CPE3500: Embedded Digital Signal Processing

Lab Exercise 3: Discrete-Time Systems and Convolution

Objective:

The purpose of this lab exercise is to introduce discrete time system definitions using C language on the Nucleo board. The students will be generating system outputs for different inputs.

Introduction:

One of the most important properties of a digital system is that it can be implemented in code form. If a suitable digital platform can execute this code block, then the system can be realized. There are two ways to implement a digital system. The first is to process each input signal sample separately. This is called sample-based processing. The second is to process a block of input signal samples all at once. This is called frame-based (buffer-based) processing.

A sample implementation of digital systems will be provided. This step is necessary to become familiar with frame-based implementation of digital systems.

In this exercise students will:

- Learn how to clone STM32Cubeide projects to work on different exercises based on previous resources.
- Create basic discrete time systems and generate system output.
- Learn how to implement a discrete time convolution function to test on a digital filter example.

Required Equipment:

Personal computer (Windows/Mac OS)

Nucleo-L476RG Board (64-pin)

USB Type-A (or Type-C) to Mini-B cable

Procedure:

1. Clone a Project in STM32CubeIDE

We will have to use CMSIS DSP libraries in this lab which we set up in Lab-2. To avoid repeating the long process that we did in Lab-2, in this procedure, we will learn how to clone a previous project with a different name to work on a new project using the same resources.

Follow the steps to clone an old project:

- Open the workspace folder in your file explorer
- Locate the project folder (Lab-2 project) and duplicate it (copy and paste) with a new **new_project** name.
- Open the new project folder and locate the **old_project.ioc** file and rename it to **new_project.ioc**
- Open **new_project.ioc** file with a text editor (Notepad or TextEdit). Edit **ProjectManager.ProjectFileName=***.ioc** and **ProjectManager.ProjectName=*** with **new_project** name.
- Locate **old_project Debug.launch** file and delete it.
- Open STM32CubeIDE. Click File->Import. From Existing Projects into Workspace, select root directory. From the project list, select the new project and click Finish.
- From Project Explorer window select your project and click on Build button.
- Click on Debug button and select **new_project.elf** file to run debug configuration on your new project.

2. Creating a Discrete Time System

A discrete time (DT) system can be defined as a C function which takes input and output array vectors as an argument and perform the mathematical function to calculate the output signal.

The sample discrete time system below calculates the system y[n]=x[n]+b*x[n-1] where b is a constant:

```
// Discrete Time System y[n] = x[n] + bx[n-1]
void system1(float input[], float output[], int size)
{
    int n;
    for(n=0;n<size;n++)
        if (n==0)
            output[n]=input[n];
        else
            output[n]=input[n]+b*input[n-1];
}
```

Here, the input and output arrays represent the input signal to the DT system and size holds the number of elements (size or length) in the input array.

Task-1:

Place the system1 function in Private user code section inside the main.c file.

/* Private user code ---*/

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

In the main function, define a DT rectangular signal with first 7 elements to be 1. Call system1 with rectangular signal as input and calculate the output signal while b=0.5. Export both input and output signals of the system and plot them in Matlab.

Task-2:

Based on system1, create another function which calculates the system;

y[n]=a*x[n]+b*x[n-1]+c*y[n-1] where a=2.2, b=-1.1 and c=0.7.

Call this function and calculate the output for the same DT signal as input. Export the output signal and plot in Matlab.

Task-3:

Use the sampleMain.c code provided on D2L to generate one single tone sine signal at 5 Hz and one composite sinusoidal signal combining 5 Hz sine with 3 more different frequency sine signals. For this, you can pick any frequency above 25 Hz and below 50 Hz.

Place the sections of the code into appropriate places in the main.c file of your project. A DT low pass filter is defined as a system in the code which you can calculate the convolution sum as the output signal (sineFiltered).

After running the code in debug mode, export single tone sine, composite sinusoidal signal and filtered signals. Plot these signals in a single figure in Matlab. Compare and comment on the results.

Note: Filtered signal may generate slightly higher amplitudes compared the single tone sine. It will be better to normalize this signal by dividing the array elements to the maximum value of the array in Matlab (i.e. sineFiltered = sineFiltered/max(sineFiltered)).

Lab Exercise 3 Report:

Prepare a lab report (as single pdf file) consisting of the followings and upload it to the D2L dropbox.

- Cover Sheet
- Task-1: All the plots with proper labeling and title.
- Task-2: Modified code section and plots.
- Task-3: Modified code section and plots with comparisons. Comment on the results
- Conclusion (1-paragraph)