EECS 3451 Lab1

Table of Contents

AUTHORS

- Jonathan Baldwin (212095691)
- Mark Savin (212921128)
- Sarwat Shaheen (214677322)

playsound function

This is a blocking version of the built-in sound function.

```
function playsound(y, Fs)
    % On Linux, refuses to play at anything other than 44100 Hz
    if isunix()
        y = resample(y,44100,Fs);
        Fs = 44100;
    end
```

```
    player = audioplayer(y,Fs);
    playblocking(player);
end
```

mydouble function

This is the double function from the previous lab.

```
function out = mydouble(in)
 tmp = 1:.5:length(in);
 out = (in(floor(tmp)) + in(ceil(tmp)))/2;
end
```

Problem 1

As the frequency varies, so does the pitch proportionally. Doubling the frequency results in a similar sound but with a higher pitch. When we lower the frequency below about 70Hz, it becomes difficult to hear without also increasing the amplitude. Below around 18Hz, it becomes inaudible even with a high amplitude. Likewise, above around 20000Hz, it becomes inaudible.

As the amplitude varies, the perceived loudness varies significantly. At around 8V, it becomes painful to hear, and below around 80mV it becomes hard to discern.

Varying offset and phase does not seem to affect the sound.

Changing the shape of the waveform changes the "flavour" of the sound - sine waves give a sharp sound, square waves a sound reminicent of 8-bit chiptunes, ramp sounds similar to sine.

Problem 2

- The lower the sampling frequency of the signal, the more muffled it sounds.
- The higher the sampling frequency of the signal, the more clear it sounds. Oversampling beyond that of the input signal has no effect, however.

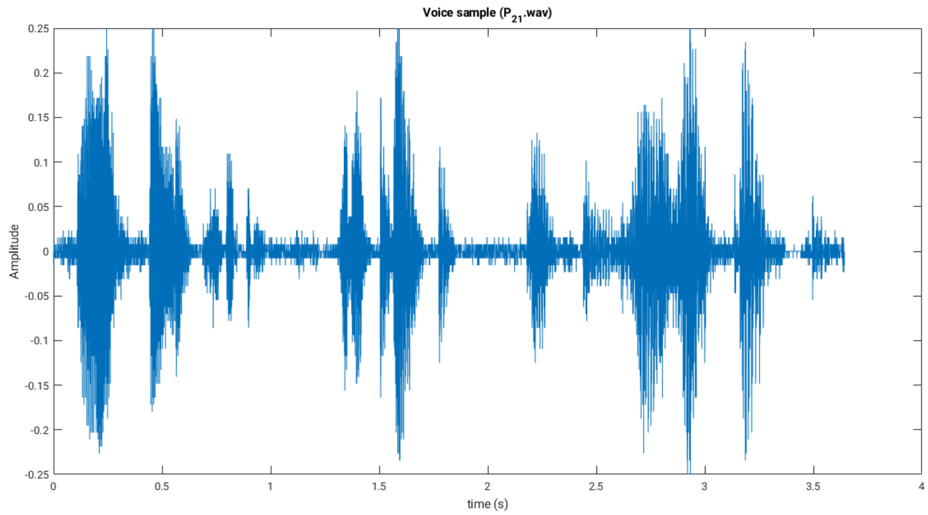
```
[y,Fs] = audioread('P_2_1.wav');
plot([1:size(y)]./Fs,y);
title('Voice sample (P_2_1.wav)');
xlabel('time (s)');
ylabel('Amplitude');

fprintf('Playing at input frequency (%d)\n', Fs);
playsound(y,Fs);

for Fs2=[2000,4000,6000,12000]
   y2=resample(y,Fs2,Fs);
   fprintf('Playing resampled to %d Hz\n', Fs2);
   playsound(y2,Fs2);
```

```
end
```

```
Playing at input frequency (8000)
Playing resampled to 2000 Hz
Playing resampled to 4000 Hz
Playing resampled to 6000 Hz
Playing resampled to 12000 Hz
```



Problem 3a - 100 Hz triangle wave

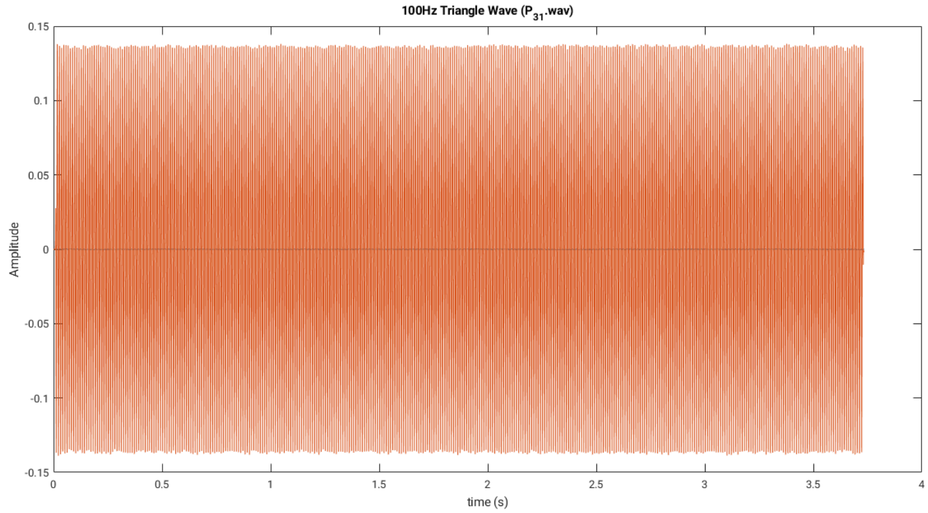
Unlike in Problem 2, changing the sampling rate doesn't seem to affect the quality of the sound.

```
[y,Fs] = audioread('P_3_1.wav');
plot([1:size(y)]./Fs,y);
title('100Hz Triangle Wave (P_3_1.wav)');
xlabel('time (s)');
ylabel('Amplitude');

fprintf('Playing at input frequency (%d)\n', Fs);
playsound(y,Fs);

for Fs2=[2000,4000,6000,12000]
   y2=resample(y,Fs2,Fs);
   fprintf('Playing resampled to %d Hz\n', Fs2);
   playsound(y2,Fs2);
end
```

```
Playing at input frequency (48000)
Playing resampled to 2000 Hz
Playing resampled to 4000 Hz
Playing resampled to 6000 Hz
Playing resampled to 12000 Hz
```



Problem 3b - voice sample

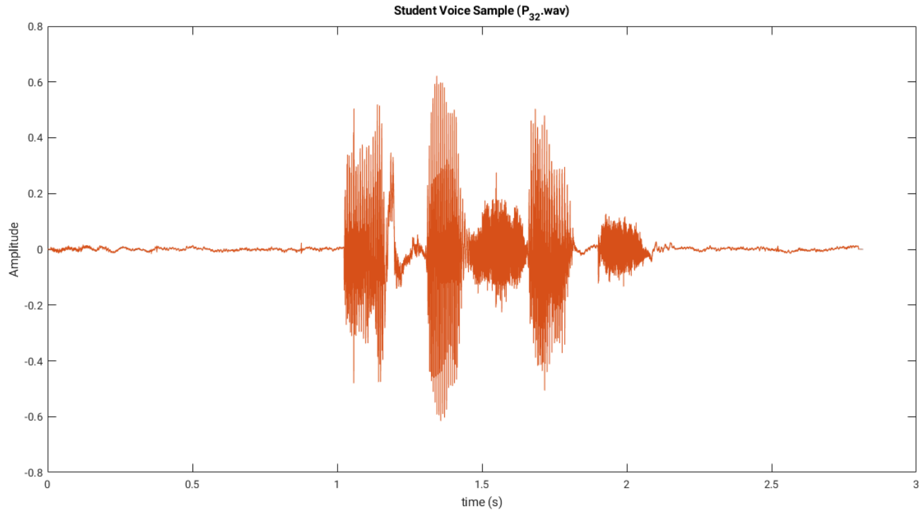
This time, much like in Problem 2, the sampling rate affects the quality of the sound greatly.

```
[y,Fs] = audioread('P_3_2.wav');
plot([1:size(y)]./Fs,y);
title('Student Voice Sample (P_3_2.wav)');
xlabel('time (s)');
ylabel('Amplitude');

fprintf('Playing at input frequency (%d)\n', Fs);
playsound(y,Fs);

for Fs2=[2000,4000,6000,12000]
   y2=resample(y,Fs2,Fs);
   fprintf('Playing resampled to %d Hz\n', Fs2);
   playsound(y2,Fs2);
end
```

```
Playing at input frequency (48000)
Playing resampled to 2000 Hz
Playing resampled to 4000 Hz
Playing resampled to 6000 Hz
Playing resampled to 12000 Hz
```



Problem 4a - double

Running the signal through double doesn't seem to affect the sound at all.

```
[y,Fs] = audioread('P_3_2.wav');

y2=mydouble(y);
fprintf('Playing at input frequency %d Hz, doubled to %d\n', Fs,
 Fs*2);
playsound(y2,Fs*2);

for Fs2=[2000,4000,6000,12000]
   y2=resample(y,Fs2,Fs);
   y3=mydouble(y2);
   fprintf('Playing resampled to %d Hz, then doubled to %d\n', Fs2,
 Fs2*2);
   playsound(y3,Fs2*2);
end
```

```
Playing at input frequency 48000 Hz, doubled to 96000
Playing resampled to 2000 Hz, then doubled to 4000
Playing resampled to 4000 Hz, then doubled to 8000
Playing resampled to 6000 Hz, then doubled to 12000
Playing resampled to 12000 Hz, then doubled to 24000
```

Problem 4b - fliplr

Running the signal through fliplr flips the left and right channels. As our voice sample is centered, the sound output doesn't perceptibly change.

```
[y,Fs] = audioread('P_3_2.wav');

y2=fliplr(y);
fprintf('Playing at input frequency %d Hz\n', Fs);
```

```
playsound(y2,Fs);
```

Playing at input frequency 48000 Hz

Problem 4b - flipud

Running the signal through flipud reverses the sound. It sounds like something out of the Exorcist.

```
[y,Fs] = audioread('P_3_2.wav');

y2=flipud(y);
fprintf('Playing at input frequency %d Hz\n', Fs);
playsound(y2,Fs);
```

Playing at input frequency 48000 Hz

Problem 5

```
function answer = message_OR_power(t)
  N = length(t);
  time_avg = sum(t) / N;
  squared_avg = sum(t.^2) / N;

  if (squared_avg ~= 0 && abs(time_avg) < 0.01)
      answer = 0; %power
  else
      answer = 1;  %message
  end
end
```

Problem 5a)

```
t = 0:0.01:10;
fprintf('sin(10*t) is a ');
if message_OR_power(sin(10*t)) == 1
    disp('message signal');
else
    disp('power signal');
end
```

*sin(10*t) is a power signal*

Problem 5b)

```
t = 0:0.01:10;
fprintf('cos(2*pi*t) is a ');
if message_OR_power(cos(2*pi*t)) == 1
    disp('message signal');
else
    disp('power signal');
```

```
end
```

*cos(2*pi*t) is a power signal*

Problem 5c)

```
t = 0:0.01:10;
fprintf('4.*exp(-t/4).*rectangularPulse((t-4)/3) is a ');
if message_OR_power(4.*exp(-t/4).*rectangularPulse((t-4)/3)) == 1
    disp('message signal');
else
    disp('power signal');
end
```

*4.*exp(-t/4).*rectangularPulse((t-4)/3) is a message signal*

Problem 5d)

```
t = 0:0.01:10;
fprintf('4.*exp(-t/4).*heaviside(t-1).*sign(t-2) is a ');
if message_OR_power(4.*exp(-t/4).*heaviside(t-1).*sign(t-2)) == 1
    disp('message signal');
else
    disp('power signal');
end
```

*4.*exp(-t/4).*heaviside(t-1).*sign(t-2) is a message signal*

Problem 6a

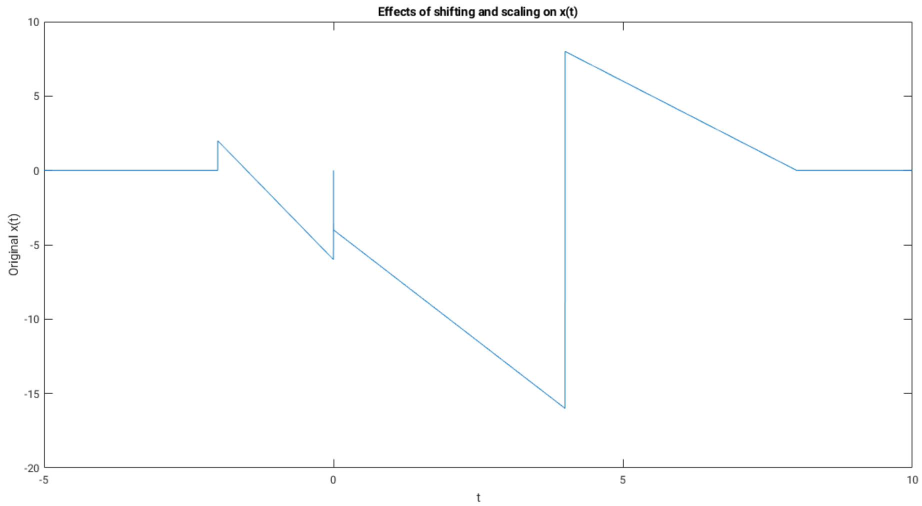
```
%M file for the function x(t) as defined with the respective time
 ranges
% 't'
function y = problem6_x(t)
 % Calculate the functional variation for each range of time, t
 x1 = (-4*t) - 6;
 x2 = -4 - (3*t);
 x3 = 16 - (2*t);

 % Splice together the different functional variations in
 % their respective ranges of validity
 y = x1.*(-2<t & t<0) + x2.*(0<t & t<4) + x3.*(4<t & t<8);
end
```

Problem 6a i

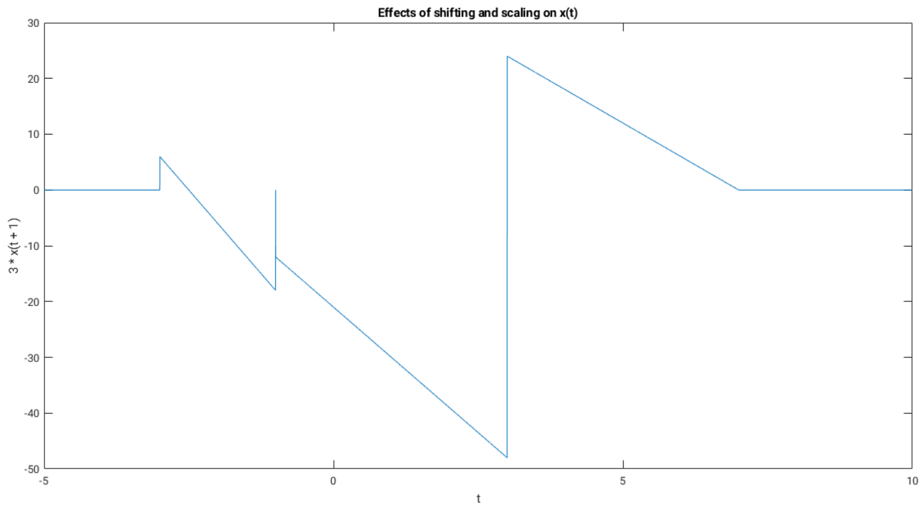
```
t = -5:0.001:10;
plot(t, problem6_x(t));
title("Effects of shifting and scaling on x(t)");
xlabel("t");
```

```
ylabel("Original x(t)");
```



Problem 6a ii

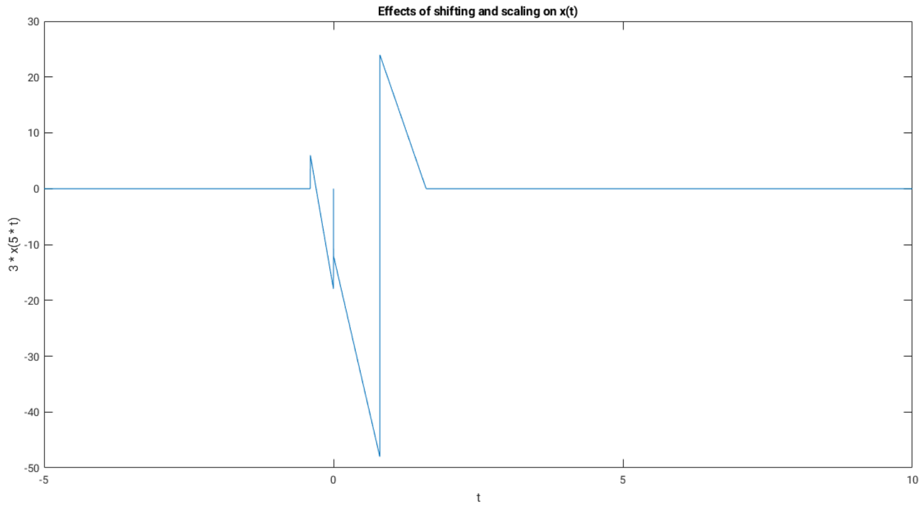
```
t = -5:0.001:10;
plot(t, 3*problem6_x(t+1));
title("Effects of shifting and scaling on x(t)");
xlabel("t");
ylabel("3 * x(t + 1)");
```



Problem 6a iii

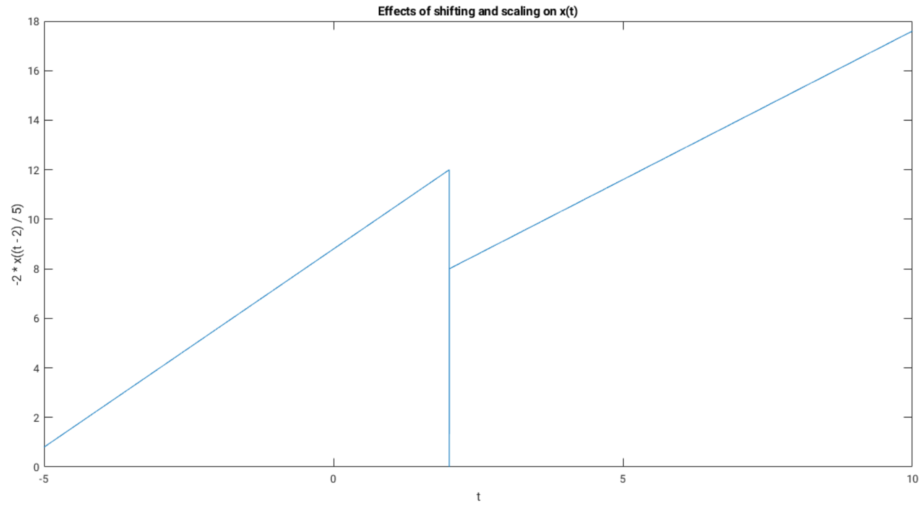
```
t = -5:0.001:10;
plot(t, 3*problem6_x(5*t));
title("Effects of shifting and scaling on x(t)");
xlabel("t");
```

```
ylabel("3 * x(5 * t)");
```



Problem 6a iv

```
t = -5:0.001:10;
plot(t, -2*problem6_x((t-2)/5));
title("Effects of shifting and scaling on x(t)");
xlabel("t");
ylabel("-2 * x((t - 2) / 5)");
```



Problem 6b - Calculating derivatives using MATLAB 'diff' command

```
syms t; % Use the symbolic math toolbox
fprintf('The derivative of %s is:\n\t%s\n', ...
```

```
        'sin(2 * pi * t) * sign(t)', ...
        diff(sin(2 * pi * t) * sign(t)));
fprintf('The derivative of %s is:\n\t%s\n', ...
        'abs(cos(2 * pi * t))', ...
        diff(abs(cos(2 * pi * t))));
```

```
The derivative of sin(2 * pi * t) * sign(t) is:
 2*sin(2*t*pi)*dirac(t) + 2*pi*cos(2*t*pi)*sign(t)
The derivative of abs(cos(2 * pi * t)) is:
 -2*pi*sin(2*t*pi)*sign(cos(2*t*pi))
```

Problem 6c - Calculating integrals using MATLAB 'int' command

```
syms t; % Use the symbolic math toolbox
fprintf('The integral of %s is:\n\t%s\n', ...
        '(3 * t) * sin(2 * pi * t)', ...
        int((3 * t) * sin(2 * pi * t)));
fprintf('The integral of %s is:\n\t%s\n', ...
        '4 * exp(-18 * t)', ...
        int(4 * exp(-18 * t)));
```

```
The integral of (3 * t) * sin(2 * pi * t) is:
 (3*(sin(2*t*pi) - 2*t*pi*cos(2*t*pi)))/(4*pi^2)
The integral of 4 * exp(-18 * t) is:
 -(2*exp(-18*t))/9
```

Conclusion

In this lab we learned how to use MATLAB's ability to record and play sound, and work with WAVE files; how sampling rate affects the quality of a signal; how to determine if a signal is a message signal or a power signal using MATLAB; and how to use the Symbolic Math Toolbox to calculate derivatives and integrals.

Published with MATLAB® R2019a