# Introducing Signals and Systems

An annotatable copy of these notes are to be found in the *_Content Library* of the **OneNote Class Notebook** for this class. You can also view the notes for this presentation as a webpage (HTML (http://nbviewer.ipython.org/github/cpjobling/EG-247-Resources/blob/master/introduction/Introduction.ipynb)) and as a downloadable PDF file (http://cpjobling.github.io/EG-247-Resources/introduction/Introduction.pdf).

Signals and Systems for Dummies (**SS4D**) provides a useful introduction to the topics that will be covered in this module and it is in the Reading List as a recommended text. I have based my first presentation on Chapter 1 which is available as a downloadable PDF (http://eu.dummies.com/store/product/Signals-and-Systems-For-Dummies.productCd-111847581X.html) from the publishers.

You should read Chapter 1 of SS4D in conjunction with these notes.

Note that Signals and Systems for Dummies is available in print and as an e-book for Kindle, Android (via Google Play) and Apple (via iTunes).

## Continuous-time signals

Continuous signals are representated 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 $x(t) = A\cos(2\pi f_0 t - \phi)$.

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

### 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 (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 shared white board.)

## Jupyter and Python

We can even use this document (Introduction.ipynb) (which is an Jupyter Notebook (http://ipython.org/notebook.html)) to compute the sinewave directly.

First we have to set things up.

In [ ]:

```
%matplotlib inline
from pylab import *
```

Then we run the plot. The instructions are similar to Matlab:

In [ ]:

```
t = np.linspace(0, 1, 100)
x = 3 * np.cos(2*pi * 3*t - 3*pi/4)

plot(t, x)
title('A Sinusoidal Signal ')
xlabel('Time t (s)')
ylabel('Amplitude')
grid()
```

### Returning to the Question

For the mathematical model of a sinusoidal signal $x(t) = A\cos(2\pi f_0 t - \phi)$ use your insight to answer these questions again:

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

Also, give the actual values, with units, for the variables used in the example $3\cos(2\pi \cdot 2t - 3\pi/4)$.

### Notes

- In communications and electronic signal processing, the frequency of sinusoidal signals us usually given in *cycles per second* or Hz.
- In mathematics, the frequency is always expressed in *radians per second*.
- In some courses, including later in this one and in EG-243 Control Systems, the frequency $2\pi f_0$ is often called the *natural frequency* and is usually written $\omega_n$.

### Try This Yourself

- Use any or all of 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:

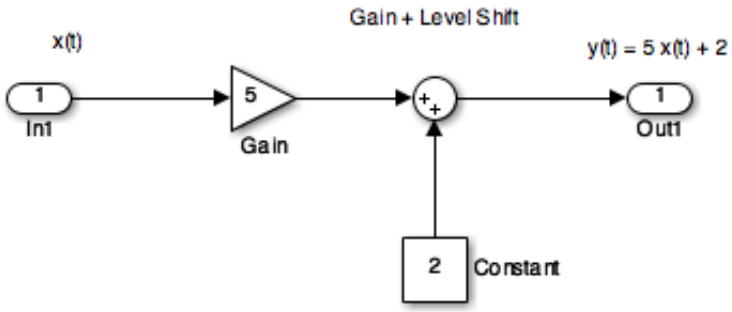
$$y(t) = H\{x(t)\}.$$

### 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




The Similink code can be downloaded from this file gain_level_shift.slx (matlab/gain_level_shift.slx).

## Demonstration

If the input to this system is replaced with a sinewave $x(t) = \sin(t)$ and the output with a scope, what do you think the output will be?

If you get a chance, try this in youself in Matlab and copy the result into your copy of these notes.

## 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:

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

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

First define the function $x[n]$

In [ ]:

```
def x(n):
    if n < 0 or n >= 10:
        return 0
    else:
        return 5
```

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

In [ ]:

```
n = np.arange(-15,18)
xn = zeros(size(n))
```

then compute the signal

In [ ]:

```
for i in range(0, size(n)):
    xn[i] = x(n[i])
```

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

In [ ]:

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

# In Matlab

The Matlab version is similar and is given in script file discrete.m (matlab/discrete.m) which uses function x.m (matlab/x.m).

### 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.

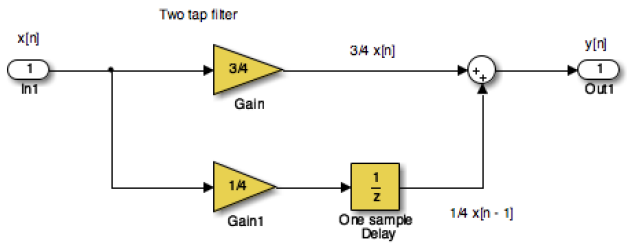
To plot this on a computer you would need to transcribe $x[n]$ into an array and then use the "lollipop" plot to plot the data. You could just create the array by hand, but you could also create a Matlab or Python function if you would like a challenge.

## 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 $n-1$. Check out a block diagram of a two-tap filter system:




This system is available as a Simulink model discrete_system.slx (matlab/discrete_system.slx)

In words, this system scales the present input by 3/4 and adds it to the past value of the input scaled by 1/4. The notion of the past input comes about because $x[n-1]$ is lagging one sample value behind $x[n]$. The term *filter* describes the output as an *averaging* of the present input and the previous input. *Averaging* is a form of filtering.

# Signal Classifications

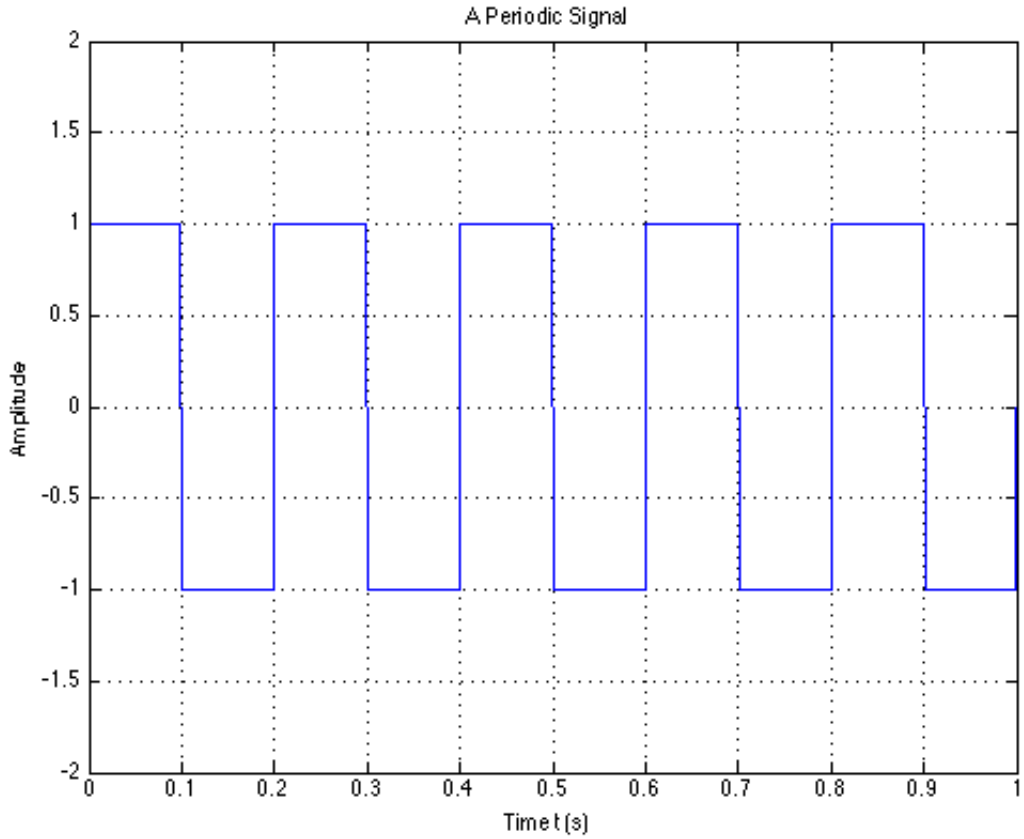
## Periodic

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* $x[n + N] = x[n]$

The smallest $T$ or $N$ for which the equality holds is the *signal period*.

The sinusoidal signal we saw earlier is periodic because of the $\mod 2\pi$ property of cosines. The signal of the sinusoid has period 0.5 seconds (s), which turns out to be the reciprocal of the frequency $1/f_0$ Hz.




This Square wave is a 5 Hz waveform sampled at 500 Hz for 1 second

$T =$?

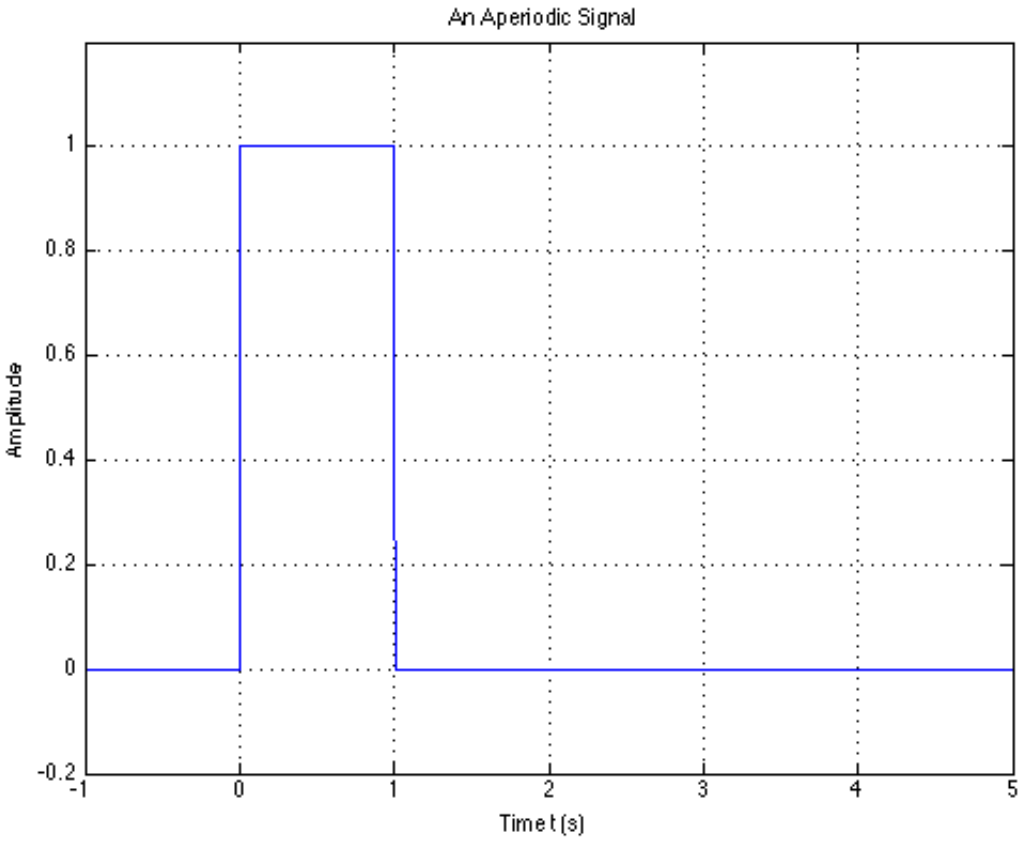
## Question

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?

**Your Answer**

## 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.



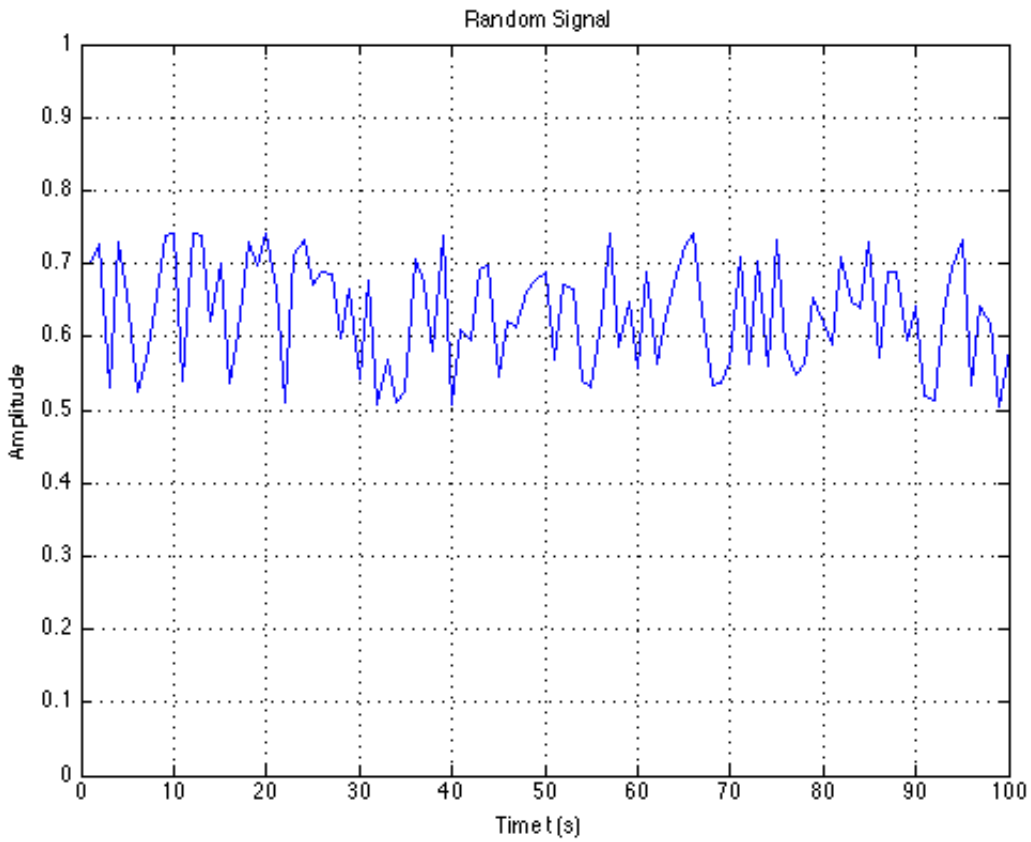
This is how we generate an aperiodic rectangular pulse of duration $\tau$ in Matlab:

```
%% 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 (matlab/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.



*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 (https://github.com/cpjobling/EG-247-Resources/blob/master/introduction/matlab/random.m)

# Domains for Signals and Systems

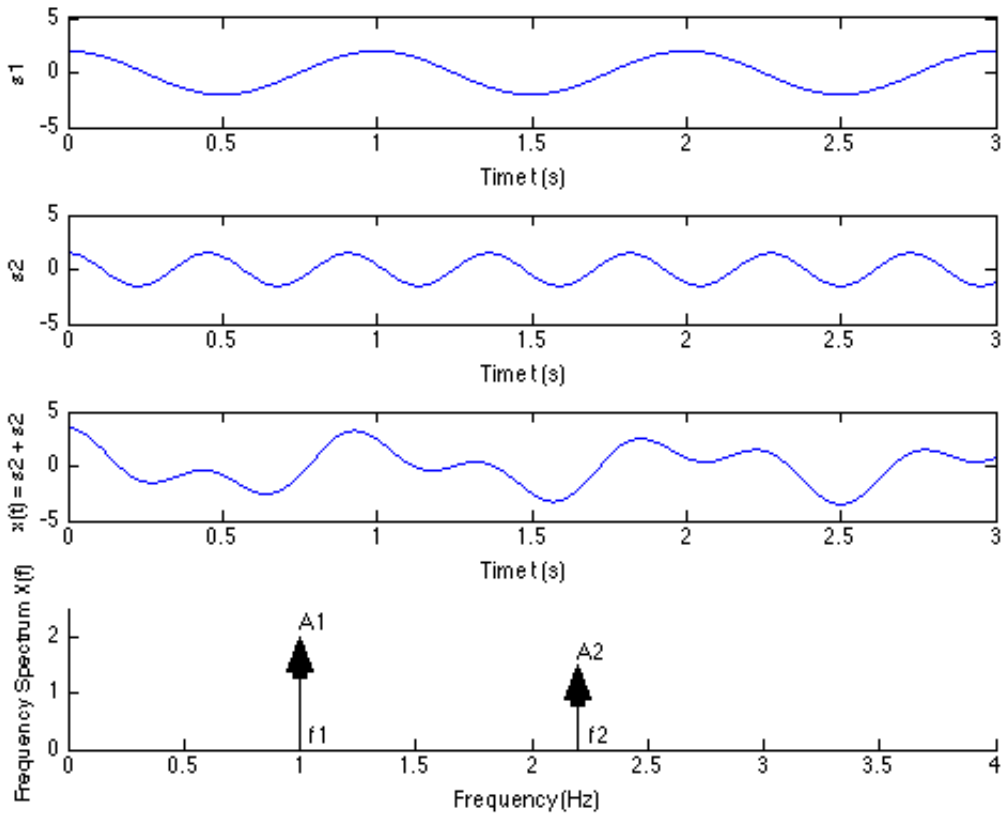
Most of the signals we encounter on a daily basis reside in the time domain. They're functions of independent variable $t$ or $n$. But sometimes when you're working with continuous-time signals, you may need to transform away from the time domain ($t$) to either the frequency domain ($f$ or $\omega$)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

Consider the sum of a two-sinusoids signal

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




Matlab code: two_sines.m (matlab/two_sines.m)

(This example relies on arrow.m (http://www.mathworks.com/matlabcentral/fileexchange/278-arrow-m) by Erik Johnson available from the Matlab File Exchange.)

## Viewing Signals in the Frequency Domain

The top waveform plot, denoted $s_1$, is a single sinusoid at frequency $f_1$ and peak amplitude $A_1$. The waveform repeats every period $T_1 = 1/f_1$. The second waveform plot, denoted $s_2$, is a single sinusoid at frequency $f_2 > f_1$ and peak amplitude $A_2 < A_1$. The sum signal, $s_1 + s_2$, in the time domain is a squiggly line (third waveform plot), but the amplitudes and frequencies (periods) of the sinusoids aren't clear here as they are in the first two plots. The frequency spectrum (bottom plot) reveals that $x(t)$ is composed of just two sinusoids, with both the frequencies and amplitudes discernible.

Think about tuning in a radio station. Stations are located at different center frequencies. The stations don't interfere with one another because they're separated from each other in the frequency domain. In the frequency spectrum plot, imagine that $f_1$ and $f_2$ are the signals from two radio stations, viewed in the frequency domain. You can design a receiving system to filter $s_1$ from $s_1 + s_2$. The filter is designed to pass $s_1$ and block $s_2$.

### 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, $t \rightarrow f$, 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 (http://www.dummies.com/store/product/Signals-and-Systems-For-Dummies.productCd-111847581X.html) (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 pages 21-23 of the free sample (Chapter 1) of SS4D (http://www.dummies.com/store/product/Signals-and-Systems-For-Dummies.productCd-111847581X.html) for notes and details.

# Concluding Example: Some Basic Signal Operations

Consider a signal

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

Sketch this signal.

## Homework

Think about the effect on this signal of applying the following basic signal operations:

- $2f(t)$
- $0.5f(t)$
- $f(2t)$
- $f(0.5t)$
- $-f(t)$
- $f(-t)$
- $-f(-t)$
- $f(t-1)$
- $f(t+1)$
- $-2f(-t+2)$

# Preparing for Second Lesson

In the second session we will work through the concluding exercise then do some exercises based on Chapter 1 of Karris (http://site.ebrary.com/lib/swansea/reader.action?docID=10547416&ppg=17).