Objectives

1. Begin using pre-designed software routines to implement discrete-time filters in real-time.
2. Learn an inefficient way to design a digital filter by iteratively hand-placing poles and zeros in the $z$-plane.
3. Gain some familiarity with some useful MATLAB functions.
4. Develop an understanding of the relationship between the location of poles/zeros and the shape of the filter magnitude response.

Assignment

TASK 1: If your Lab 2 FIR or IIR routines do not work correctly, you'll need to fix them in order to do this lab. Please work with the instructor or TA to get your code correct.

In Lab 2, you created routines to implement an arbitrary FIR filters and cascade-biquad IIR filters. You tested your routines using a 20-coefficient FIR filter, and a 4th-order IIR filter. For this task, re-write your real-time implementations using the optimized DSP library provided by STM. You'll need to use the following functions:

$$\text{arm_fir_init_f32()}, \quad \text{arm_fir_f32()}, \quad \text{arm_biquad_cascade_df2T_init_f32()}, \quad \text{arm_biquad_cascade_df2T_f32()}$$

Document how you verify that the modified routines provide the correct results, and provide execution time benchmarks comparing the performance of your own routines to the optimized routines. Be sure to enable the compiler optimization by including the "-O3" flag in the CFLAGS variable of the Makefile. (If needed, feel free to modify or improve your implementations from Lab 2.)

TASK 2: Using Matlab and knowledge you have on placement of poles and zeros in the $z$-plane, design a bandpass filter that meets the specifications of Figure 1. You should try to minimize your filter order while remaining within the filter specifications.

Figure 1 shows required limits on the gain of the filter $|H(f)|$ in dB versus the normalized frequency $f$. The filter should pass frequencies that are contained within the $0.25 \leq f \leq 0.35$ band, following the desire gain characteristic shown in red $\pm 1$ dB. The filter should attenuate frequencies below $f = 0.2$ or above $f = 0.45$ as illustrated.

You are NOT ALLOWED to use any automated filter design toolboxes for this assignment. You may use Matlab's basic polynomial manipulation functions (e.g. poly(), roots(), polyval()). You will be asked to demonstrate the design of a different filter during your group meeting. All group members are expected to be familiar with the filter design methodology.

Since you will be using a biquad filter structure to implement your design, keep the number of poles and zeros for your IIR filter designs equal to each other.

TASK 3: Write a short Matlab script or function which combines an arbitrary collection of poles and zeros and a desired gain constant into second-order sections, and provides the coefficients in a form suitable for your “biquad” filter functions which were created in Lab 2. You may use the “zp2sos()” function for this task. Structure your Matlab design scripts so that it's easy to export your filter coefficients into your C implementations. You will be asked to do this during your group demonstration.
Task 4: Test your filter coefficients from Task 2 in real time by using your biquad filter routines from Lab 2. (If you biquad filter routine did not work correctly for Lab 2, you must correct it to complete this lab. You must follow the function prototypes as given in Lab 2.) Measure the magnitude response of your filter, and compare the results with the gains predicted in your Matlab design.

What should be handed in:

1. Hand in a short (English) description of the programs you are submitting, and how they are used. Provide documentation showing how you tested your routine and give your test results showing that it works.
   Provide a table comparing your filter implementation performance versus that of the STM optimized routines.
   Include your commented MATLAB code allowing you to iteratively place poles and zeros to create the desired magnitude response. Give a short written description of how you modified the pole-zero placement to achieve the desired response.
   Give your IIR filter coefficients for the required filter, and provide the corresponding pole-zero plot and a plot of the magnitude response of the filter (in dB). On the same plot, show your measured filter gains for your C implementation.

2. Hand in commented C source code listings.
   Program listings and documentation will be collected at the beginning of class on the due date. Write-ups handed in after class begins, but before 1:00 p.m. on the due date, will receive a deduction of one letter grade for the “written lab summary” portion the lab grade. Labs handed in after 1:00 p.m. will not be accepted.
3. All programs must be archived in your group’s GIT repository by the code due time. Any code archived after the
due time will not be graded. Please include all files in the “lab3” subdirectory of your group’s project.
Your Matlab script for placing poles/zero should be left in a state required to recreate the above filter design.

4. Demonstrate your working routines during your scheduled group meeting time in the week of the due date.