ELEN90058 Signal Processing

Workshop 1, Weeks 2-4

Sampling of analog signals and design of anti-aliasing filters

Aim: To investigate the relationship between the spectrum of analog and sampled signals and to design anti-aliasing filters.

Workshop: This workshop consists of three parts.

Part A consists of tutorial type questions dealing with background theory which prepare your for the other parts. These questions should be done ahead of the workshop, and you must complete this part before you start on part B and C.

Part B is the main part where you familiarize yourself with the Analog Devices' BF533 DSP board and the CrossCore Embedded Studio (CCES) environment, supported by Matlab to verify your results. You will learn how to generate and plot signals in the time and frequency domain and implement simple low order digital anti-aliasing filters. Before you start on this part you should read the note "Supplementary Notes on CrossCore Embedded Studio" posted on LMS.

Part C deals with design of analog anti-aliasing filters using Matlab.

Assessment: The workshop is assessed based on a written report (see below) and on demonstrations of the programs implemented on the DSP board in the workshop. The workshop constitutes 2% of the overall assessment of the course. This workshop has a low weight since it acts as an introduction to the CCES environment and the DSP board which will be used in the next workshops.

Submission of the reports: The reports should be submitted by 10.00 am on Wednesday the 23rd of August. Details about how to submit will be provided on LMS. The reports should cover part A and B. You also have to sign a declaration that the report is your own work.

Student groups: The students should work in groups of three.

Reports: The reports should cover part A and B only. They should be clearly written and explain in a logical way how the different tasks in the workshop have been carried out. Choices you make (e.g. filter coefficients, critical frequencies etc.) should be explained and justified. Results obtained using Matlab should be explained, i.e. it is not sufficient to copy the output of Matlab without further explanations of what the numbers or graphs mean. Figures should be included where it is appropriate. The Matlab and c code should be included in an appendix.

Collaboration between and within groups: It is perfectly OK to discuss problems and possible solutions with other groups. However, each group has to carry

out the workshop independently, and e.g. copying of other groups' mathematical derivations, c code or Matlab code is not acceptable. Both group members should do an equal amount of work on both the workshop tasks themselves and the writing of the report.

A Questions

a) Given a continuous time signal

$$x_c(t) = \begin{cases} e^{-at}\cos(\Omega_1 t) & t \geq 0, \quad a > 0 \\ 0 & t < 0 \end{cases} \tag{1}$$

Derive an expression for the spectrum $X_c(j\Omega)$. (If your answer is a sum of two or more fractions, write your answer as one fraction.) Plot $x_c(t)$ and $X_c(j\Omega)$ in Matlab for some values of a and Ω_1 and comment upon how the signal and spectrum changes with a and Ω_1.

b) Let the sampled version of $x_c(t)$ be $x[n] = x_c(nT)$. Calculate the DTFT $X(e^{j\omega})$ for the sampled signal. (If your answer is a sum of two or more fractions, write your answer as one fraction.)

c) Let $a = 0.12$, $\Omega_1 = 0.25\pi$ and let the sampling frequency be $F = 4.8$Hz. Plot $X_c(j\Omega)$ and $X(e^{j\omega})$ in Matlab. Do the necessary magnitude and frequency scalings so that the two plots match up. Explain how the command `freqz` in Matlab works and use it to plot $X(e^{j\omega})$.

d) A simple first order digital lowpass filter is given by

$$H_{LP}(z) = \frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}}$$

Show that

$$|H_{LP}(e^{j\omega})|^2 = \frac{(1-\alpha)^2(1+\cos\omega)}{2(1+\alpha^2-2\alpha\cos\omega)}$$

For which values of α is the filter stable? Plot $|H_{LP}(e^{j\omega})|$ and $\angle H_{LP}(e^{j\omega})$ for some values of α. Comment on the results.

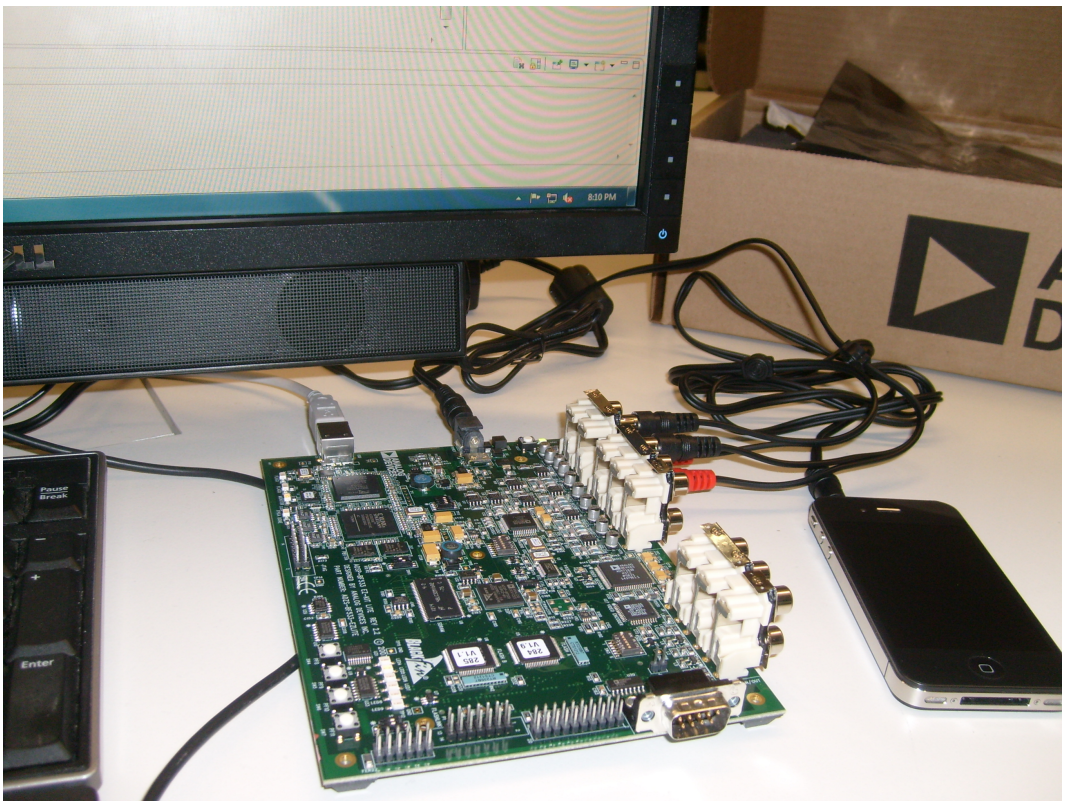

Figure 1: BF533 EZ-KIT Lite

B Implementation of digital anti-aliasing filters on a DSP

In this part of the lab you are going to implement digital anti-aliasing filters on Analog Devices Blackfin 533 Digital Signal Processor which is a 16 bit fix point processor (for further details see http://www.analog.com/en/embedded-processing-dsp/blackfin/processors/index.html). You will be using the CrossCore Embedded Studio (CCES) development environment (see separate note for a description of this development environment) and the ADSP-BF533 EZ-Kit Lite board (hereafter called the board or the DSP board).

B.1 Lab setup

The DSP board is connected to a PC through a USB port. See figure 1 for a picture of the lab setup. You will program the filter algorithms in c using CCES. The CCES environment provides facilities for debugging, compiling, linking and downloading of the code from the PC to the board.

B.2 Digital anti-aliasing filters

Instead of implementing high order analog lowpass filters, an alternative is to sample fast and use simple low order lowpass filters (or no analog anti-aliasing filters at all). In this part of the lab we are going to investigate this approach. As a test signal we use

$$x(t) = x_1(t) + x_2(t) \tag{2}$$

where

$$x_1(t) = e^{-at}\cos(\Omega_1 t),\ t \geq 0 \tag{3}$$
$$x_2(t) = 0.1\sin(\Omega_2 t), t \geq 0 \tag{4}$$

where $a = 0.12$, $\Omega_1 = 0.25\pi$ and $\Omega_2 = 1.9\pi$. Both signals are 0 for $t < 0$.

All the lab tasks below can be accomplished by writing one c-program. That is, you can just add code to your existing program as you progress through the lab tasks. You can save your plots in .jpg, .png and .bmp format, and you can also save them to the clipboard and past them into Word. For details see the note on CCES.

B.3 Preparation

Read through the note on the CCES environment.

Generate and plot the signals in tasks **b)** and **c)** in Section B.4 below in Matlab, so you know what to expect, and find the filter coefficients to be used in **d)**.

Once you have completed this part and the questions in A, **show your workings to the demonstrator**, and you will be provided with a DSP board.

It is also a good idea to do the tasks in question **e), f)** and **g)** in Matlab so you know what to expect. You can do **e), f)** and **g)** in Matlab at the same time as you attempt the tasks on the DSP board.

B.4 Lab tasks

a) Start up CCES on the PC. Open the project file SPWS1 which is can be downloaded from LMS. The file `SPWS1.c` contains the skeleton function in Appendix 1. Build the project and run it on the board (see the note on the CCES environment). Check that the program returns the values

$$x[0] = 0.000000$$
$$x[1] = 0.070709$$
$$x[2] = 0.100000$$
$$x[3] = 0.070716$$

$$
\begin{aligned}
x[4] &= 0.000009 \\
x[5] &= -0.070702 \\
x[6] &= -0.100000 \\
x[7] &= -0.070722
\end{aligned}
$$

Increase the length of the signal to 256, and use the plot facility within CCES to plot the signal in the time domain and in the frequency domain. (The plot facility is described in the note on the CCES environment.)

b) Write a c-program which generates two sequences of 256 samples (from $t = 0$ to $t = 255 \cdot T$) of $x(t)$ in (2), using the sampling frequencies $F_1 = 1.2$Hz and $F_2 = 4.8$Hz.

c) Use the plot facility within CCES to plot the sampled signals in the time domain and in the frequency domain. Comment on the results.

d) From now on use the signal $x[n] = x(nT)$ sampled at 4.8 Hz for further processing. Denote the frequency where $|X_1(e^{j\omega})|$ attains its maximum by $\omega_{\max}$. ($X_1(e^{j\omega})$ is the spectrum of the sampled signal $x_1(t)$ in (3).) We want to reduce the sampling frequency to 1.2 Hz and we want to design a simple digital anti-aliasing filter which has a gain of 0.95 at $\omega_{\max}$, and a gain less than 0.25 at the frequency of the sinusoidal signal $x_2[n]$ (the sampled version of $x_2(t)$).

(i) We consider a first order filter

$$H_{LP}(z) = \frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}}$$

Calculate α such that the filter has gain 0.95 at $\omega_{\max}$. Are the filter specifications satisfied for this filter? Give reasons for your answer.

(ii) We consider next a second order filter which is a cascade of two first order filter

$$H_2(z) = \left(\frac{1-\alpha}{2}\frac{1+z^{-1}}{1-\alpha z^{-1}}\right)^2$$

Calculate α such that the filter has gain 0.95 at $\omega_{\max}$. Are the filter specifications satisfied for this filter? Give reasons for your answer.

e) Implement the two filters in c, and filter the sequence $x[n]$. Plot the output signals in time domain and frequency domain and check your design as follows: (i) From the time domain plot, roughly check that the gain of the implemented filters agrees with the gain of the designed filters at the frequency of the $x_2[n]$ signal. (ii) From the frequency domain plots, check that frequencies you want to removed have been removed and that the frequency content of the output signal is in agreement with what you expect from the frequency content of the input signals and the frequency response of the designed filters.

f) Filter the signal $x_2[n]$ using the designed filters. Plot $x_2[n]$ and the output in the time domain and verify that the gain and phase of the implemented filter agrees with the gain of the designed filters at that frequency.

g) Return to the signal $x[n]$. Let $Y(z) = H_2(z)X(z)$ be the output of the second order filter. Down sample $y[n]$ with a factor 4, that is let $y_{\text{ds}}[n] = y[4n]$, $n = 0, \ldots, 63$. Plot this signal in the time and frequency domain and compare it with the signal obtained by sampling $x(t)$ at 1.2 Hz. Comment on the results.

Once you have completed this part, **show the plots and explain your results and how the code works to the demonstrator. This is part of the assessment of the workshop.**

C Design of analog anti-aliasing filters

In this part you will design and implement analog anti-aliasing filters using Matlab.

A signal $v(t)$ is given by

$$v(t) = v_1(t) + v_2(t)$$

where

$$v_1(t) = e^{-0.1t}\cos(0.25\pi t) \quad (5)$$
$$v_2(t) = e^{-0.4t}\cos(2\pi t) \quad (6)$$

$v_1(t)$ is the interesting signal, and particularly the frequency range where $|V_1(j\Omega)|$ is larger than -6 dB. The signal $v(t)$ is going to be sampled, and we want to design an anti-aliasing filter such that the magnitude of the spectrum of the lowpass filtered $v_2(t)$ signal (that is the signal $v_2(t)$ after it has passed through the anti-alasing filter) is below -50 dB in the frequency range which is aliased onto the interesting frequency range of the $v_1(t)$ signal. Moreover, the maximum allowed passband ripple is 0.01.

a) The signal $v(t)$ is sampled at 1.2732 Hz. Design a Butterworth filter satisfying the above specifications. Implement the filter and the signals $v_1(t)$ and $v_2(t)$ in Matlab or Simulink and verify that the design and implementation are according to the specifications. In order to verify the design and implementation you can e.g. do the following

1 (Design) Check that ripples and gains of the filter are in agreement with specifications.

2 (Design) Check that the spectra of the filtered signals are in agreement with specifications.

3 (Implementation) Simulate the filter in Matlab or Simulink using a sinusoid as an input signal. Check that the output signal agrees with what you expect from the frequency response of the designed filter. Check for a number of frequencies (e.g the cutoff frequency, frequencies in the passband, stopband and transition region, and critical frequencies in the design specification),

4 (Design and Implementation) Let the input signal to the filter be $v(t)$ and check that the output signal is what you expect it to be. This can e.g. be done by (i) plotting the signals in time domain and verify that certain components have been removed, amplified etc. (ii) Compute the Fourier Transform of the input and output signal and check that certain frequency components have been removed/amplified/left unchanged etc. (As Matlab works with finite length discrete time versions of the analog signals, you will have to compute DTFTs of finite length signals which approximate the Fourier transforms of the analog signals.)

b) Determine a sampling rate, and design a fourth order Butterworth filter such that the specifications above are satisfied. Implement this filter as a cascade of two second order filters and verify your design in Matlab or Simulink. (It is sufficient to verify that the two filters are working correctly as a cascade. However, if the cascade is not working according to your design you may have to troubleshoot and check the individual filters.)

1 SPWS1.c

```
#include <stdio.h>
#include <math.h>

// Globals
#define N 8
#define PI 3.1415

float x[N];

int main(void)
{
    int i;
    float omega1 = 0.25 * PI, omega2 = 1.0 * PI;
    float T = 1.0;

    for (i = 0; i < N; i++)
    {
```

```
        x[i] = 0.1 * sin((float)omega1 * T * i);
    }

    for (i = 0; i < N; i++)
    {
        printf("x[%d] = %f\n", i, x[i]);
    }

    printf("Done.\n");

    return 0;
}
```

1.1 Points to note

In c an array with `N` elements is indexed from `0` to `N-1`.

Variables defined as `float` may lose resolution if multiplied or divided by a variable of type `int`. Any such multiplications or divisions should be typecast as floating point by typing `(float)` immediately before it.