Problem Set #6: ADC & DAC with ECG Data

DUE: Monday, 2013-03-31 at 5:00 PM in the grader box.

1. **Binary Search Algorithms / Successive Approximation ADC**

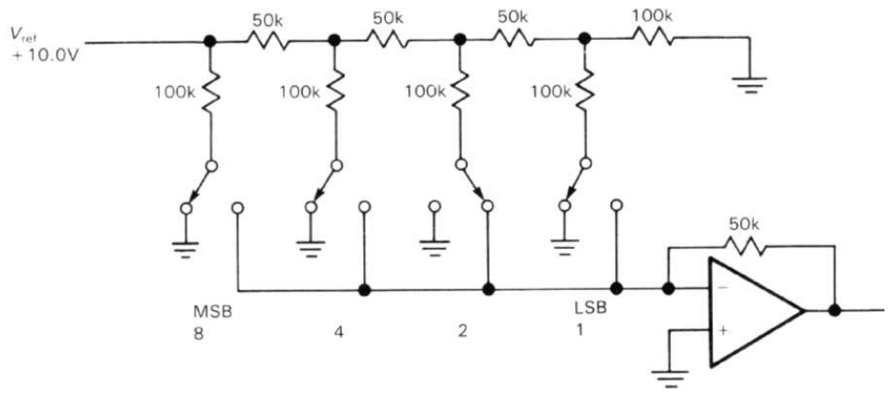
   The successive approximation ADC that we covered in lecture uses a binary search algorithm to determine the most accurate binary representation of a base-10 number. Write a Matlab function that does the following:

   - Accepts input arguments of:
     - Base-10 number to approximate with a binary number
     - Number of bits in binary representation
     - Minimum base-10 number to represent
     - Maximum base-10 number to represent
   - Outputs your binary number and generates a plot of the approximations that it makes with each iteration of the binary search (both the binary and the base-10 equivalents; think of using `plotyy` in Matlab).

   Use you binary search algorithm that you wrote for the following problem. An EMG experiment records the number of fasciculations of the eye lid that occur overnight. It is expected that between 10,000 and 10,000,000 discrete events will be recorded, with a precision of $\pm$ 1000 discrete events.

   - Determine the fewest number of bits that you need to represent your final event count from the experiment as a binary number.
   - Generate a plot of the approximations that a successive approximation ADC would make estimating this final event count using your Matlab function.
   - For your chosen bit depth, your least significant bit (LSB) corresponds to how many "events"?

2. **DAC: R-2R ladder**



   - Solve for the currents through the 2R (100 kΩ) resistors associated with each bit branch for the binary numbers 1100 and 0011. (A switch will be in the GND position when it represents a '0'.)
   - How do the bit branch currents (1) compare to one another, and (2) change as a function of the binary digit they represent (0 or 1)? Does this make sense?
   - What is V_o (the output voltage from the op amp) for the two binary numbers?

3. **Canine ECG Data**

   We have finally reached the point in the course where we can start working with some real biosignals! Working with real signals means that we no longer have "simple" analytic expressions to represent our input signals... so we are going to rely on numerical methods to evaluate our signals. Be sure to include all of your code for this problem.

   The first signal that we will be working with is an ECG trace from a canine cardiac study. The dog had undergone several radiofrequency ablations before this trace was acquired, and it experienced some degree of myocardial ischemia (i.e., this is not a normal ECG trace). The ECG data (`ecg`) is in millivolts, and the time variable (`t`) is in seconds.

   (a) Download `BME354L_CANINE_ECG.mat` from Sakai.

   (b) Load `BME354L_CANINE_ECG.mat` into your Matlab workspace.

   (c) Plot the ECG signal as a function of time, with the axes properly labeled with units. Print this plot out to turn in with your problem set.

   (d) What is the range of voltages for the ECG signal? What is the approximate voltage range for the noise?[1]

   (e) What is the minimum number of bits that we need to accurately represent the data? What was the approximate SNR for this ECG trace?

   (f) What should the voltage of the least-significant-bit (LSB) be set to in your ADC?

   (g) What is the sampling frequency (f_s) for these data?

   (h) Generate a plot clearly showing the frequency content of this ECG data trace. There are several ways to do this in Matlab; the most direct way involves using the `fft` and `fftshift` commands.[2]

   (i) Next, we're going to manually downsample the data to qualitatively get a feel for what minimum sampling frequency can be to still accurately represent the morphology of the ECG signal. Downsample the ECG data (and the time data) by factors of 10, 50, 100, 200, 500, and 1000. You can do this in Matlab by skipping data points (e.g., you can create a vector of every tenth sample in the ECG signal using the syntax `ecg(1:10:end)`).

   Generate plots comparing these waveforms (zoom in on just one of the wave complexes) and notice the aliasing / signal corruption that occurs when the waveform is undersampled. Print these plots to hand in with your problem set (you can put them all on the same plot as long as you use different line styles to distinguish them; the `subplot` command can also be used to save space/pages).

   (j) Generate plots of the frequency content of these downsampled ECG data traces and comment on what you expected these to look like relative to the original signal and if your plots match your intuition.

   (k) Using this manual technique to visually evaluate the effects of aliasing, what is the minimum sampling frequency that could be used to adequately represent these data digitally? What does this imply about the maximum frequency content of the ECG signal? How does this compare with the Nyquist limit that you would have chosen based on your power spectrum from Problem Set #6?

   (l) How much greater than this minimum sampling frequency was the data acquisition sampling frequency? Is this adequate?

[1]There are several different ways that you can quantify these values that are equally valid but may provide different values. Clearly state/show the way that you estimated these values.

[2]If you need to brush up on your FFT background from BME171, then there is a great online reference: `http://www.dspguide.com/`; the chapters are freely available online.

(m) Many datasets in research and industry are saved as "raw" binary files, as opposed to ASCII files (too large) or proprietary formats (such as Matlab's `*.mat`). Download `BME354L_CANINE_ECG.dat` from Sakai; this data file contains a modified version of the ECG data that you had in the Matlab file. The data are formatted as 32-bit floating point numbers and were saved as a serial stream of time/voltage pairs (e.g., t1,v1,t2,v2,...). Read these data in Matlab or your program of choice (something that will let you perform Fast Fourier Transforms and cross correlations)[3]. In Matlab, commands that may make your life easier are `fopen`, `fseek`, `fread`, and `fclose` (and yes, this is more work than just loading in the `*.mat` file, but much more generalized). What are the differences between this ECG signal and the original Matlab signal? (`BME354L_CANINE_ECG.mat`)? Include your code and plots with your problem set.

(n) Compute the power spectrum for this new ECG signal and compare/contrast it with the power spectrum for the original ECG signal. What would you need to do to this new ECG signal to achieve the same power spectrum as the original ECG signal?

[3] In addition to Matlab, you can also use R, Excel, LabVIEW, Maple, Mathematica, or Octave... it is your choice.