INTRODUCING SIGNALS AND SYSTEMS – THE BERKELEY APPROACH

Edward A. Lee and Pravin Varaiya

eal@eecs.berkeley.edu, varaiya@eecs.berkeley.edu
Electrical Engineering & Computer Science
University of California, Berkeley

First Signal Processing Education Workshop, Hunt, Texas, October 15 - 18, 2000

ABSTRACT

At Berkeley, we have recently revised the curriculum in EECS to have a common core in electrical engineering and computer science, where the common core reflects the contemporary reality of a digital, networked, computational world. Part of this curriculum revision is a new introductory course that reflects the signals and systems side of electrical engineering. The course is aimed as sophomores, although it is taken by significant numbers of students at all levels, from freshmen to seniors. The course is designed to be as relevant to computer scientists as to electrical engineers. Thus, it does not have a circuits prerequisite, and does not use circuits as an illustration of systems. Instead, it motivates signals and systems through media, primarily sound and images, with occasional references to radio and electrical signals. The course presents a unified view of signals and systems that is much broader than the traditional focus on linear-time-invariant systems. It uses sets and functions on sets as a unified notation, and defines discrete-time and continuous-time signals, as well as event sequences, images, and video within this notation. Systems are functions whose domain and range are sets of functions.

1. INTRODUCTION

Like so many engineering schools, Berkeley used to have an introductory course entitled "Introduction to Electrical Engineering" that was about analog circuits. Many of the faculty felt that this was no longer a good introduction to our discipline, since it seemed to talk more about the origins of the discipline that its contemporary reality. We undertook to "digitize" the curriculum, first by making the introductory circuits course more digital, and second by designing a new course that introduces the systems side of the discipline. This paper is about this new course.

A great deal has changed in our discipline since the introductory circuits courses were first designed. Whereas circuits used to be the heart of the discipline, it is arguable that today it is the analytical techniques that emerged from circuit theory that are the heart of the discipline. The circuits themselves have become an area of specialization, albeit a very important one. Whereas a signal used to be primarily a voltage that varies over time, an electromagnetic waveform, or an acoustic waveform, now it is likely to be a sequence of discrete message. Whereas the state of a system used to be represented by variables in a differential equation, now it is likely to be represented by the registers and memory of a computer. Whereas a system used to be well-modeled by a linear time-invariant transfer function, now it is likely to be a computation in a Turing-complete computation engine.

Fundamental limits have also changed. Although we still face thermal noise and the speed of light, we are likely to encounter other limits before we get to these, such as complexity, computability, and chaos. The mathematical basis for the discipline has also shifted. Although we still need calculus and differential equations, we more frequently need discrete math, set theory, and mathematical logic.

The new course is about signals and systems, but with a twist. First, the course embraces a computational view of signals and systems. Thus, like other innovative introductory courses in EE, it puts more emphasis on discrete-time modeling than on continuous time. In this regard, we have been heavily influenced by the excellent and innovative textbooks by McClellan, Schafer and Yoder [1] and by Steiglitz [4] that also introduce signals and systems emphasizing discrete-time models and applications. The themes of the course are

- The connection between imperative (computational) and declarative (mathematical) descriptions of signals and systems.
- The use of sets and functions as a universal language for declarative descriptions of signals and systems.
- State machines and frequency domain analysis as complementary tools for designing and analyzing signals and systems.
- Early and frequent discussion of applications.

On the latter issue, it has been our objective that every one of the 45 lectures in this course touch upon at least one real-world application. Although we have fallen somewhat short on this goal, we do much better in this regard than most other signals and systems courses.

2. COURSE CONTENT

The course begins by describing signals as functions, focusing on characterizing the domain and the range. Systems are also described as functions, but now the domain and range are sets of signals. Characterizing these functions is *the* topic of this course. We begin by describing systems using the notion of state, first using automata theory and then progressing to linear systems. Frequency domain concepts are introduced as a complementary toolset, different from that of state machines, and much more powerful when applicable. Frequency decomposition of signals is introduced using psychoacoustics, and gradually developed until all four Fourier transforms (the Fourier series, the Fourier transform, the discrete-time Fourier transform, and the DFT) have been described. We linger on the first of these, the Fourier series, since it is conceptually the easiest, and then quickly present the others as simple generalizations of the Fourier series. Finally, the course closes by using these concepts to study sampling and aliasing.

The course is designed for Berkeley's 15 week semester, and has a accompanying textbook and web page [3].

Week 1 – Signals as Functions. The first week motivates forthcoming material by illustrating how signals can be modeled abstractly as functions on sets. The emphasis is on characterizing the domain and the range, not on characterizing the function itself. The startup sequence of a voiceband data modem is used as an illustration, with a supporting applet (see figure 1) that plays the very familiar sound of the startup handshake of V32.bis modem, and examines the waveform in both the time and frequency domain. The domain and range of the following signal types is given: sound, images, position in space, angles of a robot arm, binary sequences, word sequences, and event sequences.

Week 2 – Systems as Functions. The second week introduces systems as functions that map functions (signals) into functions (signals). Again, it should focus not on how the function is defined, but rather on what is the domain and range. Block diagrams are defined as a visual syntax for composing functions. Applications considered are DTMF signaling, modems, digital voice, and audio storage and retrieval. These all share the property that systems are required to convert domains of functions. For example, to transmit a digital signal through the telephone system, the digital signal has to be converted into a signal in the domain of the telephone system (i.e., a bandlimited audio signal).

Week 3 – State. Week 3 is when the students get seriously into Matlab (see [2]). The first lecture in this week is therefore devoted to the problem of relating declarative and imperative descriptions of signals and systems. This sets the framework for making the intellectual connection between the labs and the mathematics.

The rest of the week is devoted to introducing the notion of state and state machines. State machines are described by a function *update* that, given the current state and input, returns the new state and output. In anticipation of composing state machines, the concept of *stuttering* is introduced. This is a slightly difficult concept to introduce at this time because it has no utility until you compose state machines. But introducing it now means that we don't have to change the rules later when we compose machines.

Week 4 – Nondeterminism and Equivalence. The fourth week deals with nondeterminism and equivalence in state machines. Equivalence is based on the notion of simulation, so simulation relations and bisimulation are defined for both deterministic and nondeterministic machines. These are used to explain that two state machines may be equivalent even if they have a different number of states, and that one state machine may be an abstraction of another, in that it has all input/output behaviors of the other (and then some).

Week 5 – Composition. This week is devoted to composition of state machines. The deep concepts are synchrony, which gives a rigorous semantics to block diagrams, and feedback. The most useful concept to help subsequent material is that feedback loops with delays are always well formed.

Week 6 – Linear Systems. We consider linear systems as state machines where the state is a vector of reals. Difference equations and differential equations are shown to describe such state machines. The notions of linearity and superposition are introduced.

Week 7 – Response of Linear Systems. Matrices and vectors are used to compactly describe systems with linear and time-invariant state updates. Impulses and impulse response are introduced. The deep concept here is linearity, and the benefits it brings, specifically being able to write the state and output response as a convolution sum.

We begin to develop frequency domain concepts, using musical notes as a way to introduce the idea that signals can be given as sums of sinusoids.

Week 8 – Frequency Domain. This week introduces frequency domain concepts and the Fourier series. Periodic signals are defined, and Fourier series coefficients are calculated by inspection for certain signals. The frequency domain decomposition is motivated by the linearity of systems considered last week (using the superposition principle), and by psychoacoustics and music.

Week 9 – Frequency Response. In this week, we consider linear, time-invariant (LTI) systems, and introduce the notion of frequency response. We show that a complex exponential is an eigenfunction of an LTI system. The Fourier series is redone using complex exponentials, and frequency response is defined in terms of this Fourier series, for periodic inputs.

Week 10 – Filtering. The use of complex exponentials is further explored, and phasors and negative frequencies are discussed. The concept of filtering is introduced, with the terms lowpass, bandpass, and highpass, with applications to audio and images. Composition of LTI systems is introduced, with a light treatment of feedback.

Week 11 – Convolution. We describe signals as sums of weighted impulses and then use linearity and time invariance to derive convolution. FIR systems are introduced,

with a moving average being the prime example. Implementation of FIR systems in software and hardware is discussed, and signal flow graphs are introduced. Causality is defined.

Week 12 – Fourier Transforms. We relate frequency response and convolution, building the bridge between time and frequency domain views of systems. We introduce the DTFT and the continuous-time Fourier transform and derive various properties. These transforms are described as generalizations of the Fourier series where the signal need not be be periodic.

Week 13 – Sampling and Aliasing. We discuss sampling and aliasing as a major application of Fourier analysis techniques. Emphasis is on intuitive understanding of aliasing and its relationship to the periodicity of the DTFT. The Nyquist-Shannon sampling theorem is stated and related to this intuition, but its proof is not emphasized.

Week 14 – Filter Design. This week begins a review that focuses on how to apply the techniques of the course in practice. Filter design is considered with the objective of illustrating how frequency response applies to real problems, and with the objective of enabling educated use of filter design software. The modem startup sequence example is considered again in some detail, zeroing in on detection of the answer tone to illustrate design tradeoffs.

Week 15 – Comprehensive Examples. This week develops applications that combine techniques of the course. The precise topics depend on the interests and expertise of the instructors, but we have specifically covered the following:

- Speech analysis and synthesis, using a historical Bell Labs recording of the Voder and Vocoder from 1939 and 1940 respectively, and explaining how the methods illustrated there (parametric modeling) are used in today's digital cellular telephones.
- Digital audio, with emphasis on encoding techniques such as MP3. Psychoacoustic concepts such as perceptual masking are related to the frequency domain ideas in the course.
- Vehicle automation, with emphasis on feedback control systems for automated highways. The use of discrete magnets in the road and sensors on the vehicles provides a superb illustration of the risks of aliasing.

2.1. Discussion

The first few times we offered this course, automata appeared after frequency domain concepts. The new ordering, however, is far better. In particular, it introduces mathematical concepts gradually. Specifically, the mathematical concepts on which the course relies are, sets and functions, matrix multiplication, complex numbers, and series and integrals. In particular, note that although students need to be comfortable with matrix multiplication, most of linear algebra is not required. We never mention an eigenvalue nor a matrix inverse, for example. The calculus required is also quite simple. The few exercises in the text that require calculus provide any integration formulas that a student might otherwise look up. Although series figure prominently, we only lightly touch on convergence, raising but not resolving the issue.

Some instructors may be tempted to omit the material on automata. We advise strongly against this. First, it gets students used to formally characterizing signals and systems in the context of a *much simpler* framework than linear systems. Most students find this material quite easy. Moreover, the methods apply much more broadly than frequency domain analysis, which applies primarily to LTI systems. **Most systems are not LTI.** Thus, inclusion of this material properly reflects the breadth of electrical engineering, which includes such specialties as data networks, which have little to with LTI systems. Even in specializations that heavily leverage frequency domain concepts, such as signal processing and communications, practitioners find that a huge fraction of their design effort deals with control logic and software-based services. Regrettably, classically trained electrical engineers harbor the misapprehension that these parts of their work are not compatible with rigor. This is wrong.

3. USE OF THE WEB

Our version of this course makes extensive use of the web. Detailed notes for every lecture are on line, many with illustrative applets that include interactive manipulation of sounds and images (see figure 1). The web content is used in lecture in order to leverage the applets, but mathematical concepts are developed in the traditional way, on the blackboard. This seems to lead to better pacing.

A major difficulty with the web materials for this course is that web has no widely available mechanism for displayed typeset mathematics. Use of GIF images, as for instance provided by LaTeXto HTML converters, is inadequate because these images are too small to be used in lecture. Moreover, they do not print well. Our (interim) solution here is to use the symbol font with ordinary HTML to render mathematics. Regrettably, this means that these web pages cannot be viewed on Unix or Apple computers without font errors. Apple computers have a symbol font, but regrettably, it is a different font from the one provided by Microsoft, and so symbols get remapped to other symbols, resulting in nonsensical expressions.

We experimented with using applets to display mathematics (which could then be specified, for example, in MathML), but the various implementations of this that we tried were hopelessly slow. A page with more than one or two mathematical expressions takes too long to render.

We find it amazing that the web, which was largely developed by the technical community, so poorly serves that technical community. We hope that browsers will provide native rendering of MathML soon so that our pages do are not limited to Microsoft platforms.

The course has a tightly integrated software lab, based on Matlab and Simulink, that is described in [2].

4. NOTATION

The notation we use is somewhat unusual when compared to standard notation in the vast majority of texts on signals and systems. However, we believe that the standard notation is seriously flawed. As a community, we have been able to get away with it for many years because signals and systems dealt only with continuous-time LTI systems. But to be useful, the discipline must be much broader now. Our specific complaints about the standard notation include:

4.1. Domains and Ranges

It is all too common to use the form of the argument of a function to define the function. For example, $x(n)$ is a discrete-time signal, while $x(t)$ is a continuous-time signal. This leads to mathematical nonsense like the $x(n) = x(nT)$ to define sampling. Similarly, many authors use ω for frequency in radians per second (unnormalized) and Ω for frequency in radians per sample (normalized). This means that $X(\Omega) \neq X(\omega)$ even when $\Omega = \omega$. The same problem arises when using the form $X(j\omega)$ for the continuous-time Fourier transform and $X(e^{j\omega})$ for the discrete-time Fourier transform. Worse, these latter forms are used specifically to establish the relationship to the Laplace and Z transforms. So $X(j\omega) = X(s)$ when $s = j\omega$, but $X(j\omega) \neq X(e^{j\omega})$ when $e^{j\omega} = j\omega$.

The intent in using the form of the argument is to indicate what the domain of the function is. However, the form of the argument is not the best way to do this. Instead, we treat the domain of a function as an integral part of its definition. Thus, for example, a discrete-time (real-valued) signal is a function x: *Ints* $\rightarrow$ *Reals*, and it has a discrete-time Fourier transform that is also a function X: *Reals* $\rightarrow$ *Comps*. The DTFT itself is a function whose domain and range are sets of functions

$$\mathit{DTFT}\colon [\mathit{Ints} \rightarrow \mathit{Reals}] \rightarrow [\mathit{Reals} \rightarrow \mathit{Comps}].$$

Thus, we can write $X = \mathit{DTFT}(x)$.

4.2. Functions as Values

Most texts call the expression $x(t)$ a function. A better interpretation is that $x(t)$ is an element in the range of the function x. The difficulty with the former interpretation becomes obvious when talking about systems. Many texts pay lip service to the notion that a system is a function by introducing a notation like $y(t) = T(x(t))$. This makes no distinction between the value of the function at t and the function y itself.

Why does this matter? Consider our favorite type of system, an LTI system. We write $y(t) = x(t) * h(t)$ to indicate convolution. Under any reasonable interpretation of mathematics, this would seem to imply that $y(t-\tau) = x(t-\tau)*h(t-\tau)$. But it is not so! How is a student supposed to conclude that $y(t-2\tau) = x(t-\tau) * h(t-\tau)$? This sort of sloppy notation could easily undermine the students' confidence in mathematics.

In our notation, a function is the element of a set of functions, just as its value for a given element in the domain is an element of its range. Convolution is a function whose domain is the cross product of two sets of functions. Continuous-time convolution, for example, is

$$\begin{aligned} \mathit{Convolution} \quad &: \quad [\mathit{Reals} \rightarrow \mathit{Reals}] \times [\mathit{Reals} \rightarrow \mathit{Reals}] \\ &\rightarrow \quad [\mathit{Reals} \rightarrow \mathit{Reals}]. \end{aligned}$$

We then introduce the notation $*$ as a shorthand,

$$x * y = \mathit{Convolution}(x, y),$$

and define the convolution function by

$$(x * y)(t) = \int_{-\infty}^{\infty} x(\tau) y(t-\tau) d\tau.$$

Note the careful parenthesization.

A major advantage of our notation is that it easily extends beyond LTI systems to the sorts of systems that inevitably arise in any real world application. For example, the events generated by the buttons of an audio component are a signal given as a function,

$$\mathit{Commands}\colon \mathit{Nats} \rightarrow \{\mathit{Rec}, \mathit{Play}, \mathit{Stop}, \mathit{FastFwd}, \mathit{Rewind}\},$$

where *Nats* is the set of natural numbers. This is now a signal! With traditional notation, it is a whole new animal.

4.3. Names of Functions

We have chosen to use long names for functions and variables when they have a concrete interpretation. Thus, instead of x we might use *Sound*. This follows a long-standing tradition in software, where readability is considerably improved by long names. By giving us a much richer set of names to use, this helps us avoid some of the pitfalls we cite above. For example, to define sampling of an audio signal, we might write

$$\mathit{SampledSound} = \mathit{Sampler}_T(\mathit{Sound}).$$

It also helps bridge the gap between realizations of systems (which are often software) and their mathematical models. How to manage and understand this gap is a major theme of our approach.

5. RESULTS

We have taught EECS 20 four times. In Spring of 2000, we had 227 students, representing all four class levels, with the

vast majority of the students being sophomores and juniors.[1] We conducted a survey and performed a detailed statistical analysis of the factors that affected the performance of students in the class. We present the highlights of this analysis here.[2]

We summarize the results as follows. First, we define **diligent students** to be those who responded to the survey (the data bear that up). We define **one grade level** to be the increment from B and B+, or B+ and A-, for example. We define **class standing** to be freshman, sophomore, junior, or senior. And we define **taking a class** to mean either concurrently or before taking this one.

We found the following Math classes to be helpful:

- **Math 53.** Multivariable calculus. Parametric equations and polar coordinates. Vectors in 2- and 3-dimensional Euclidean spaces. Partial derivatives. Multiple integrals. Vector calculus. Theorems of Green, Gauss, and Stokes.
- **Math 54.** Linear Algebra and Differential Equations. Basic linear algebra; matrix arithmetic and determinants. Vector spaces; inner product as spaces. Eigenvalues and eigenvectors; linear transformations. Homogeneous ordinary differential equations; first-order differential equations with constant coefficients. Fourier series and partial differential equations.
- **Math 55.** Discrete Mathematics. Logic, mathematical induction sets, relations, and functions. Introduction to graphs, elementary number theory, combinatorics, algebraic structures, discrete probability, theory, and statistics. Emphasis on topics of interest to students in computer science.

These classes are taken after a basic calculus sequence, and are not currently prerequisites for this course.

- Class standing had little significant effect on performance.
- On average, the GPA of students was neither raised nor lowered by this class.
- Students who attend lecture do better than those who do not.
- Taking at least one of Math 53, 54, and 55 significantly helps (by slightly more than one grade level).
- Taking Math 53 or Math 55 helps by half a grade level.
- Taking Math 54 helps by almost a grade level.
- Taking more than one of Math 53, 54, or 55 helps somewhat, but many students do well without this.
- Many students believe Math 54 should be a prerequisite.
- Computing classes have little effect on performance, assuming as was the case with this class, that all students have had some computing classes.
- Students do not believe that a computing class prerequisite is needed.

[1] At Berkeley, many of our students spend their first two years in junior colleges, where there is no comparable course. These students must take this course in their junior or senior year, after arriving at Berkeley.

[2] For details, see http://www.eecs.berkeley.edu/ eal/eecs20/adm/sp00/analysis.pdf.

6. CHALLENGES

Any instructor in electrical engineering is qualified to teach this class. Although some of the material, such as automata theory, might be initially unfamiliar, they are easy for an educated person to learn. Indeed, it is these nontraditional aspects of the course that the students find easiest. The harder parts of the course, primarily the Fourier transforms, are part of the traditional EE curriculum, and thus are familiar to instructors in this field. Although our notation might be initially unfamiliar, we believe that instructors will become quickly used to it, and will appreciate its strengths.

There is one challenging aspect to teaching this course, however. Since it emphasizes applications, an instructor needs to be current on these applications. For example, although MP3 is not explicitly discussed in the course, be assured that unless the students are asleep, they will ask about it. An instructor needs to know what it is. Other questions that arise are: What is oversampling? How do cable modems and DSL work? What is the difference between CDMA and TDMA? What is the difference between analog cellular and digital cellular? What does “cellular” mean? What is GIF? PNG? JPEG? MPEG? How do CDs work? DVDs? Of course, it is not possible to go into these in much detail, but an effective instructor must be able to connect at least a reasonable subset of these with the course material. This is what keeps the students interested, because it gives them deep insight into the world they live in. More traditional courses in signals and systems given them deep insight into a world they never knew existed, and which apparently has no connection to the one they live in.

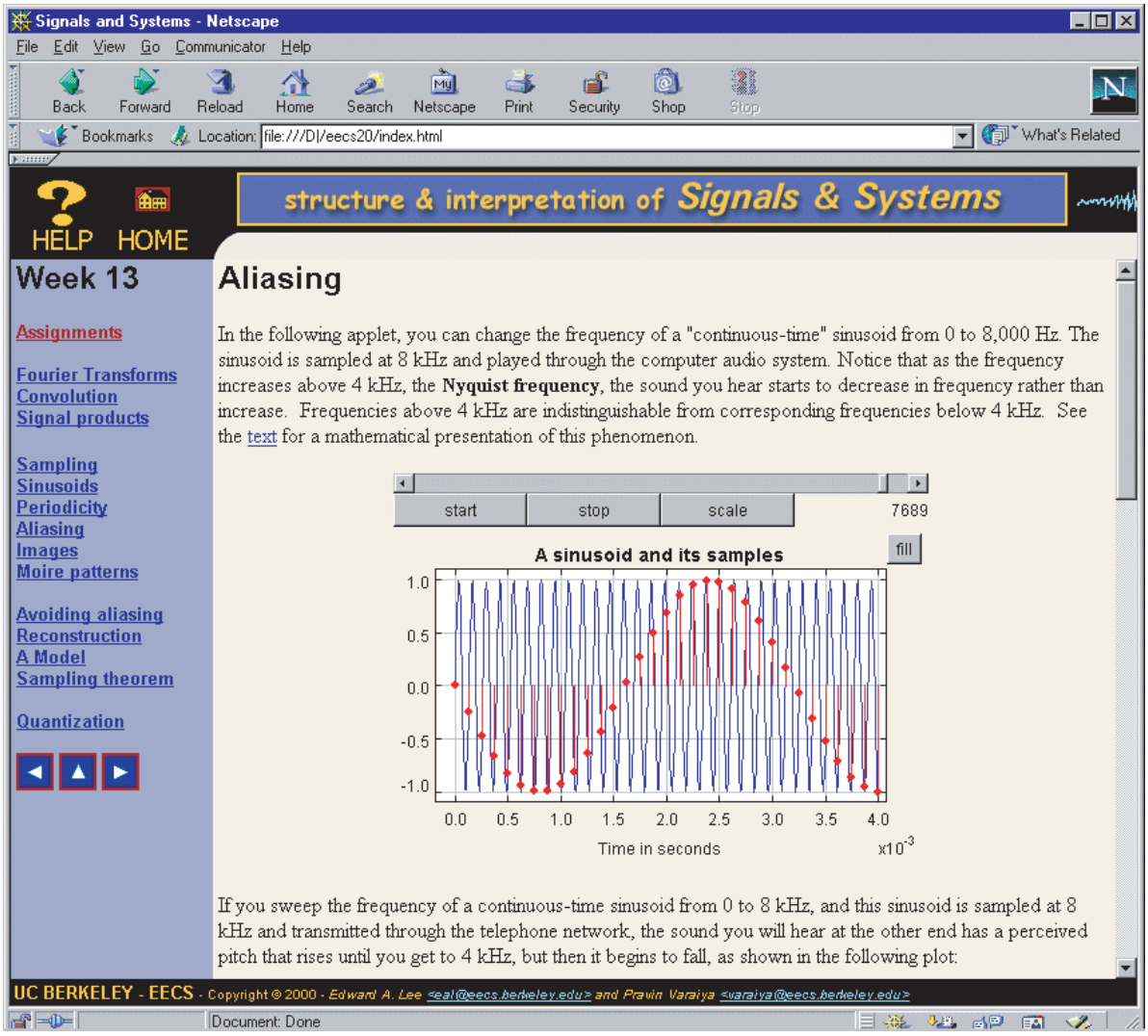

Figure 1. A web page showing aliasing in an interactive applet.

7. REFERENCES

[1] James H. McClellan, Ronald W. Schafer, and Mark A. Yoder, *DSP First: A Multimedia Approach*, Prentice-Hall, 1998.

[2] Edward A. Lee, "Designing a Relevant Lab for Introductory Signals and Systems," *Proc. of the First Signal Processing Education Workshop*, Hunt, Texas, October 15 - 18, 2000 (this volume).

[3] Edward A. Lee & Pravin Varaiya, *Structure and Interpretation of Signals and Systems*, textbook draft, 2000 (http://www.eecs.berkeley.edu/ eal/eecs20).

[4] Ken Steiglitz, *A DSP Primer: With Applications to Digital Audio and Computer Music*, Addison-Wesley, 1996.