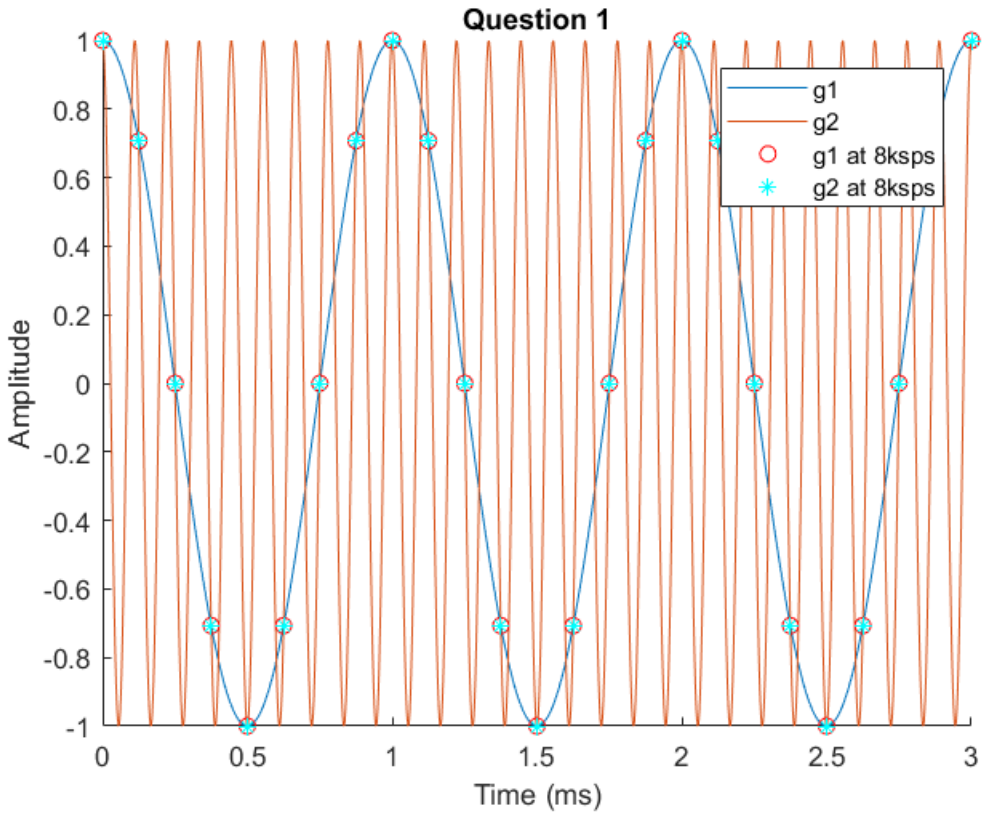
Table of Contents


```
%Ryan Plante
%DSP Homework 2
%2/6/2018
```

Question 1

```
%(A)
time = [0:1/1000000:3/1000]; %arbitrary large sampling rate
g1 = cos(2*pi*(1000)*time);
g2 = cos(2*pi*(9000)*time);
figure(1)
hold on
plot(time*1000, g1)
plot(time*1000, g2)
xlabel('Time (ms)');
ylabel('Amplitude');
title('Question 1');

%(B)
time = [0:1/8000:3/1000]; %sampling now at 8ksps
g18k = cos(2*pi*(1000)*time);
g28k = cos(2*pi*(9000)*time);
plot(time*1000, g18k, 'ro')
plot(time*1000, g28k, 'c*')
legend('g1', 'g2', 'g1 at 8ksps', 'g2 at 8ksps');
hold off
%(C)
%See paper
```



Question 2

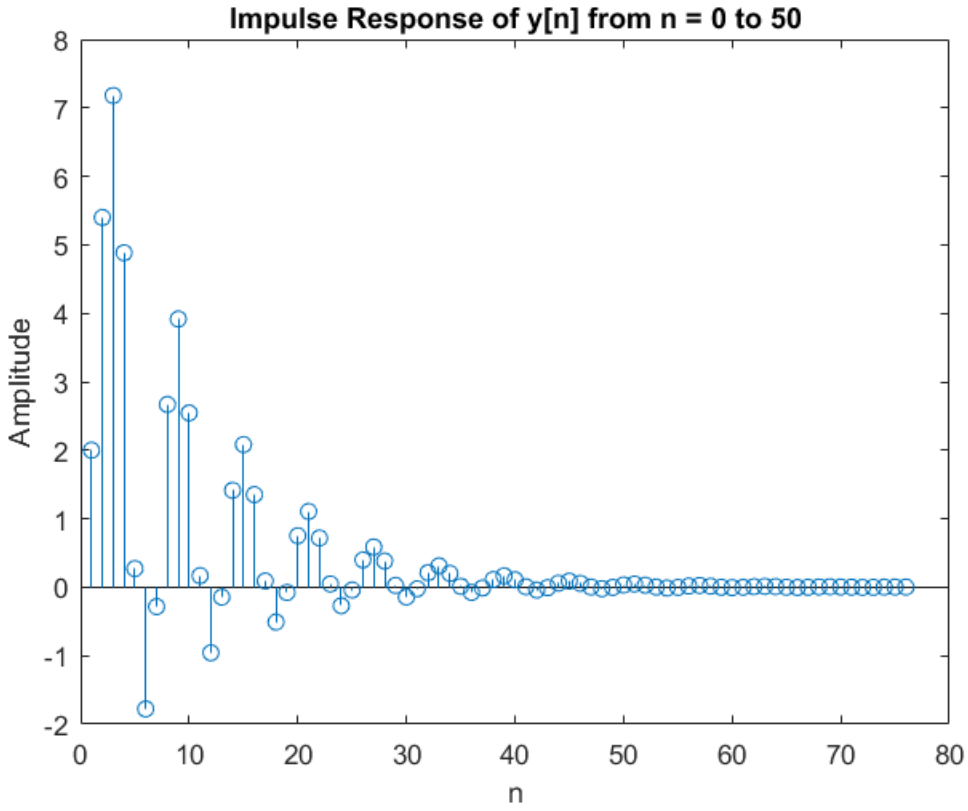
```
%See paper
```

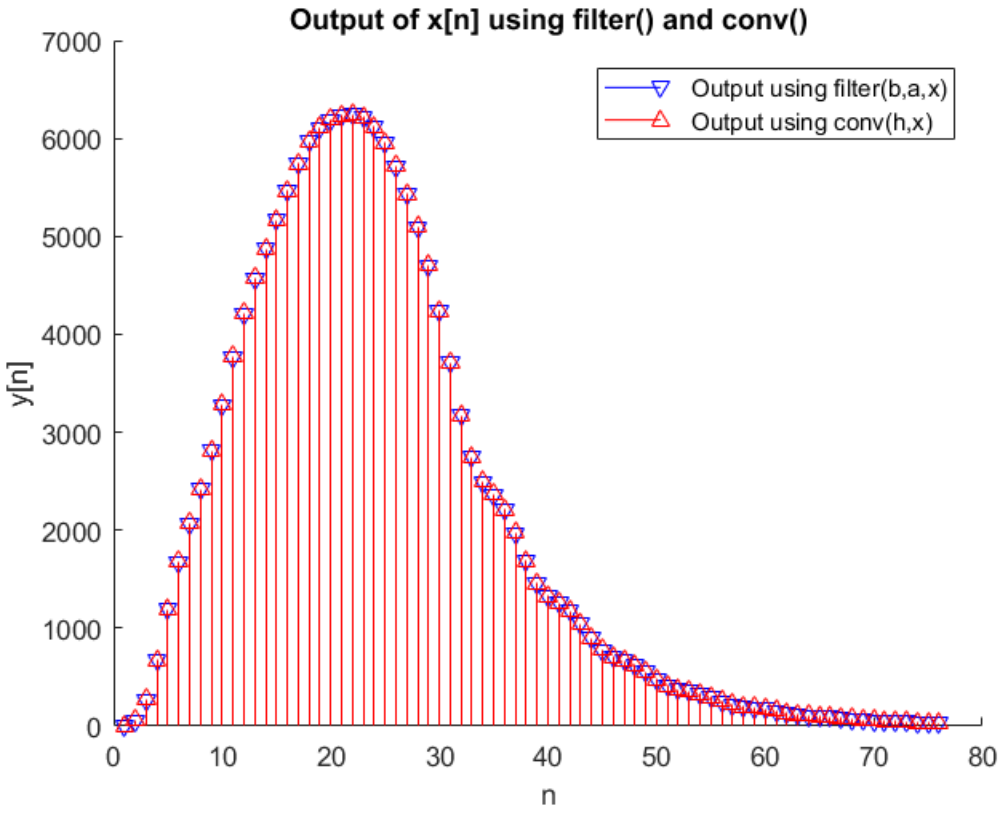
Question 3

```
%(A)
n = 0:75;
a = [1 -1.3 0.72 0.081 -0.3645];
b = [2 2.8 1.6 -0.4 -1.2];
h = impz(b, a, n);
figure(2)
stem(h)
xlabel('n');
ylabel('Amplitude');
title('Impulse Response of y[n] from n = 0 to 50');

%(B)
figure(3)
hold on
xn = [];
for n = 0:75
    if n <=30
        xn(n+1) = n*(30-n);
    else
```

```
        xn(n+1) = 0;
    end
end
z = filter(b, a, xn);
stem(z, '-vb')
title('Output of x[n] using filter() and conv()');
xlabel('n');
ylabel('y[n]');
%(C)

d = conv(h, xn);
d(77:end) = [];
stem(d, '-^r')
legend('Output using filter(b,a,x)', 'Output using conv(h,x)');
%(D)
%to compensate for matlab rounding errors we need to round each matrix
%before checking for equality
equalityCheck = isequal(round(d),round(z))
```

```
equalityCheck =

  logical

   1
```





Question 4

```
%(A)
%See paper

%(B)
%load in our audio file as specified in handout
[x_long,Fs] = audioread('guitar10min.ogg');
x = x_long(55*Fs:65*Fs,1);
x = x/max(abs(x));
obj1 = audioplayer(x,Fs);

%Find blocksize and number of blocks
%Probably a more efficient way to do this???
num = size(x);
num = num(1);
factors = factor(num); %very computationally expensive
blocksize = factors(3);
blocks = factors(1) * factors(2);
windowsize = 151;

%Reshape array for use with our running mean functions
blockedSignal = reshape(x,[blocksize, blocks])';

%Create our lowpass filter
```

```
lowpass = [];
s = init_running_mean(windowsize, blocksize);
for index = 1:blocks
[y, s] = calc_running_mean(blockedSignal(index, :), s);
lowpass = horzcat(lowpass, y);
end
obj2 = audioplayer(lowpass, Fs);

%Create our highpass filter
delay = (windowsize-1)/2;
delayMatrix = zeros(1, delay);
delayedSignal = horzcat(delayMatrix, x');
highpass = delayedSignal(1,1:end-delay) - lowpass;
obj3 = audioplayer(highpass, Fs);

%Create our audio mixer. Change the lowpass coefficient for more/less
 bass,
%change the highpass coefficient for more/less highs
mixer = 5*lowpass + 1*highpass;
mixer = mixer/max(abs(mixer)); %Normalize values to +/- 1
obj4 = audioplayer(mixer, Fs);
```

Question 5

```
%See paper
```

Published with MATLAB® R2017a

Ryan Plante
DSP
2/6/18

Problem 1

a.) see last

b.) see last

c.) Any frequency at $1k + 8kn$ will alias to the same discrete time sequence. Given that 17k and 25k will work.

Ryan Pierce
DSP
2/6/18

Problem 2

$y(n) - 0.35\,y(n-1) = x(n)$

Impulse Response?
Stable?

$y(n) = 0.35\,y(n-1) + x(n)$

$x[n] = \delta[n] = [\ldots 0, 0, \underset{\uparrow}{1}, 0, 0 \ldots]$

$h[0] = 0.35(0) + 1 = 1$

$h[1] = 0.35(1) + 0 = 0.35$

$h[2] = 0.35(0.35) + 0 = 0.1225$

$\vdots$

$h[n] = (0.35)^n h_n \quad n \geq 0$

This system is stable as for any value of n the system will return a finite value.

Problem 4

a.)

i. Impulse Response of $y_1[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n-k]$

can be defined as $h(n) = \begin{cases} \frac{1}{M} & n < M-1 \\ 0 & \geq M-1 \end{cases}$

ii. $F = \frac{F_s}{M} = \frac{44100}{151} = 292.05\ Hz$

iii. $D = \frac{151-1}{2} = 75\ samples$ $\quad \frac{75}{44100} = 1.7\ ms$

b.) see code

Problem 5

Impulse response of $y(n) = y(n-1) + \frac{1}{10}(x(n) - x(n-10))$

$x[n] = \delta[n] = [\ldots 0, 0, 1, 0, 0 \ldots]$

$h[0] = 0 + \frac{1}{10}(1 - 0) = \frac{1}{10}$

$h[1] = \frac{1}{10} + \frac{1}{10}(0 - 0) = \frac{1}{10}$

$\vdots$

$h[10] = \frac{1}{10} + \frac{1}{10}(0 - 1) = 0$

$$h[n] = \frac{1}{10}(u_n - u_n(n-10))$$