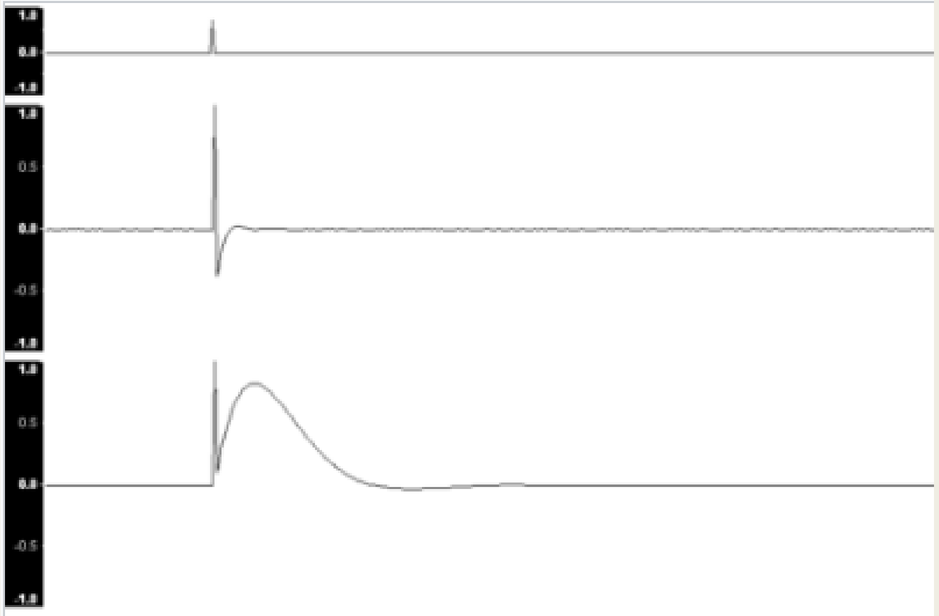
THEORETICAL ANALYSIS ON FINITE IMPULSE RESPONSE

Problem Statement: Parallel simulation of moving average finite impulse response filter

Faculty guide : Dr. Noor Mahammad

Paleti Krishnasai - CED18I039

INTRODUCTION



The Impulse response from a simple audio system. Showing, from top to bottom, the original impulse, the response after high frequency boosting, and the response after low frequency boosting.

In signal processing, the **impulse response**, or **impulse response function (IRF)**, of a dynamic system is its output when presented with a brief input signal, called an impulse. More generally, an impulse response is the reaction of any dynamic system in response to some external change. In both cases, the impulse response describes the reaction of the system as a function of time (or possibly as a function of some other independent variable that parametrizes the dynamic behavior of the system).

In all these cases, the dynamic system and its impulse response may be actual physical objects or may be mathematical systems of equations describing such objects.

FINITE IMPULSE RESPONSE FILTER

$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$
$$= \sum_{i=0}^{N} b_i \cdot x[n-i],$$

- In signal processing, a **finite impulse response** (**FIR**) **filter** is a filter whose impulse response (or response to any finite length input) is of *finite* duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).
- The impulse response (that is, the output in response to a Kronecker delta input) of an N^{th}-order discrete-time FIR filter lasts exactly *N + 1* samples (from first nonzero element through last nonzero element) before it then settles to zero.

PROPERTIES

- Require no feedback. This means that any rounding errors are not compounded by summed iterations. The same relative error occurs in each calculation. This also makes implementation simpler.
- Are inherently stable, since the output is a sum of a finite number of finite multiples of the input values, so can be no greater than $\sum | b_i |$ times the largest value appearing in the input.
- Can easily be designed to be a linear phase by making the coefficient sequence symmetric. This property is sometimes desired for phase-sensitive applications, for example data communications, seismology, crossover filters, and mastering.

MOVING AVERAGE FIR FILTER ANALYSIS

- A moving average filter is a very simple FIR filter. It is sometimes called a boxcar filter, especially when followed by decimation. The filter coefficients, b_0, ... , b_N, are found via the following equation:

$$b_i = \frac{1}{N+1}$$

- Defining the filter order : N = 2
- The impulse response of the resulting filter is:

$$h[n] = \frac{1}{3}\delta[n] + \frac{1}{3}\delta[n-1] + \frac{1}{3}\delta[n-2]$$

APPLICATIONS

Acoustic and audio applications

Electronic Processing

Control Systems

Economics

INSIGHTS AND CHALLENGES OBSERVED

- The major challenge with this filter is to generate the current output based on the 3 separate instances of input at 3 different time instances.
- This would cause a heavy computational load when executed serially given the input is a large value (both in magnitude and precision).
- Another challenge would be to define how to represent the input signal and most importantly, how to determine the edge cases and initial and starting points.
- Based on the above-mentioned arguments and the fact that the moving average FIR filter is the basis for the boxcar filter and the fact that it can be scaled or generalized for most FIR filters necessitates a parallel execution style .

SOFTWARES

- C/C++, programming language
- OpenMP, API for shared-memory parallel programming
- MPI, High performance Message Passing library
- Cuda C/C++, API for utilizing CUDA-enabled GPU for computation

CONCLUSION

- Based on the reading and searching for existing serial codes for the moving average FIR filter, it can be said that the current parallel execution of discrete time moving average FIR filter is not found by the student on an open-source platform. This gives a vast amount of room to work with given that there are no existing codes to base the current understanding on how to simulate/execute the filter parallelly.

- The other major point to be noted is that the student will have to prepare a serial simulation of the discrete time moving average FIR filter. This would enable the student to understand how to represent the inputs and figure out how to tackle the edge cases (cases where there is an interruption in the signal) and the end point cases (y[0], y[n]) .