Jonathan W. Valvano April 4, 2014, 10:00 to 10:50am

(10) Question 1. Consider the following 60-Hz notch IIR filter. $Q = f_c/\Delta f = 60/80 = 0.75$

(5) Part a) B is mathematically the same as A. C is the same because it is an amplitude scale (multiply all x terms by the constant 5/8). You did not need to find H(z) for this problem

Filter A $Y(z) = X(z) + z^{-2} X(z) - 0.25 \bullet z^{-2} Y(z)$

$H(z) = (1 + z^{-2}) /(1 + 0.25 \bullet z^{-2})$

Filter C $Y(z) = 0.625 \bullet X(z) + 0.625 \bullet z^{-2} X(z) - 0.25 \bullet z^{-2} Y(z)$

$H(z) = 0.625 \bullet (1 + z^{-2}) /(1 + 0.25 \bullet z^{-2})$

(5) Part b) What is the DC gain of filter C? Two ways

Method 1) Use H(z) and look at |H(z)| at z=1

$H(z) = 0.625 \bullet (1 + 1^{-2}) /(1 + 0.25 \bullet 1^{-2}) = 1$

Method 2) Let x and y be constants and solve for y/x

$y = (5 \bullet (x + x) - 2 \bullet y)/8$

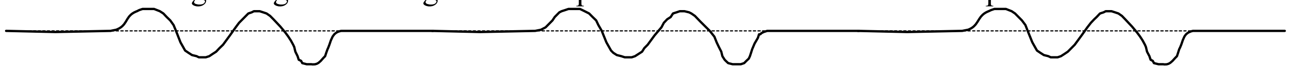
$8 \bullet y = 10 \bullet x - 2 \bullet y$

$10 \bullet y = 10 \bullet x$

$y/x = 1$

(15) Question 2. The goal is to study the frequency components of a signal using the FFT. The frequencies of interest are 0 to 50 Hz. The desired frequency resolution is 1 Hz. The SNR of the signal is 40 dB. The sampling rate is 512 Hz.

Part a) No, we do not need to window the data before applying the FFT. There will be no leakage, because the beginning and ending will line up to create a reasonable infinite periodic wave



The basic idea with spectral leakage is often the beginning and ending do not line up, creating a discontinuity in the infinite periodic wave. The waveform drawn in the problem is exactly like what the typical data would look it if a window were to be applied.

Part b) N = 512, so the frequency resolution will be $\Delta f = f_s/N = 1\text{Hz}$

Part c) What is the smallest number of ADC bits you could you choose and still capture all the information in the signal? Why?

$40 \text{ db} \leq 20 \log_{10} 2^n$, where n is the number of ADC bits

$2 \leq \log_{10} 2^n$,

$100 \leq 2^n$, so n = 7

(20) Question 3. Consider this microphone circuit. The opamp is powered with +3.3V and ground. The basic idea is to identify the two filters: C1, R2, R3 create a high pass filter

$f_c = 1/(2\pi*1\mu F*11k) = 14$ Hz

and C3, R6 create a low pass filter (very bad filter, gain only goes to 1)

$f_c = 1/(2\pi*220pF*100k) = 7$ kHz

(5) Part a) Derive an equation for the negative terminal of the op amp based on V_1, V_2 or V_3.
No current into op amp, $V_+ = V_2$
Negative feedback drives voltage inputs of op amp to be equal, $V_+ = V_-$
So, $V_- = V_2$

(5) Part b) What is the gain (V_3/V_1) for this circuit at DC?

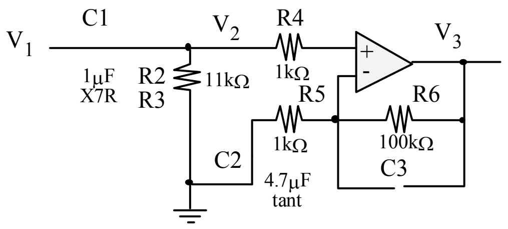
Open all caps.

Gain will be zero (C1 blocks)

(5) Part c) What is the approximate gain (V_3/V_1) for this circuit at audio signals (around 1 kHz)?

Short the uF caps, open the pF cap, DC supplies to ground

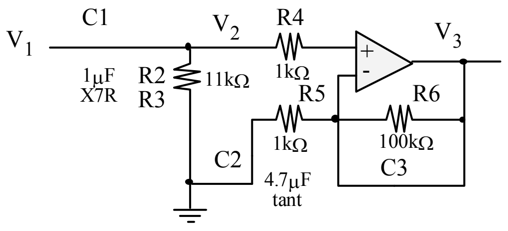
Gain = 101



(5) Part d) What is the approximate gain (V_3/V_1) for this circuit at high frequencies (above 1 MHz)?

Short all caps, DC supplies to ground

Gain = 1



(35) Question 4. Consider a file system that manages a solid state disk

(5) Part a) At this point in time, does this file system have external fragmentation?
Yes there is external fragmentation, there are 256 entries in the IT, meaning the largest block that can be allocated is 256 blocks times 512 bytes/block, which is much less than the size of the disk, which is 65535*512 bytes

(5) Part b) Exactly how many bytes of internal fragmentation does this file system have?
File JV is wasting 412 bytes and the other two are not wasting any. Internal fragmentation is 412 bytes.

(25) Part c) Write a C function that returns a byte from a file at a random location.

```
int eDisk_ReadBlock(unsigned char *pt,    // returned by reference
  unsigned short blockNum);               // which block to read
unsigned short IT[256];        // place to store IT
unsigned char Databuf[512];    // place to store data
unsigned char eFile_Read(unsigned short numIT,
                         unsigned long address){
unsigned long logical,byte,physical;
  eDisk_ReadBlock(IT, numIT);  // read IT
  logical = address/512;       // logical block number
  physical = IT[logical];      // which block has data
  byte = address&0xFF;         // byte in that block
  eDisk_ReadBlock(Databuf, physical);  // read data
  return Databuf[byte];
}
```

A good OS would check to see if `physical` and `numIT` were valid block numbers

(20) Question 5. Write software to implement this moving averaging low-pass digital filter.

$$y(n) = (x(n) + x(n-1) + x(n-2) + x(n-3) + x(n-4) + ... + x(n-127))/128$$

```
long static x[128]; // works only if this is a power of 2
long static Sum;
unsigned long static I;  // index into buffer 0 to 127
void Filter_Init(void){
  I = 0; // (could initialize x if you want)
  Sum = 0;
}
long Filter(long data){
  Sum = Sum – x[I];  // subtract oldest
  x[I] = data;       // new data into place that used to hold old
  I = (I+1)&0x7F;    // set up for next time
  Sum = Sum + data;  // add newest
  return Sum>>7;
}
```