Lab 9
Fixed-Point Considerations

Dr. Aly El-Osery

November 28, 2011

We have discussed several ways to implement a digital filter. Due to the fact that we implement the filter using finite-word length, there is rounding and truncation errors. Also, in order to have a more optimized code, in some cases you may need to use fixed-point arithmetic instead of floating-point. In this lab, you will learn how to implement a filter that performs fixed-point arithmetic and see that the performance of the filter differs based on the implementation.

1 Introduction

Floating-point arithmetic requires more clock cycles than fixed-point. Depending on your application, using fixed-point might be desirable. To design a filter function that uses fixed-point arithmetic, you will need to use `int` (signed 32-bits) or `short` (signed 16-bits) for all your data types. But, the filter coefficients are of type `float` usually between -1 and 1. Therefore, you need to scale your filter coefficients by 2^{n-1} where n is the desired number of bits including a sign bit, then round the numbers to an integer. Further scaling is required if the coefficients are not between -1 and 1.

2 Lab

2.1 Part 1

1. Design a Butterworth IIR filter with the following specifications:
   - $f_{pass} = 4$kHz,
   - $A_{pass} = 0.1$dB,
   - $f_{stop} = 4.5$kHz,
   - $A_{stop} = 50$dB, and
   - $f_s = 48$kHz.

   once by having the `butter` function return a transfer function numerator and denominator and a second time by having the function return poles and zeros.

2. What is the order of the filter?
3. Plot the pole-zeros of both approaches.

2.2 Part 2

1. Design an elliptic filter with the above specifications listed in Part 1.
2. Scale the coefficients to obtain them in 16-bit format and store them.
3. Design a filter that uses fixed-point operation.
4. There are several ways to group the poles and zeros of the filters. Implement the filter twice: once by pairing the poles closest to the unit circle and go inward, and the other by pairing the poles furthest from the unit circle and go outward. Each time pair the poles with the zeros closest to it.
5. Compare the performance of the two implementations.