# EG-247 Signals and Systems

# Lesson 1: Introducing the Module

# About this presentation

Dr Chris Jobling (c.p.jobling@swansea.ac.uk)

Digital Technium 123

Office Hours: Digital Technium 123, Mondays at 12:00 (noon)

You can view the notes for this presentation in HTML and PDF.

The source code of this presentation is available in Markdown format from GitHub: Introduction.md.

The GitHub repository EG-247 Resources also contains the source code for all the Matlab/Simulink examples and the Laboratory Exercises.

# Lesson 1

# Agenda

- Introduction to the Module
- The e-Learning System Unio
- Activities Supporting Week 1
- Introducing Signals and Systems
- Preparing for Lesson 2

# Introduction to the Module

Refer to the Blackboard site

- The Approach
- The Syllabus
- The Learning Outcomes
- The Reading List
- The Assessment

# The e-Learning System Unio

# Activities Supporting Week 1

- *Lesson 1*: This Introduction based on Chapter 1 of Signals and Systems for Dummies (SS4D) by Mark Wickert available as a free sample
- *Lesson 2*: Elementary Signals (Thursday 1:00 pm, Faraday J) based on Chapter 1 of Required Text Signals and Systems with Matlab Computing and Simulink Modeling by Stephen Karris (available as an e-Book)
- *Lab 1*: Matlab for Signals and Systems (Week on Friday, 13th February, 9-11 am, PC Lab 002 Digitial Technium)

# Introducing Signals and Systems

# Continuous-time signals

Continuous signals are represented mathematically by functions which vary continuously with time.

Sinusoidal signals (e.g. AC) are pretty fundamental in Electrical Engineering. The mathematical model of a sinusoidal signal is $x(t) = A\cos(2\pi f_0 t - \phi)$.

# Exercise

Take a moment to think about:

- What is $A$?
- What is $f_0$?
- What is $\phi$?
- What is the period $T_0$ of this signal?

Write down your answer in the notes pane.

# Gaining insight using computers

To help us answer these questions, let's use our Mathematical tools to plot a signal like this and explore it. The example we will use is from *Signals and Systems for Dummies* (SS4D: page 12):

$$3\cos(2\pi \cdot 2t - 3\pi/4)$$

# Wolfram|Alpha

Here's the link: `http://www.wolframalpha.com`

Paste this into the search box

```
plot 3 cos(2 pi 2 t - 3 pi/4)
```

# Matlab

In Matlab we would need to tackle this slightly differently. Here's the code:

```
t = linspace(0, 1, 100);
x = 3 * cos(2*pi*2*t - 3*pi/4);
plot(t,x)
title('A Sinusoidal Signal')
xlabel('Time t (s)')
ylabel('Amplitude')
grid
```

(I will run this code during the live session and we'll import the result into the lesson record.)

# Returning to the Question

Run the quiz!

# Exercises

- Use any or all of the computing tools that you have access to to explore other sinusoids. Change the values of the variables and explain what happens.
- Try adding sinusoids of different amplitudes and different frequencies together and see what happens.

# Continuous-time Systems

Systems operate on signals. In mathematical terms, a *system* is a function or an *operator*, $H\{\}$ that maps the input signal $x(t)$ to an output signal $y(t)$.

Mathematically we would write this:

$$y(t) = H\{x(t)\}.$$

# Example

An example of a continuous-time system is an electronic amplifier with a gain of 5 and level shift of 2: $y(t) = H\{x(t)\} = 5x(t) + 2$.

In this course, we will model such systems as block diagram models in Simulink.

# Block diagram model in Simulink

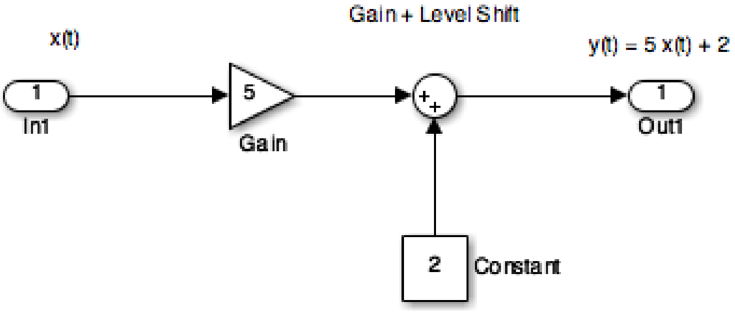


Figure 1: Simulink model of a a continuous-time system

The Simulink code can be downloaded from this file gain_level_shift.slx.

# Discrete-time Signals

Disrete-time signals are a function of a time index $n$. A discrete-time signal $x[n]$, unlike a continuous-time signal $x(t)$, is only defined at integer values of the independent variable $n$. This means that the signal is only active at specific periods of time. Discrete-time signals can be stored in computer memory.

# Example

Consider the following simple signal, a pulse sequence:

$$x[n] = \begin{cases} 5, & 0 \leq n < 10 \\ 0, & \text{otherwise} \end{cases}$$

We can plot this in Matlab as a *stem plot*

First define the function $x[n]$

```
% Define the function $x[n]$
function [ y ] = x( n )
  if n < 0 | n >= 10
    y = 0;
  else
    y = 5;
  end
end
```

Then set up $n$ and reserve some space for storing $x[n]$

```
% Define sample points
n = -15:18;
% Make space for the signal
xn = zeros(size(n));
```

Compute the signal

```
% Compute the signal x[n]
for i = 1:length(xn)
  xn(i) = x(n(i));
end
```

Finally, plot the signal as a *stem* (or *lollipop*) plot

```
stem(n,xn)
axis([-15,18,0,6])
title('Stem Plot for a Discrete Signal')
xlabel('Sample n')
ylabel('Signal x[n]')
grid
```

We'll run this code and paste in the resulting plot during the session.

# Exercise

Draw a digital signal that represents your student number in some way. For example if your number was 765443, then you could generate a signal for which $x[n] = 0$ when $n < 7$, then $x[n] = 7$ for 7 periods, then $x[n] = 6$ for the next 6 periods, $x[n] = 5$ for 5 periods, and so on. The signal should return to 0 when the last digit has been transmitted.

To plot this on a computer you would need to transcribe $x[n]$ into an array and then use the "lollipop" plot to plot the data. You could just create the array by hand, but you could also create a Matlab or Python function if you would like a challenge.

# Discrete-time Systems

A discrete-time system, like its continuous-time counterpart, is a function, $H\{\}$, that maps the input $x[n]$ to the output $y[n] = H\{x[n]\}$. An example of a discrete-time system is the *two-tap* filter:

$$y[n] = H\{x[n]\} = \frac{3}{4}x[n] + \frac{1}{4}x[n-1]$$

The term *tap* denotes that output at time instant $n$ is formed from two time instants of the input, $n$ and $n1$.

Check out a block diagram of a two-tap filter system:

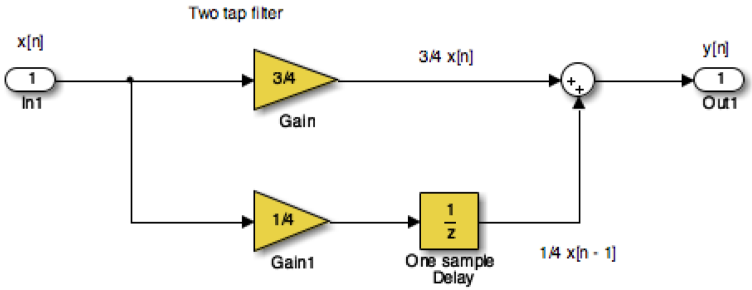


Figure 2: Block diagram of a two-tap filter system

This system is available as a Simulink model discrete_system.slx

# Signal Classifications

# Periodic

Signals that repeat over and over are said to be *periodic*. In mathematical terms, a signal is periodic if:

- *Continuous signal* $x(t + T) = x(t)$
- *Discrete signal* $x[n + N] = x[n]$

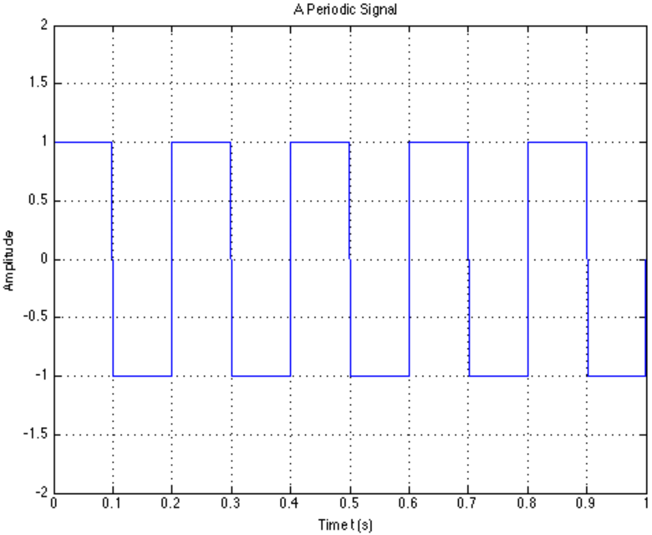


Figure 3: A square wave

*Matlab*

```
%% A Periodic signal (square wave)
t = linspace(0, 1, 500);
plot(t, square(2 * pi * 5 * t));
ylim([-2, 2]);
grid
title('A Periodic Signal')
xlabel('Time t (s)')
ylabel('Amplitude')
```

See: periodic.m

# Question

For the example we started with $x(t) = 2\cos(2\pi.2t + 3\pi/4)$. Say we sample the cosine wave at 20 times the frequency, what would the sampling period be and what would $N$ be for the sampled waveform?

# Write down your answer in the note area

# Aperiodic

Signals that are *deterministic* (completely determined functions of time) but not periodic are known as *aperiodic*. Point of view matters. If a signal occurs infrequently, you may view it as aperiodic.

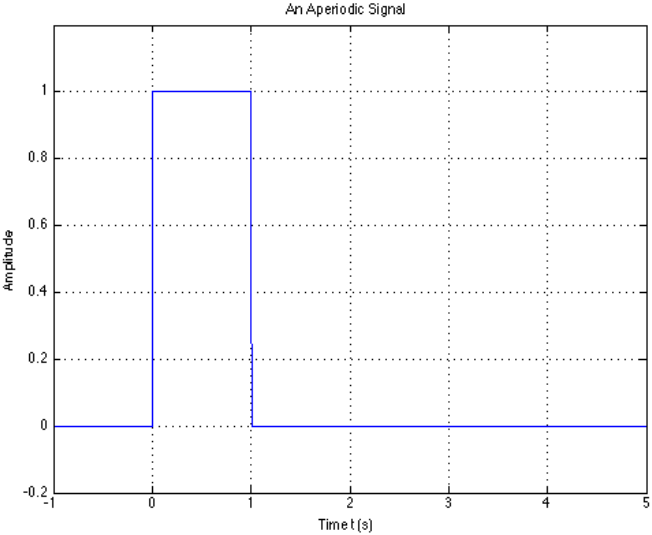


Figure 4: An aperiodic function

This is how we generate this aperiodic rectangular pulse of duration $\tau$ in Matlab:

*Matlab*

```
%% An aperiodic function
tau = 1
x = linspace(-1,5,1000);
y = rectangularPulse(0,tau,x);
plot(x,y)
ylim([-0.2,1.2])
grid
title('An Aperiodic Signal')
xlabel('Time t (s)')
ylabel('Amplitude')
```

See aperiodic.m

# Random

A signal is random if one or more signal attributes takes on unpredictable values in a probability sense.
Engineers working with communication receivers are concerned with random signals, especially noise.

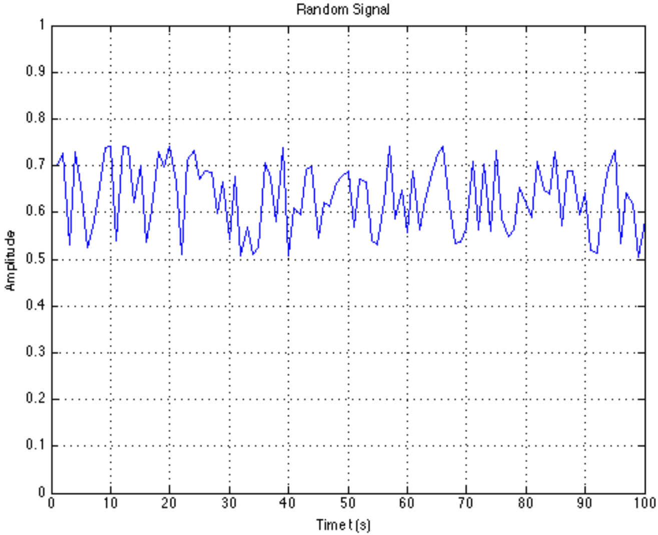


Figure 5: A random signal

*Matlab*

```
%% Plot a Random Signal
plot(0.5 + 0.25 * rand(100,1))
ylim([0,1])
grid
title('Random Signal')
xlabel('Time t (s)')
ylabel('Amplitude')
```

See: random.m

# Domains for Signals and Systems

Most of the signals we encounter on a daily basis reside in the time domain. They're functions of independent variable $t$ or $n$. But sometimes when you're working with continuous-time signals, you may need to transform away from the time domain ($t$) to either the frequency domain ($f$ or $\omega$) or the [Laplace] $s$-domain ($s$). Similarly, for discrete-time signals, you may need to transform from the discrete-time domain ($n$) to the frequency domain ($\hat{\omega}$ ) or the z-domain ($z$).

Systems, continuous and discrete, can also be transformed to the frequency and s- and z-domains, respectively. Signals can, in fact, be passed through systems in these alternative domains. When a signal is passed through a system in the frequency domain, for example, the frequency domain output signal can later be returned to the time domain and appear just as if the time- domain version of the system operated on the signal in the time domain.

This section briefly introduces the world of signals and systems in the frequency, s-, and z-domains. More on these domains will follow.

# Viewing Signals in the Frequency Domain

Consider the sum of a two-sinusoids signal

$$x(t) = \underbrace{A_1 \cos(2\pi f_1 t)}_{s_1} + \underbrace{A_2 \cos(2\pi f_2 t)}_{s_2}$$

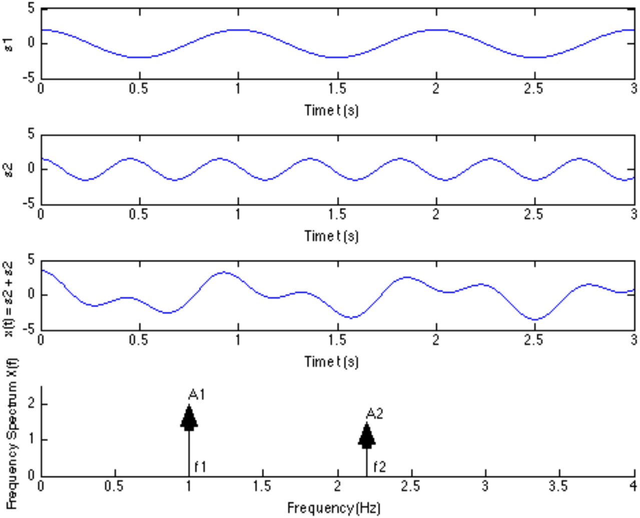


Figure 6: A two-sinusoids signal

Matlab code: two_sines.m

(This example relies on arrow.m by Erik Johnson available from the Matlab File Exchange.)

# Fourier Transform

We use the *Fourier transform* to move away from the time domain and into the frequency domain. To get back to the time domain, use the *inverse Fourier transform*. We will found out more about these transforms in this module.

# Laplace and Z-Transform Domains

From the time domain to the frequency domain, only one independent variable, $t \to f$, exists. When a signal is transformed to the $s$-domain, it becomes a function of a complex variable $s = \sigma + j\omega$. The two variables (real and imaginary parts) describe a location in the $s$-plane.

In addition to visualization properties, the $s$-domain reduces differential equation solving to algebraic manipulation. For discrete-time signals, the $z$-transform accomplishes the same thing, except differential equations are replaced by difference equations.

# Systems Thinking and Systems Design

See section **Testing Product Concepts with Behavioral Level Modeling** from Chapter 1 of SS4D (pages 18–20) and summarize this for yourself.

- We will use *behavioural modelling*
- We will rely on *abstraction*
- We work *top-down*
- We make use of *mathematics* and *mathematical software*.

# Familiar Signals and Systems

See Chapter 1 of SS4D for notes and details.

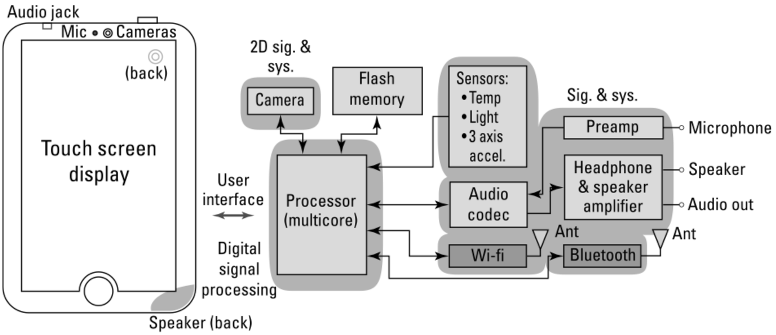


Figure 7: An MP3 player

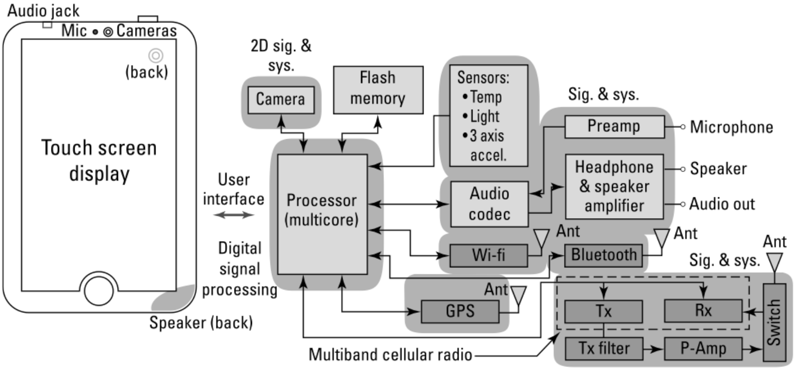


Figure 8: A smart phone

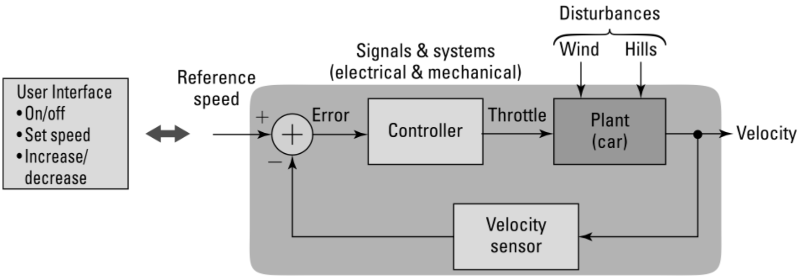


Figure 9: Cruise control – a control system

# Concluding Example: Some Basic Signal Operations

Consider a signal $f(t)$

$$x = f(t) = \begin{cases} 0 & : \ t < -1 \\ t+1 & : \ -1 \leq t \leq 1 \\ 0 & : \ t > 1 \end{cases}$$

Sketch this signal.

# Preparing for Second Session

- Do the homework (post-class activity)
- In the second lesson we will work through the hoemwork exercise then do some exercises based on Chapter 1 of Karris.