a time and display each one as an image

obj = VideoReader('imagefile.ext','tag','myreader1');
% change imagefile.ext to video file

z = read(obj); % color video
sz = size(z)
r = round(sz(1)/2);
y = z(r,:,:,:);
sy = size(y);
x = squeeze(y); % squeeze removes singleton dimensions
sx = size(x);
w = permute(x,[1 3 2]); % permutes the dimensions
    % need RGB in 3rd dimension of image (see also ipermute)
sw = size(w);
imagesc(w); axis image;

nframes = obj.NumberOfFrames ;
for i=1:nframes
    yi = read(obj,i); % read the next video frame
    if(i==1)
        imhan = image(yi); axis image; drawnow;
    else
        set(imhan,'CData',yi); drawnow ;
    end
end
```