Linear Systems and Signal Convolution

by Evgeniy Gryaznov and Vitaly Romanov

Due: Saturday, 14 March 2020, 11:59 PM

Format: This is a group assignment, you'll have to divide into groups of 2 or 3 people. It consists of two separate tasks. You'll have to make a short presentation during a lab session. The presentation shouldn't exceed 15 minutes. Your TA will grade the performance of your group and may give any number of points from 0 to 15. Your individual grade is equal to the grade of your group. Your group can perform extra tasks to receive additional points as instructors' gratitude. Also, your group can receive instructors' gratitude for some outstanding features of your solution (e.g., quality, novelty, creativity, etc.).

Files for assignment: files

Moodle Submission: PDF, sources, and accompanying data files.

How to split into groups: ask representative of your student group.

In this assignment, you will learn what are FIR and IIR filters, how to implement them, and how you can use them to apply sound effects to your recordings.

Task 1: Impulse Response and Convolution

Imagine a talented musician playing the violin. It's clear that the environment in which it's played affects the sound. For example, imagine a musician playing the violin in your room and in a giant concert hall. There'll be a great difference in sound. By hearing a recording of a violin, you can certainly deduce some information about the environment in which it was recorded. This effect can be distilled by playing a single high pitch sound and then record how an environment echoed it back. You are already familiar with an ideal "single high pitch sound," its called Dirac δ-sequence. This response of the environment on δ-sequence is called its *Impulse Response Characteristic*, or IRC for short.

It turns out that you can record IRC of any building, room, or other closed environment and then use it on any kind of sound to make it sound like it was actually recorded in the given environment. In order to do that, you have to convolve IRC with your sound. By convolving we mean computing the discrete convolution of your signal (a.k.a. sound) with the recorded IRC using the following formula:

$$(f \star g)_n = \sum_{k=0}^{n-1} f_k g_{n-k}$$

Here, $\star$ sign means convolution of your signal f and IRC g. Most of the time the duration of the IRC g is limited. The convolution of a signal f with a finite-length kernel g corresponds to applying Finite Impulse Responce (FIR) filter to the signal.

In this task, you have the following goals:

1. Record IRC of some interesting place in Kazan or Innopolis. Be creative, go to a church or a concert hall, or metro. Take some `pictures` of the place. The group who recorded the most interesting place will receive the instructors' gratitude!
2. Implement the convolution algorithm in Scilab. Try to optimize its execution time. The group with the fastest algorithm will receive instructors' gratitude. As a rule of thumb, the implementation that is vectorized (no loops) will be the fastest. Use SciLab's `tic` and `toc` to time the execution.
3. Apply the environment effect of your chosen place on your favorite track by convolving it with IRC using your custom algorithm. Also convolve the following tracks with IRC: `voice.wav`, `violin.wav`, `speech.wav`, and `drums.wav`. All files can be downloaded from the document head.
4. Do the same, but with the built-in Scilab's method `convol`. Can you hear any differences? Compare results with your algorithm. Explain any differences and artifacts in the resulting sound. **Make sure to play them during your presentation!**
5. Extra goal: Find an *anechoic* room, record your own track and then apply on it effects on your chosen environment.

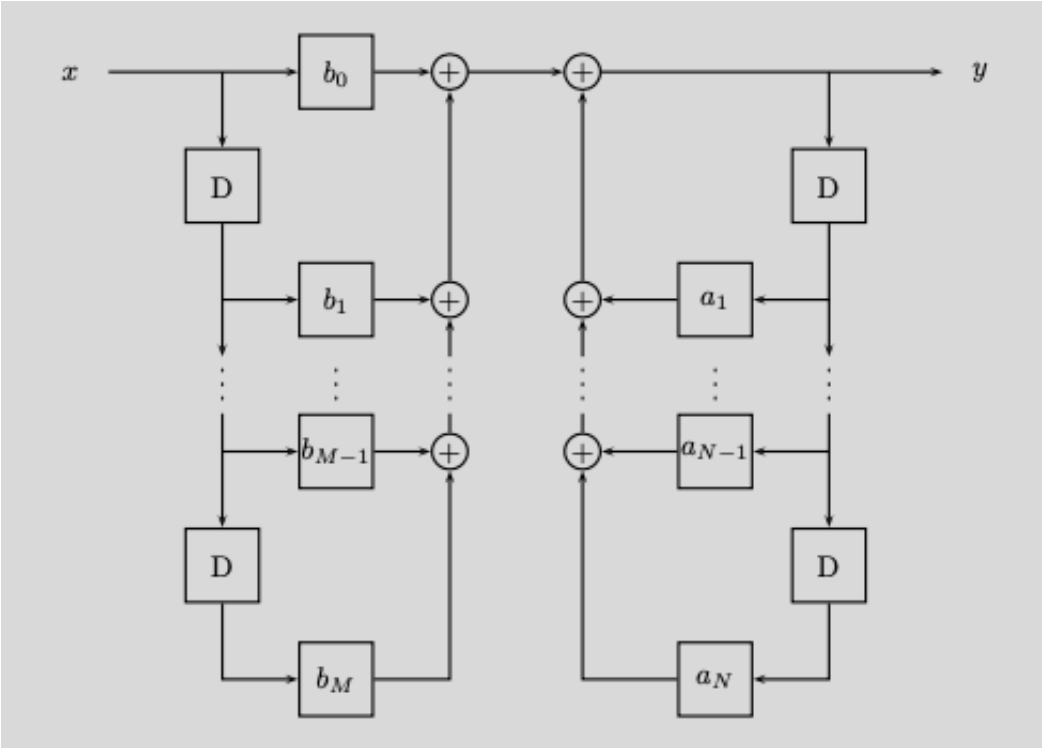
> ## Q&A for Task 1
>
> **Q: How exactly should we record an IRC of our environment? A:** You'll need a good microphone and a pair of hands. First of all, prepare your room. There should be an absolute silence before you start recording. Any background noise may interfere with the quality of IRC. Turn on your microphone and clap your hands once as loud as you can. Then turn the microphone off after echo is gone. Congrats, your microphone just recorded the IRC of your room! Clapping will simulate the sound of Dirac δ-sequence. You can also think of any other method of producing this sequence (for example, you can play an actual δ-sequence with your speakers). **Q: How can we import soundtracks into Scilab?** A: Firstly, convert them to `.WAV` format, then use Scilabs' built-in method `loadwave` or `wavread` (read the descriptions of these functions before using them). You can play the imported sound back by passing the result into `playsnd` function like that: `playsnd(loadwave('myIRC.wav', 44100))`.

Task 2: Frequency Filtering with IIR Filters

FIR filters explain how to perform filtering using a convolution with a finite length kernel. In contrast, IIR filters allow processing signals using a kernel of infinite length. The most common way to represent an IIR filter is by using a recursive equation:

$$y[k] = \sum_{m=0}^{M} b_m \cdot x_{k-m} + \sum_{n=1}^{N} a_n \cdot y_{k-n}$$

This equation corresponds to a system:

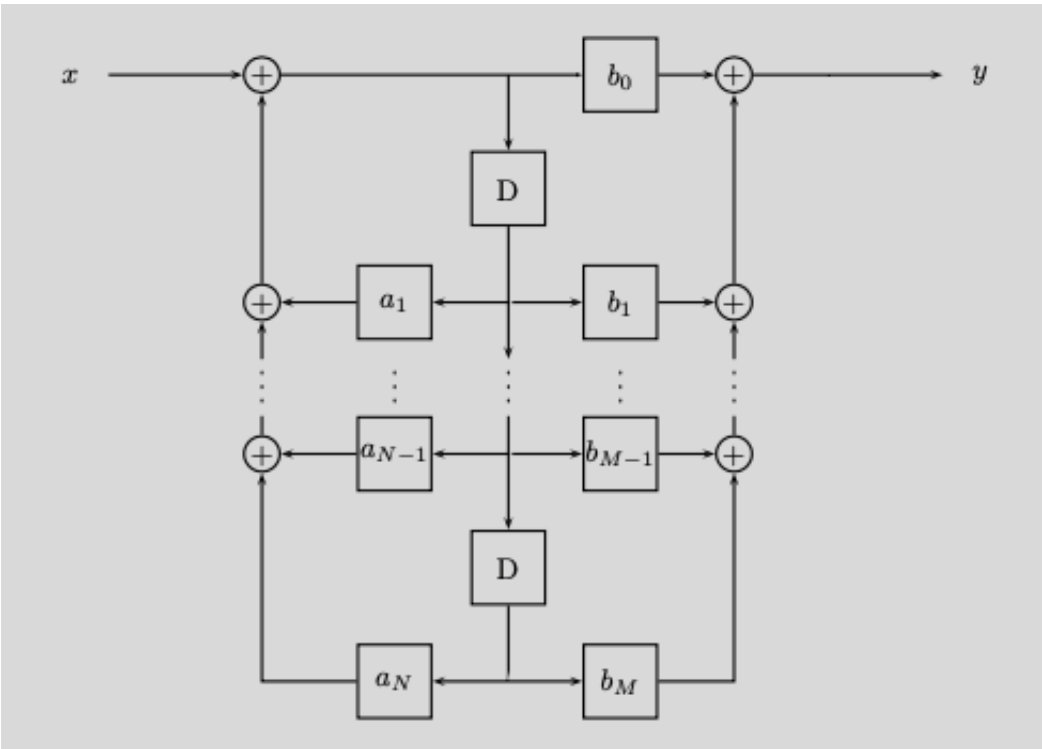


System representation of recursive filter

Looking at the system above, one can see two distinct processing steps. The first is a simple FIR filter with coefficients b, and the second is a filter with a feedback loop with coefficients a.

$$\begin{aligned} y &= T_a(T_b(x)) \\ &= T_b(T_a(x)) \end{aligned}$$

Both of these filters represent linear shift-invariant systems (LSI). Hence, we can safely swap the order of filtering (due to the associativity of convolution). After the order of filtering is changed, we can reduce memory requirements in half.



Canonical representation of IIR filter

IIR filters were a useful solution during the dawn of digital systems. They allowed expressing complex behavior with a few physical components. Typically those components are memory cells (implemented as shift buffers) and multipliers. An IIR filter can do with a dozen components, whereas an FIR filter would require a hundred. This happens because a recursive equation defines the dependency between the consecutive samples of an IRC, and, in the case of an FIR filter, these values should be stored explicitly.

Nowadays, allocating hundreds of memory cells is hardly a problem, and an FIR filter is usually a way to go. Moreover, the design process of an FIR filter allows for much better control of the final filter properties. IIR filters offer more complex behaviors with a fewer number of coefficients. However, unlike FIR filters, which are always BIBO stable when all of the coefficients have finite values, the stability of IIR filters should be verified during the filter design process. This topic will be discussed in more detail in the future.

The implementation of an IIR filter consists of one memory buffer and two sets of coefficients. The implementation, in general, is straightforward.

1. Write down the equation for the canonical representation of the IIR filter.
2. Implement a function that creates an IIR filter. Since IIR filters possess a state, it is useful to store the filter in some data structure. You can create one using SciLab's `struct`. In this structure, you can store the sets of coefficients and the memory buffer. Remember that SciLab can create a copy of an object when you pass it as an argument to a function. Be aware of this when updating the state of a filter.
3. Create two filters: lowpass filter - a filter that allows only slowly changing signals through, and a highpass filter - the filter that allows only fast-changing signals through. The sets of coefficients are the following:

```
// Lowpass:
a = [1.9733442497812987, ...
      -0.9736948719763] // coefficients for a_1 and a_2
b = [0.00008765554875401547, ...
      0.00017531109750803094, ...
      0.00008765554875401547] // coefficients for b_0, b_1 and b_2
// Highpass
a = [-0.3769782747249014, ...
      -0.19680764477614976] // coefficients for a_1 and a_2
b = [0.40495734254626874, ...
      -0.8099146850925375, ...
      0.4049573425462687] // coefficients for b_0, b_1 and b_2
```

4. Apply these filters to a piece of music recording. You should produce similar outputs: lowpass and highpass.

Self-check Questions

1. What is an IRC of an environment?
2. Why the Dirac's Delta sequence is used in recording IRC? (Look at the lecture slides from

weeks 4&5)

3. What is a FIR filter?
4. What is an IIR filter and how it differs from an FIR filter?
5. Define convolution? In what way the convolution operation similar to cross-correlation operation?
6. Assume you have a system $\mathcal{L}$. You want to simulate the response of a system to the input signal $\mathcal{S}_1$. How can you do this? What will be the response of this system to the input signal $\mathcal{S}_1 + \mathcal{S}_2$?