Embedded Scientific Computing Project

November 18, 2020

The project will eventually require a complete design implementation of a digital, fixed-point, fourth-order Butterworth filter with a -3 dB frequency of $f = \frac{\omega}{2\pi} = 1046.502$ Hz and a dc gain of $|H(i0)| = 5$. Butterworth filters are often used because they have a frequency response that is maximally flat in the passband (given their order). A continuous-time fourth-order Butterworth filter has four poles equally spaced around a half circle in the left-hand plane (note that this is the Laplace domain, or s-plane). For a -3 dB frequency of ω_c radians per second, the poles are located at

$$s \in \left\{ \omega_c e^{(i\frac{5\pi}{8})}, \omega_c e^{(i\frac{7\pi}{8})}, \omega_c e^{(i\frac{9\pi}{8})}, \omega_c e^{(i\frac{11\pi}{8})} \right\}.$$

(I) Part One of the project involves discretizing the continuous-time filter; later parts will include choosing a computational structure and implementing those structures in fixed-point mathematics.

(A) Find the s-domain transfer function $H_c(s)$ for a fourth-order Butterworth filter with a $-3dB$ frequency of $f = 1046.502$ Hz. Please find the transfer function by writing the denominator as the product of four factors whose pole locations are specified above. Then choose the numerator to be a constant selected so that the DC gain of the filter is 5. Everything that you need to design this filter is in the first paragraph of this handout.

(B) Determine an appropriate sampling time Δt for the discretization of the filter. There are two criteria that must be met for your selection of Δt. First, the system must allow for the reconstruction of signals of up to 2 kHz without aliasing (think the Nyqist rate). Additionally, the sampling time Δt must be selected so that the absolute value of the error between $H_{d,exact}(z)$ and $H_d(z)$ for your discretization is less than one percent of the maximum value of maximum gain of $H_c(s)$.

(C) Discretize the $H_c(s)$ to find a z-domain transfer function $H_d(z)$ that approximates the continuous-time Butterworth filter.

(D) Determine the direct-form II state-space representation for your resulting fourth-order filter.

(E) Generate a frequency response for the original continuous-time filter $H_c(s)$ and the discretized filter $H_d(z)$. Compare the two filters, comparing the frequency repsonses to make sure that $H_d(z)$ has the correct -3 dB frequency, dc gain, and roll-off. (You will want the frequency response of $H_d(z)$ to use $\omega = \frac{\Omega}{\Delta t}$ as the independent frequency variable, rather than Ω.)

(F) Turn in a write-up describing your process so far. This write-up should be written formally. (This will eventually become part of your design report; your goal should be to document your design thoroughly enough that someone could repeat your work.) Be sure to report on all the design choices you made: sampling time, computational structure, the fixed-point format for each coefficient in that structure. Also report on the quality of the resulting filter so far. Include analysis of any structures that you considered.

(II) Part Two of the project involves implementing the discretized filter whose computational coefficients are in a fixed-point representation scheme in C.

(A) Correct any technical issues with Part One of the project.

(B) Determine a parallel-form implementation for this filter with subsystems whose order each is at most two. Make sure that all matrix entries are real.

(C) Approximate the multipliers of your filter in the fixed-point representation of you choice. Target having the poles move no more than about 5% of their distance from the unit circle; this will prevent the frequency response from changing very much.

(D) In Matlab, check the frequency response of your discrete-time filter with fixed-point coefficients to be sure you are still satisfied with the results. You may want to evaluate more than one computational structure.

(E) Write C code (using floating-point mathematics) to implement the discretized filter (with "infinite precision" coefficients).

(F) Write C code (using floating-point mathematics) to implement the approximated filter (with fixed-point coefficients).

(G) Turn in a write-up describing your process so far. (This will eventually become part of your design report; your goal should be to document your design thoroughly enough that someone could repeat your work.) Add to the report from the first part to document all your work on this project so far. The submission for this part of the project should include the (corrected) first report.

(III) Part Three Completes the project.

(A) Correct any technical issues with Part One or Part Two of the project.

(B) Suppose that the total error at the output due to quantization of signals must be bounded by 2^{-7}. Determine the number of fraction bits for the input signal, the signals in the output equation, and each of the signals in the state-update equations to achieve this output error. (This is like Homework XX).

(C) Assume that the filter input $x[k]$ is the output of a analog-to-digital converter (such as Texas Instruments' ADS8887) whose output is represented using the $[18, -16]$ two's complement representation scheme. Determine a bound on $\|x\|_{\ell_\infty}$. Are there a sufficient number of fraction bits on the input compared to part (B)?

(D) Using the bound on $\|x\|_{\ell_\infty}$ that you determined in part (C), determine the number of integer bits required for each signal to ensure that no overflow occurs. (This is like Homework XX).

(E) Draw an implementation diagram of your filter with the format of each signal clearly marked.

(F) Write the C-code to implement the finite-precision filter sing additions, multiplications, and shifts.

(G) Turn in a design report discussing your design and the results generated.