Title: An Introduction to Discrete Time Signals and Filters
Speaker: Kurt Butler
Date: October 16, 2019

Abstract: From the sounds of crunching leaves to the camera feed of humans fleeing from their AI overlords, digital systems have been employed to record and analyze all kinds of data. The need to be able to process data in an efficient manner spurred the development of digital signal processing (DSP) as a discipline. A signal is any data that varies with respect to time or space. DSP employs a variety of mathematical techniques to analyze signals and extract useful information from them, by filtering out noise, augmenting noteworthy features and other manipulations. In this talk we study a particular class of signal filters, called \textit{linear-time invariant systems} that admit an elegant description and are employed in a wide variety of settings.

There are no formal prerequisites for the talk, but comfort with linear algebra and complex numbers will make the discussion much more interesting.

\textbf{Signals} are formally defined to be functions $x: T \to \mathbb{R}$ (or $T \to \mathbb{C}$). The set $T$ is called the \textbf{domain} of the signal. Commonly, we consider two domains; When $T = \mathbb{R}$, we call $x$ a \textbf{continuous time (CT) signal} and we call it a \textbf{discrete time (DT) signal} when $T = \mathbb{Z}$. The set of all signals over a common domain form a \textbf{vector space} over $\mathbb{R}$ (using operations 1 and 3 below). If we add vector multiplication (operation 2), it becomes an \textbf{algebra} over the field $\mathbb{R}$. The value of a signal $x$ at a given time $t$ is denoted by $x(t), x[t]$ or $x_t$ interchangeably in the literature.

1. Vector Addition: $(x + y)(t) := x(t) + y(t)$
2. Vector Multiplication: $(x \times y)(t) := x(t)y(t)$
3. Scalar Multiplication: $(\beta x)(t) := \beta(x(t)); \beta \in \mathbb{R}$

In this paper, we will consider the case of complex-valued \textit{discrete time signals} ($x: \mathbb{Z} \to \mathbb{C}$), but there are many analogies between the discrete and continuous cases. You can embed discrete signals in the continuous in several convenient ways. If you want to turn a continuous time signal into a discrete time one, the most common way is called \textit{sampling}. While real-valued signals represent physical quantities, complex signals can also have physical interpretation.

Example DT signals:

<table>
<thead>
<tr>
<th>Impulse/Delta Function</th>
<th>Unit Step</th>
<th>Exponential</th>
<th>Sinusoid</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta(t) := \begin{cases} 1 &amp; t = 0 \ 0 &amp; \text{otherwise} \end{cases}$</td>
<td>$u(t) := \begin{cases} 1 &amp; t \geq 0 \ 0 &amp; \text{otherwise} \end{cases}$</td>
<td>$x(t) = e^{st}; s \in \mathbb{R}$</td>
<td>$y(t) := \sin \left(\frac{\pi t}{2}\right)$</td>
</tr>
</tbody>
</table>

A \textbf{filter} (also called a \textbf{system}) is a mapping between signals. You can imagine it like a box that takes an input signal on one side, and outputs another signal on the other side (Figure 1). We can represent a filter $H$ with input signal $x$ and output signal $y$. Since we know that signals form a vector space, we are interested in \textbf{linear} filters that preserve that algebraic structure.
Examples of Filters: Amplitude squaring: $Ax := x^2$. Polynomial Filter $Bx := \sum_j a_j x^j$, Switch Filter (output is switched on at time $0$): $Sx := x \times u$.

The delay operator $\Delta_k$ is an important filter and can be defined as $\Delta_k [x] := x[t - k]$, where $k$ is any integer that we want to delay our signal by. A filter $H$ is called time-invariant (TI) if it commutes with the delay operator ($\Delta_k H = H \Delta_k$).

<table>
<thead>
<tr>
<th>Filter</th>
<th>$x \mapsto x^2$</th>
<th>$x \mapsto \sum_j a_j x^j$</th>
<th>$\Delta_k$</th>
<th>$x \mapsto u \times x$</th>
<th>$x \mapsto (\Delta_1 + \Delta_{-1})x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear?</td>
<td>NO</td>
<td>$a_j = \delta_j^2$</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>TI?</td>
<td>OK</td>
<td>OK</td>
<td>OK</td>
<td>NO</td>
<td>OK</td>
</tr>
</tbody>
</table>

We will be interested in how linear time-invariant (LTI) filters interact with DT signals, but first a proposition.

**Proposition 1.1 (Delta Decomposition):** Every DT signal can be written as a series of $\Delta_k$ and $\delta$.

$$x(t) \doteq x_t = \sum_{k \in \mathbb{Z}} x_k \delta_{t-k} = \sum_k x_k \Delta_k \delta_t \blacksquare$$

We can define this operation as discrete convolution $(x * \delta)(t) := \sum_k x_k \delta_{t-k} = \sum_k x_k \Delta_k \delta_t$. We can show convolution is commutative using a change of variables, i.e. $x * y = y * x$. By examining the proposition we see that $x = x * \delta = \delta * x$, thus $\delta$ is the identity of $\ast$.

**Theorem 1.2:** The action of an LTI filter can be represented as a convolution.

**Proof:** Let $H$ be an LTI filter. Applying $H$ to a discrete signal $x$ yields

$$H x_t = H \left( \sum_{k \in \mathbb{Z}} x_k \Delta_k \delta_t \right) = \sum_{k \in \mathbb{Z}} x_k H(\Delta_k \delta_t) = \sum_{k \in \mathbb{Z}} x_k \Delta_k (H\delta_t) = \sum_{k \in \mathbb{Z}} x_k \Delta_k h_t = (x * h)(t) \blacksquare$$

We define the impulse response of an LTI filter $H$ to be $h(t) := H\delta(t)$. The impulse response can be thought of as the “echoing” of a single impulse at the input. Delta decomposition tells us that every DT signal is like a train of scaled and delayed impulses, so discrete convolution is liked adding up the echoes of each impulse in the input signal! A filter is called causal whenever $h(t) = u(t) \times h(t)$. Intuitively, causal echoes only exist after the impulse appears, and so any part of the echo before time $t = 0$ must be 0!

Computing the action of an LTI filter with a convolution is feasible, and there are many digital systems in audio and image editing that do exactly this, but the issue is that a convolution can
be somewhat time-consuming. If you compute the action of multiple filters sequentially, this becomes computationally exhausting. Ideally, we will find a way to compose filters more conveniently, and we can do this by borrowing techniques from functional analysis.

**Lemma 1.3:** For any LTI filter, exponential functions are eigenvectors.

**Proof:** Let \( H \) be an LTI filter and \( s \in \mathbb{C} \). We can show that

\[
H(e^{st}) = \sum_{k \in \mathbb{Z}} e^{s(t-k)} h(k) = e^{st} \left( \sum_{k \in \mathbb{Z}} e^{-sk} h(k) \right) = e^{st} \lambda(s) \]

The lemma tells us that exponential functions are **eigensignals** of an arbitrary LTI filter. As it turns out, the family of complex-exponential signals \( \{ e^{i\omega t}; \omega \in \mathbb{R} \} \) form a basis of the DT signal space. The procedure for calculating the coordinates of a signal in the complex-exponential basis is called the **Fourier Transform**. We will use a version called the **Discrete Time Fourier Transform (DTFT)** which is inherited from the theory of continuous time signals. The coordinate of a signal \( x \) for any frequency \( \omega \) is given by a function \( \hat{x}(\omega) \) which is calculated by another sum

\[
\hat{x}(\omega) := \sum_{t \in \mathbb{Z}} x(t) e^{-i\omega t} = \text{DTFT}\{x\}
\]

To recover the time-domain signal, we use the inverse DTFT.

\[
x(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{x}(\omega) e^{i\omega t} d\omega = \text{DTFT}^{-1}\{x\}
\]

The new signal \( \hat{x} \) is said to be the **frequency-domain** representation of the (discrete) time domain signal. It should be noted that every real-valued signal can be represented with complex-exponentials in the same way, because it’s Fourier decomposition simplifies into sines and cosines (e.g. \( \sin(\omega t) = \frac{e^{i\omega t} - e^{-i\omega t}}{2i} \)). We are now ready to state the main result.

**Big Theorem 1.4 (Convolution Theorem of Sequences):** Convolution of signals in the time domain is multiplication of signals in the frequency domain. That is,

\[
\text{DTFT}\{h \ast x\} \equiv \hat{h} \times \hat{x}
\]

**Proof:** Let \( x, h \) be DT signals with frequency-domain representations \( \hat{x}, \hat{h} \) respectively. If we let \( r := t - k \), we can show that

\[
\text{DTFT}\{x \ast h\} = \sum_{t \in \mathbb{Z}} (x \ast h)_t e^{-i\omega t} = \sum_{t \in \mathbb{Z}} \sum_{k \in \mathbb{Z}} x_k h_{t-k} e^{-i\omega (t-k+k)} = \sum_{r \in \mathbb{Z}} \sum_{k \in \mathbb{Z}} x_k h_r e^{-i\omega r - i\omega k}
\]

\[
= \sum_{r \in \mathbb{Z}} \sum_{k \in \mathbb{Z}} x_k e^{-i\omega k} h_r e^{-i\omega r} = \sum_{k \in \mathbb{Z}} x_k e^{-i\omega k} \left( \sum_{r \in \mathbb{Z}} h_r e^{-i\omega r} \right) = \hat{x} \times \hat{h} \]

\[\square\]
**Corollary I**: The action of any LTI filter on a discrete time signal can be computed by multiplication in the frequency-domain, and then applying the inverse DTFT:

\[
\text{DTFT}^{-1}\{\hat{h} \times \hat{x}\} = \text{DTFT}^{-1}\{\text{DTFT}\{h\} \times \text{DTFT}\{x\}\} = \text{DTFT}^{-1}\{\text{DTFT}\{h \ast x\}\} = h \ast x = Hx
\]

The frequency-domain representation of a filter is called the **transfer function** of \( H \) and is given by the DTFT of the filter’s impulse response.

**Corollary II**: The composition of two LTI filters can be computed by multiplication of transfer functions.

**Proof**: For any LTI filters \( G, H \), with frequency responses \( \hat{g}, \hat{h} \) and any DT signal \( x \), we have that

\[
(G \circ H)x = G(Hx) = \text{DTFT}^{-1}\{\hat{g} \times \text{DTFT}\{Hx\}\} = \text{DTFT}^{-1}\{\hat{g} \times \hat{h} \times \hat{x}\}
\]

**Corollary III**: Any complex network of LTI filters can be represented as one filter, whose entire transfer function is an algebraic combination of the component filters.

**Example: Networks of filters**. The flowchart below describes a network of filters which is a familiar example in control theory. The basic idea is that the output signal \( y \) is reused as an input to the system in a process called **feedback**. Under certain conditions, the system is stable, but it depends deeply on our choice of filter \( H \) and the nature of the input signal \( x \).

![Flowchart](image)

Despite the presence of feedback, we can still describe the signal as a single filter with input signal \( x \) and output signal \( y \), denoted \( y = Qx \). If we want the transfer function of the composite filter \( Q \), we can use algebra to show that \( \hat{y} = \hat{q} \hat{x} = \left( \frac{\hat{g} \hat{p}}{1 + \hat{g} \hat{p} \hat{h}} \right) \hat{x} \).

From here, you can continue to generalize the description of CT and DT signals in a few ways.

**Multiple signals**: In this paper, we considered only filters with a single input and single output (SISO). In general, a multi-input multi-output LTI filter (MIMO) is described by a matrix of transfer functions that have familiar properties that we study in linear algebra.

**Irregular domains**: Image processing works with signals where \( T = [0,a] \times [0, b] \subset \mathbb{Z}^2 \) and literally describes filters that we use in applications like Photoshop. Graph signal processing is when \( T \) is the vertex set of a graph, and we generalize the idea of time-invariant to shift-invariance (permutations of graph vertices that preserve the topology).

**Applications**: DSP is used extensively in robotic control systems, digital communication, audio engineering, image processing, electrical circuit analysis, finance and machine learning.