ECL333 Digital Signal Processing Lab
Department of Electronics & Communication Engineering
Govt Engineering College, Barton Hill

Instructors: Sruthi B. R., Birenjith Sasidharan

February 16, 2022

General Instructions

1. Each machine is shared by two students.
2. You may create a folder "*rollnum*_firstname" inside the home folder, and save all your scripts inside. Please name the scripts and images based on experiment number. For example, if the script is related to first experiment, then you may name it as e1abc.py where abc signifies the task executed by the script. Your folder will be checked periodically by the instructor.
3. The default programming language used will be python. Students are encouraged to try out the same experiments using MATLAB as well.

Installation of Python Environment

It is strongly encouraged to work in Linux OS. The guidelines are specifically for Linux machines. The $ symbol here implies they are linux terminal commands.

1. Download miniconda from this website: `https://conda.io/miniconda.html`
2. Choose 64-bit Python 3.8 bash installer under 64-bit Linux. (If you are running 32-bit Linux, please download 32-bit Python 3.7 bash installer, and follow the same procedure below). It is preferable to use Ubuntu 18.04 LTS or later LTS versions, but other versions should also work. To know whether your machine is 64-bit or 32-bit, take terminal and type the command:

   $ uname -m

   x86_64 (for 64 bit)

   i686 or i386 (for 32 bit)
3. Miniconda3-latest-Linux-x86_64.sh is the binary installer for 64-bit Linux. To install from the terminal, use the command:

   $ bash Miniconda3-latest-Linux-x86_64.sh

   This will install and creates a folder named miniconda3 in the home directory. The installation will also update the .bashrc configuration file by setting the path for miniconda3. Default Python version in miniconda3 is Python-3.8.

4. Now, open a linux terminal, and follow the instructions as below:

   - To create new environment named sclab:
     $ conda create -n sclab python=3.8
   - To activate a particular environment, use the command
     $ conda activate sclab
     In some installations, the command to activate will be $ source activate sclab
   - Once the environment is activated, we will see (sclab), enclosed within parentheses before user-name symbol in the terminal. If (sclab) user@computer-name $ is not seen in the terminal, then it means the environment is not activated. Inside the activated environment, install the following:
     (sclab) – $ pip3 install numpy scipy matplotlib
     (sclab) – $ pip3 install pandas jupyter
     (sclab) – $ pip3 install scikit-learn
     (sclab) – $ pip3 install scikit-image
     In some cases you may have to do installation of pip (Python Package Installer) separately before moving on to the next line. You can install pip as:
     (sclab) – $ conda install pip

5. Open ipython and run the test script as follows:
   (sclab) – $ ipython
   Open jupyter and run the test script as follows:
   (sclab) – $ jupyter notebook

6. To deactivate the environment, use either of the following commands:
   (sclab) – $ conda deactivate
   (sclab) – $ source deactivate

Experiments

Experiment 1: Simulation of Signals

Aim

1. We shall write python scripts that simulate
   (a) unit impulse
   (b) delayed unit impulse
   (c) unit step
   (d) ramp signal
   (e) rectangular pulse
   (f) bipolar pulse
   (f) triangular pulse

2. The signals shall be plotted for sufficiently long duration so that the behaviour of the signal is visible from the plots.

3. The programs shall be written in such a manner that *width* and *delay* parameters (wherever applicable) can be adjusted by changing assignment of respective variables.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 2: Linear Convolution

Aim

1. We shall write a python script that computes the energy of a finite-duration signal $x[n]$ using the fact that Energy$(x[n]) = \mathbf{x}^T\mathbf{x}$ where $\mathbf{x}$ is $x[n]$ represented as a vector.
2. We shall write a python script that implements linear convolution between two finite-duration signals $x[n]$ and $h[n]$. The signal $x[n]$ is of length $(K+1)$ residing at $0-K$. The signal $h[n]$ is of length $(M+1)$ residing at $0-M$. (Assume that $K \geq M$)
3. The function that implements linear convolution shall be extended to take care of signals residing in arbitrary supports.
4. We shall write a python script that implements correlation between two finite-duration signals $y[n]$ and $x[n]$. The signal $y[n]$ is of length $(N+1)$ residing at $0-N$. The signal $x[n]$ is of length $(K+1)$ residing at $0-K$. (Assume that $N \geq K$)

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 3: Discrete Fourier Transform

Aim

1. We shall write a python program that computes the DFT of an N-point signal.
2. We will suitably modify the DFT program to obtain a program that computes inverse DFT.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 4: Fast Fourier Transform

Aim

1. We shall implement in python, the Cooley-Tukey recursive algorithm for N-point DFT computation. We will assume that $N = 2^m$ for some m.
2. We will make suitable modifications to obtain a program for fast inverse DFT computation.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 5: FIR Filter

Aim

1. We shall write python functions *sinc_filter*, *hamming* and *kaiser* that respectively outputs impulse response of ideal low pass filter, Hamming window function and Kaiser window function.
2. We shall write a python program to use the above functions to construct an FIR filter obtained by window method.
3. We shall test the filter on an input signal that is linear combination of multiple tones. The FFT of input and output will be plotted so as to verify whether the filtering happens as expected.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 6: Block Convolution Using Overlap Save and Overlap Add Methods

Aim We shall implement overlap-save and overlap-add methods to carry out block convolution for a long input sequence with a filter of shorter impulse response.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 7: Familiarisation of DSP Hardware

Aim

1. We shall familiarise with TMS320C6713 evaluation board, input-output ports and JTAG emulator. We shall also install Code Composer Studio version 7 (CCS) and familiarise with how to cross compile a C code to obtain the out file, and debug it using CCS.
2. We will write C programs (a) to toggle a specific LED (out of 0,1,2,3) and to sense status of DIP switches, (b) play a single tone, and (c) loop back an audio signal. The C code will be cross-compiled and executed in the DSP hardware.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 8: Convolution Using DSP Hardware

Aim

1. We shall write a C program to convolve two signals. The code will be cross compiled and executed in the DSP hardware TMS320C6713 evaluation board.
2. We shall use memory view of CCS software to check if the operations are correctly performed. The Graph tool of the CCS software will be used to display the signals.
3. We shall also write a C program that reads input signal for convolution from a file.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 9: DFT Computation Using DSP Hardware

Aim

1. We shall write a C program to compute DFT of a complex signal. The program will be cross-compiled and executed in the DSP hardware TMS320C6713 evaluation board.
2. We shall use memory view of CCS software to check if the operations are correctly performed. The Graph tool of the CCS software will be used to display the signals.

Algorithm

Program

Plots and Outputs

Result and Inference

Experiment 10: FIR Filter Using DSP Hardware

Aim

1. We shall write a C program implements a low-pass FIR filter using window method. The program fetches an audio input signal via line-in port, filters using the low-pass filter, and plays it via head-phone port using AIC23 codec.
2. The program will be cross-compiled and executed in the DSP hardware TMS320C6713 evaluation board.

Algorithm

Program

Plots and Outputs

Result and Inference