

# Chapter 5

## Amplitude Modulation

### Contents

- Slide 1 Amplitude Modulation
- Slide 2 The Envelope and No Overmodulation
- Slide 3 Example for Single Tone Modulation
- Slide 4 Measuring the Modulation Index
- Slide 5 Transmitted vs. Message Power in  $s(t)$
- Slide 6 Powers in Single Tone Case (cont.)
- Slide 7 Spectrum of an AM Signal (cont.)
- Slide 8 Demodulating by Envelope Detection
- Slide 9 Square-Law Envelope Detector (cont.)
- Slide 10 Sampling Rate for Square-Law Detector
- Slide 11 Hilbert Transforms and Complex  
Envelope
- Slide 12 Hilbert Transforms (cont.)
- Slide 13 Pre-Envelope or Analytic Signal
- Slide 14 Spectrum of Pre-Envelope
- Slide 15 Pre-Envelope of AM Signal
- Slide 16 Envelope Detector Using the Hilbert  
Transform
  
- Slide 17 **Experiment 5.1 Making an AM**

## **Modulator**

- Slide 18** Experiment 5.1 (cont. 1)
- Slide 19** Experiment 5.1 (cont. 2)
  
- Slide 20** Capturing DSK Outputs for Plotting
- Slide 21** Code Segment to Allow Capture
- Slide 22** Setting Up CCS for Capture
- Slide 23** Setting Up CCS for Capture (cont.)
- Slide 24** An Easier Method of Sending Data from the DSK to the PC
- Slide 25** Using Windows XP Sound Recorder to Capture the DSK LINE OUT
- Slide 26** Using Windows XP Sound Recorder to Capture the DSK LINE OUT (cont.)
  
- Slide 27** **Experiment 5.2 Making a Square-Law Envelope Detector**
- Slide 28** Experiment 5.2 Square-Law Detector (cont.)
- Slide 29** Experiment 5.2.1 The Square-Law Detector Output with No Input Noise
- Slide 29** Experiment 5.2.2 The Square-Law Detector Output with Input Noise
- Slide 30** Experiment 5.5.2 Square-Law Detector with Input Noise

**Slide 31** Generating Gaussian Random Numbers

**Slide 32** Gaussian RV's (cont. 1)

**Slide 33** Gaussian RV's (cont. 2)

**Slide 34** Gaussian RV's (cont. 3)

**Slide 35** Measuring the Signal Power

**Slide 36** **Experiment 5.3 Envelope Detector Using the Hilbert Transform**

**Slide 37** Experiment 5.3 Envelope Detector Using the Hilbert Transform (cont. 1)

**Slide 38** Experiment 5.3 Envelope Detector Using the Hilbert Transform (cont. 2)

## Chapter 5 Amplitude Modulation

AM was the first widespread technique used in commercial radio broadcasting.

An AM signal has the mathematical form

$$s(t) = A_c[1 + k_a m(t)] \cos \omega_c t$$

where

- $m(t)$  is the *baseband message*.
- $c(t) = A_c \cos \omega_c t$  is called the *carrier wave*.
- The carrier frequency,  $f_c$ , should be larger than the highest spectral component in  $m(t)$ .
- The parameter  $k_a$  is a positive constant called the *amplitude sensitivity* of the modulator.

## The Envelope and No Overmodulation

- $e(t) = A_c|1 + k_a m(t)|$  is called the *envelope* of the AM signal. When  $f_c$  is large relative to the bandwidth of  $m(t)$ , the envelope is a smooth signal that passes through the positive peaks of  $s(t)$  and it can be viewed as *modulating* (changing) the amplitude of the carrier wave in a way related to  $m(t)$ .

### Condition for No Overmodulation

In standard AM broadcasting, the envelope should be positive, so

$$e(t) = A_c[1 + k_a m(t)] \geq 0 \quad \text{for all } t$$

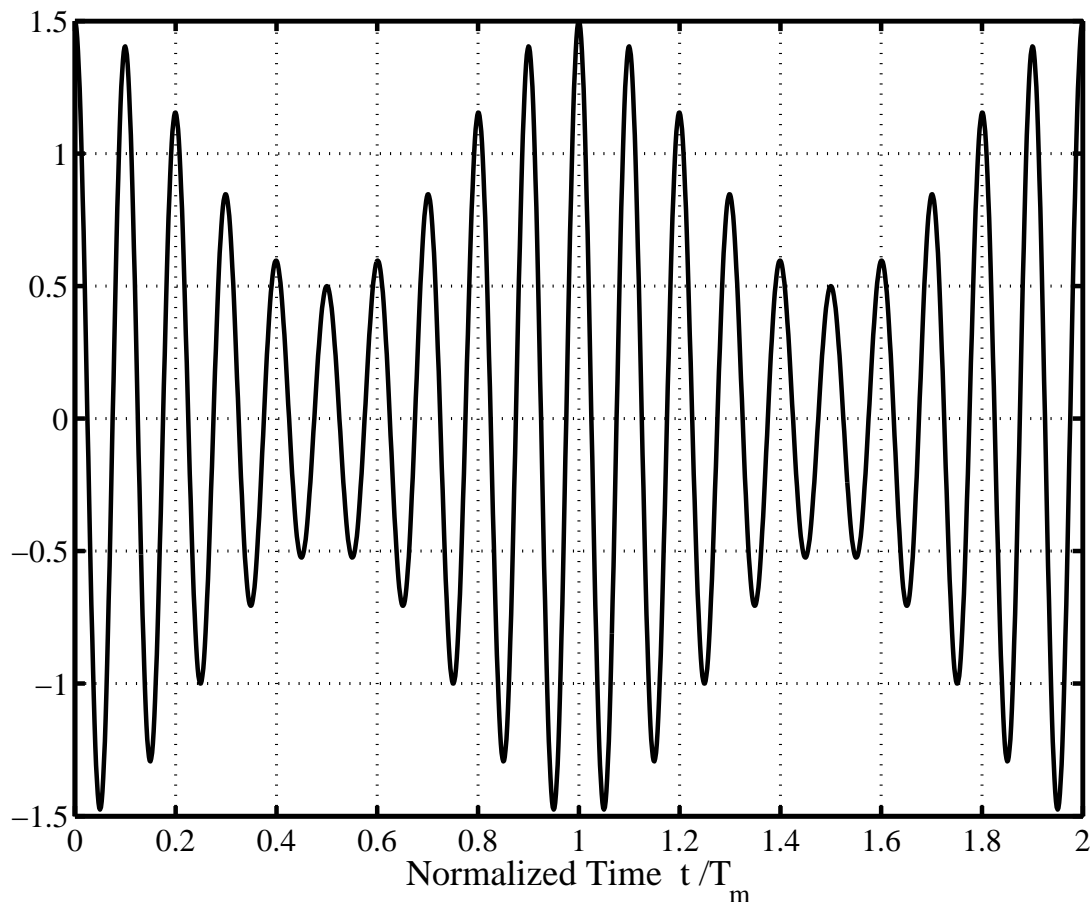
Then  $m(t)$  can be recovered from the envelope to within a scale factor and constant offset. An envelope detector is called a *noncoherent* demodulator because it makes no use of the carrier phase and frequency.

## Example for Single Tone Modulation

Let  $m(t) = A_m \cos \omega_m t$ . Then

$$s(t) = A_c(1 + \mu \cos \omega_m t) \cos \omega_c t$$

where  $\mu = k_a A_m$  is called the *modulation index*.



Example with  $\mu = 0.5$  and  $\omega_c = 10\omega_m$

## Measuring the Modulation Index for the Single Tone Case

For  $0 \leq \mu \leq 1$ , the envelope has the maximum value

$$e_{max} = A_c(1 + \mu)$$

and minimum value

$$e_{min} = A_c(1 - \mu)$$

Taking the ratio of these two equations and solving for  $\mu$  gives the following formula for easily computing the modulation index from a display of the modulated signal.

$$\mu = \frac{1 - \frac{e_{min}}{e_{max}}}{1 + \frac{e_{min}}{e_{max}}}$$

## Single Tone Case (cont.)

### Overmodulation

When  $\mu = 1$  the AM signal is said to be 100% modulated and the envelope periodically reaches 0. The signal is said to be *overmodulated* when  $\mu > 1$ .

### Transmitted vs. Message Power in $s(t)$

The transmitted signal can be expressed as

$$s(t) = A_c \cos \omega_c t + 0.5A_c\mu \cos(\omega_c + \omega_m)t + 0.5A_c\mu \cos(\omega_c - \omega_m)t$$

- The first term is a sinusoid at the carrier frequency and carries no message information.
- The other two terms are called *sidebands* and carry the information in  $m(t)$ .

## Power Relations for Single Tone Case (cont.)

The total power in  $s(t)$  is

$$P_s = 0.5A_c^2 + 0.25A_c^2\mu^2$$

while the power in the sidebands due to the message is

$$P_m = 0.25A_c^2\mu^2$$

and their ratio is

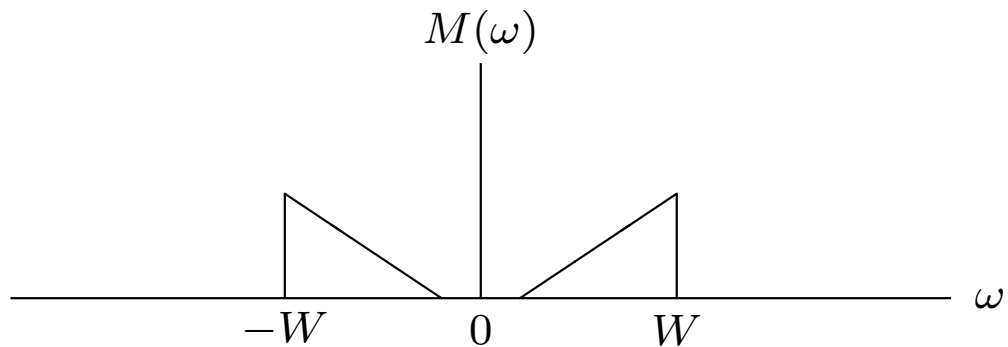
$$\eta = \frac{P_m}{P_s} = \frac{\mu^2}{2 + \mu^2}$$

This ratio increases monotonically from 0 to 1/3 as  $\mu$  increases from 0 to 1. Since the carrier component carries no message information, the modulation is most efficient for 100% modulation.

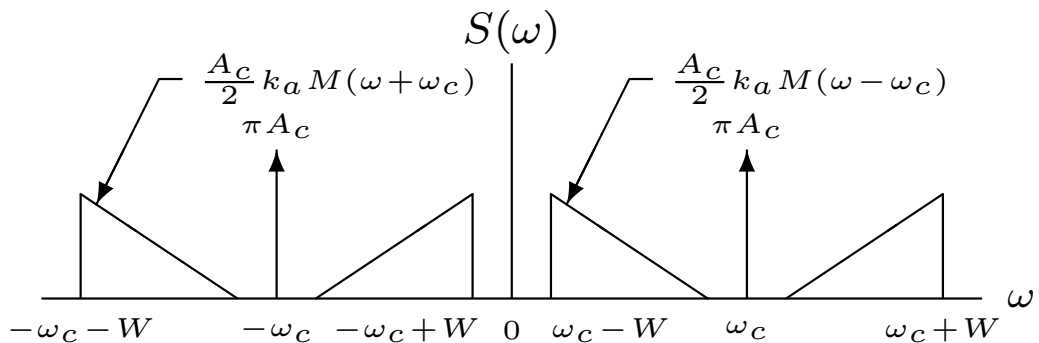
## The Spectrum of an AM Signal

Suppose the baseband message  $m(t)$  has a Fourier transform  $M(\omega)$  and  $M(\omega) = 0$  for  $|\omega| \geq W$  and  $\omega_c > W$ . Then

$$S(\omega) = A_c \pi \delta(\omega + \omega_c) + A_c \pi \delta(\omega - \omega_c) + \frac{A_c}{2} k_a M(\omega + \omega_c) + \frac{A_c}{2} k_a M(\omega - \omega_c)$$



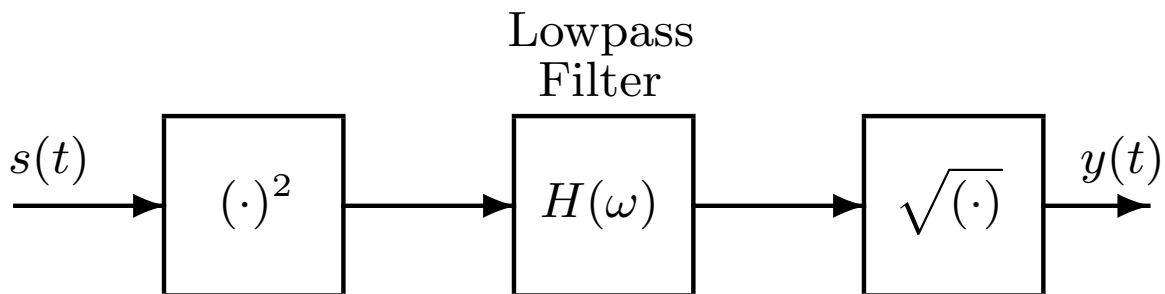
(a) Fourier Transform of Baseband Signal



(b) Fourier Transform of Transmitted AM Signal

# Demodulating an AM Signal by Envelope Detection

## Method 1: Square-Law Demodulation



Square-Law Envelope Detector

$$s(t) = A_c[1 + k_a m(t)] \cos \omega_c t$$

The baseband message  $m(t)$  is a lowpass signal with cutoff frequency  $W$ , that is,  $M(\omega) = 0$  for  $|\omega| \geq W$ .

The squarer output is

$$\begin{aligned} s^2(t) &= A_c^2[1 + k_a m(t)]^2 \cos^2 \omega_c t \\ &= 0.5A_c^2[1 + k_a m(t)]^2 \\ &\quad + 0.5A_c^2[1 + k_a m(t)]^2 \cos 2\omega_c t \end{aligned}$$

## Square-Law Envelope Detector (cont.)

- The first term on the right-hand side is a lowpass signal except that the cutoff frequency has been increased to  $2W$  by the squaring operation.
- The second term has a spectrum centered about  $\pm 2\omega_c$ . For positive frequencies, this spectrum is confined to the interval  $(2\omega_c - 2W, 2\omega_c + 2W)$ .
- The spectra for these two terms must not overlap. This requirement is met if

$$2W < 2\omega_c - 2W \quad \text{or} \quad \omega_c > 2W$$

- $H(\omega)$  is an ideal lowpass filter with cutoff frequency  $2W$  so that its output is  $0.5A_c^2[1 + k_a m(t)]^2$ .
- The square-rooter output is proportional to  $m(t)$  with a dc offset.

## Required Sampling Rate for Square-Law Demodulator

- $s^2(t)$  is band limited with upper cutoff frequency  $2(\omega_c + W)$ . Therefore,  $s(t)$  must be sampled at a rate of at least  $4(\omega_c + W)$  to prevent aliasing.
- The lowpass filter  $H(\omega)$  must operate on samples of  $s^2(t)$  taken at the rate  $4(\omega_c + W)$ .
- The output of this lowpass filter is bandlimited with a cutoff of  $2W$ . Thus, if  $H(\omega)$  is implemented by an FIR filter with tap spacing corresponding to the required fast input sampling rate, computation can be reduced by computing the output only at times resulting in an output sampling rate of at least  $4W$ . This technique is called *skip sampling* or *decimation*.

## Hilbert Transforms and the Complex Envelope

Hilbert transforms are used extensively for analysis and signal processing in passband communication systems.

Let  $x(t)$  have the Fourier transform  $X(\omega)$ . The Hilbert transform of  $x(t)$  will be denoted by  $\hat{x}(t)$  and its Fourier transform by  $\hat{X}(\omega)$ . The Hilbert transform is defined by the integral

$$\hat{x}(t) = x(t) * \frac{1}{\pi t} = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x(\tau)}{t - \tau} d\tau$$

where  $*$  represents convolution. Thus, the Hilbert transform of a signal is obtained by passing it through a filter with the impulse response

$$h(t) = \frac{1}{\pi t}$$

## Hilbert Transforms (cont.)

It can be shown that

$$H(\omega) = -j \operatorname{sign} \omega = \begin{cases} -j & \text{for } \omega > 0 \\ 0 & \text{for } \omega = 0 \\ j & \text{for } \omega < 0 \end{cases}$$

Therefore, in the frequency domain

$$\hat{X}(\omega) = H(\omega)X(\omega) = (-j \operatorname{sign} \omega) X(\omega)$$

### Some Useful Hilbert Transform Pairs

- $\cos \omega_c t \xrightarrow{\mathcal{H}} \sin \omega_c t$
- $\sin \omega_c t \xrightarrow{\mathcal{H}} -\cos \omega_c t$
- $\cos(\omega_c t + \theta) \xrightarrow{\mathcal{H}} \cos(\omega_c t + \theta - \frac{\pi}{2})$
- Let  $m(t)$  be a lowpass signal with cutoff frequency  $W_1$  and  $c(t)$  a highpass signal with lower cutoff frequency  $W_2 > W_1$ . Then

$$m(t)c(t) \xrightarrow{\mathcal{H}} m(t)\hat{c}(t)$$

## The Pre-Envelope or Analytic Signal

The *analytic signal* or *pre-envelope* associated with  $x(t)$  is

$$x_+(t) = x(t) + j \hat{x}(t)$$

### Example 1.

The pre-envelope of  $x(t) = \cos \omega_c t$  is

$$x_+(t) = \cos \omega_c t + j \sin \omega_c t = e^{j \omega_c t}$$

### Example 2.

Let  $m(t)$  be a lowpass signal with cutoff frequency  $W$  which is less than the carrier frequency  $\omega_c$ .

Then the pre-envelope of  $x(t) = m(t) \cos \omega_c t$  is

$$x_+(t) = m(t) \cos \omega_c t + j m(t) \sin \omega_c t = m(t) e^{j \omega_c t}$$

## Fourier Transform of Pre-Envelope

$$X_+(\omega) = 2X(\omega)u(\omega) = \begin{cases} 2X(\omega) & \text{for } \omega > 0 \\ X(0) & \text{for } \omega = 0 \\ 0 & \text{for } \omega < 0 \end{cases}$$

Notice that the pre-envelope has a one-sided spectrum.

## The Complex Envelope

The *complex envelope* of a signal  $x(t)$  with respect to carrier frequency  $\omega_c$  is defined to be

$$\tilde{x}(t) = x_+(t) e^{-j\omega_c t}$$

and has the Fourier transform

$$\tilde{X}(\omega) = X_+(\omega + \omega_c) = 2X(\omega + \omega_c)u(\omega + \omega_c)$$

When these definitions are used,  $x(t)$  is usually a bandpass signal and  $\omega_c$  is a frequency in the passband. Then  $\tilde{x}(t)$  is a lowpass signal.

## Pre and Complex Envelopes of AM Signal

As an example, consider the AM signal  $s(t)$ . Its pre-envelope is

$$s_+(t) = A_c[1 + k_a m(t)]e^{j\omega_c t}$$

and its complex envelope is

$$\tilde{s}(t) = A_c[1 + k_a m(t)]$$

### The Real Envelope

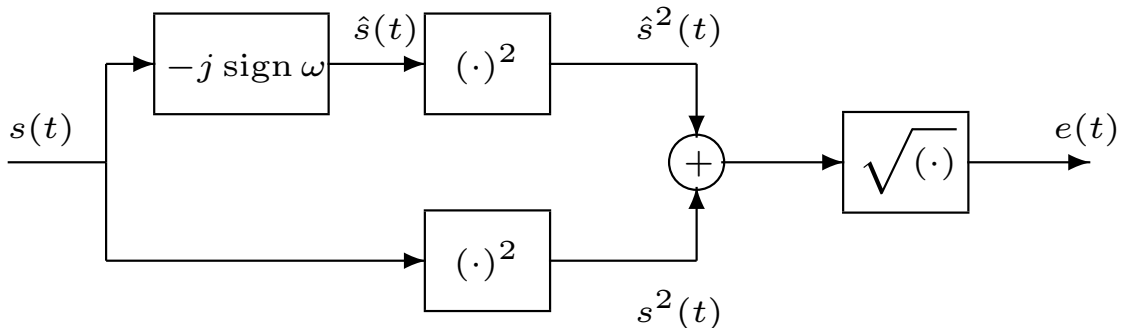
The *real envelope* of a bandpass signal  $x(t)$  is

$$e(t) = |\tilde{x}(t)|$$

Another equivalent formula for the real envelope is

$$e(t) = |x_+(t)| = [x^2(t) + \hat{x}^2(t)]^{1/2}$$

## Envelope Detector Using the Hilbert Transform



### The Required Sampling Rate

Let the message  $m(t)$  be bandlimited with cutoff  $W$ . Then  $s(t)$  is bandlimited with cutoff  $W + \omega_c$ . Thus,  $s(t)$  must be sampled at a rate of at least  $2(W + \omega_c)$  to prevent aliasing. The envelope is bandlimited with cutoff  $W$  so the output of the Hilbert transform filter and  $s(t)$  in the lower branch can be decimated to an effective sampling rate of at least  $2W$ . Also, the bound relating  $\omega_c$  and  $W$  for the square-law detector is not required for this detector.

## Chapter 5, Experiment 1

### Making an AM Modulator

Write a C program for the TMS320C6713 to:

1. Initialize McBSP0, McBSP1, and the codec as in Chapters 2 and 3, and use the left channel.
2. Read samples  $m(nT)$  from the codec at a 16 kHz rate.

Convert the samples into 32-bit integers by shifting them arithmetically right by 16 bits. The resulting integers lie in the range  $\pm 2^{15}$ .

3. AM modulate the input samples to form the sequence

$$s(nT) = A_c [1 + k_a m(nT)] \cos 2\pi f_c nT$$

where the carrier frequency is  $f_c = 3$  kHz and  $A_c$  is a constant chosen to give a reasonable size output.

## Experiment 5.1 AM Modulator (cont. 1)

- Convert the input samples to floating point numbers and do the modulation using floating point arithmetic.
  - Since the input samples lie in the range  $\pm 2^{15}$ , you must choose  $k_a$  or scale the 1 in the AM equation so that the signal is not overmodulated.
4. Send  $s(nT)$  to the DAC.
- Convert the modulated samples back into appropriately scaled integers and send them to the left channel of the DAC using polling.

### Observing Your Modulator Output

- Attach the signal generator to the DSK left channel line input and the left channel line output to the oscilloscope.

## Experiment 5.1 AM Modulator (cont. 2)

- Set the signal generator to output a 320 Hz sine-wave with an amplitude that creates less than 100% modulation. Calculate and sketch the spectrum of the AM signal.
- Synch the oscilloscope to the signal generator sine-wave and sketch the signal you observe on the oscilloscope. Better yet, use Code Composer Studio to capture the output samples and write them to a PC file as described in the following slides. Then plot the file.
- Increase the amplitude of the input signal until the AM signal is overmodulated and plot the resulting waveform. What is the effect of overmodulation on the spectrum?
- Connect the line output to the speakers of the PC and vary the modulating frequency  $f_m$ . You should be able to hear the two sidebands move up and down in frequency as you sweep  $f_m$ .

## How to Capture DSK Output Samples with CCS for Plotting

The File I/O feature of Code Composer Studio can be used to transfer blocks of words from the 'C6713 memory to a text file on the PC. To do this using the polling method:

- Include a loop inside the usual infinite `while(1)` or `for(;;)` loop that writes the integer output samples to a *global* array, for example, `output []`.
- When this inner loop exits, the array is filled and ready to be captured by CCS.
- Immediately after the inner loop, put a dummy statement that involves `output []` for a probe point, for example,  
`dummy = output [0];`.

The following code segment illustrates this approach.

## Code Segment to Allow Capture

```
#define BUFFER_SIZE 256
int output[BUFFER_SIZE]; /* Must be global so CCS    */
                          /* can see it            */

main()
{
    /* Initialize DSK, etc.    */
    while(1)
    {
        for(i=0; i < BUFFER_SIZE; i++)
        {
            /* 1. Poll XRDY flag and read new input sample.    */
            /* 2. Process the sample as desired.                */
            /* 3. Write output sample to buffer.                */
            output[i] = processed_sample;
            /* 4. Write output sample to codec.                 */
        }
        dummy = output[0]; /* Dummy line for probe point    */
    }
}
```

## Setting Up CCS for Capture

Connect the probe point to the output file by the following steps:

1. Set the probe point in your code.

## Setting Up CCS for Capture

2. Choose “File → File I/O” from the CCS main menu.
3. In the dialog box that appears, click the “File Output” tab and then “Add File.”
4. Enter the filename of your choice but be sure to choose a file of type “\*.dat(Integer).”
5. Return to the File I/O dialog box. Enter output (or whatever you call your output buffer array) and its length in the “address” and “length” boxes respectively.
6. Click “Add Probe Point” and the Break/Probe/Profile Points dialog box will appear.
7. Select “[*your-source=file.c*] line zzz → No Connection” in the Probe Point list.

## Setting Up CCS for Capture (cont.)

8. Select “File Out [*your-output-file.out*]” in the “Connect To:” drop-down box. Then click “Replace.” The Probe Point list should be updated appropriately.
9. Click “OK” to close the Probe Points dialog box.
10. Click “OK” to close the File I/O box. The “tape player” window should appear.
11. Push the “play button” on the tape player and run your program. Hit the “stop” button when you’ve collected enough samples.

**NOTE:** The first line in the output file contains irrelevant data. After the first line is removed, the file should be your captured samples, one per line, and can be plotted with your favorite plotting routine.

## An Easier Method of Sending Data from the DSK to the PC

You can use the standard file I/O functions, `fopen()`, `fprintf()`, and `fclose()` to send data from the DSK to a file on the PC. A sample code segment is shown below. File I/O does not run in real time.

```
#include 'stdio.h'
#include 'stdlib.h'
#define BUFFER_SIZE 512
float foutput[BUFFER_SIZE];
FILE *fp;
    fp = fopen('PC_data_file_name', 'w');
    /* Initialize DSK, etc. */
    for(i=0; i < BUFFER_SIZE; i++)
    {
        /* 1. Poll XRDY flag and read new input sample. */
        /* 2, Process the sample as desired. */
        /* 3. Put floating point output sample in buffer.*/
        foutput[i] = processed_sample;
        /* 4. Write sample to DAC. */
    }
    /* Write buffer, foutput[], to PC file. */
    for(i=0; i < BUFFER_SIZE; i++) fprintf(fp, '%15.5e\n',
                                           foutput[i]);
fclose(fp);
```

## Using Windows XP Sound Recorder to Capture the DSK LINE OUT

The DSK LINE OUT signal can be recorded as a .wav file using the Sound Recorder of Windows XP. To do this,

1. Connect the DSK LINE OUT to the input on the PC's sound card.
2. Then click on start, All Programs, Accessories, Entertainment, and Sound Recorder.
3. In the Sound Recorder window, click on Edit, Audio Properties, and then Volume... under Sound recording.
4. Select Line In and set the desired Balance and Volume. "OK" back to the Sound Recorder window.
5. To set the sampling rate and sample format, click on File, Properties, and then Convert Now in the Format Conversion box.

## Using Windows XP Sound Recorder to Capture the DSK LINE OUT (cont.)

You can select from a wide variety of sampling rates ranging from 8 to 48 kHz; mono or stereo; and a variety of formats including PCM, MPEG Layer-3, A-Law, and  $\mu$ -Law. When you are done, “OK” back to the Sound Recorder window.

6. Click on File and then New. Click on the red dot in the lower right-hand corner to begin recording. Click on the black rectangle to stop.
7. Click on File and then Save. The saved .wav file can be plotted or listened to with MATLAB or Windows applications including Sound Recorder.

## Chapter 5, Experiment 5.2

### Making a Square-Law Envelope Detector

- Write a program for the TMS320C6713 to implement the square-law envelope detector shown on Slide 5-8. Continue to use a 16 kHz sampling frequency. Use the signal generator as the source of the AM signal  $s(t)$ . Take the input samples from the ADC, perform the demodulation, and send the demodulated output samples to the DAC.
- Set the carrier frequency to 3 kHz and assume the baseband message  $m(t)$  is bandlimited with a cutoff frequency of 400 Hz.
- Use a Butterworth lowpass IIR filter for  $H(\omega)$  that has an order sufficient to suppress the unwanted components around  $2f_c$  by at least 40 dB.

## Experiment 5.2 Making a Square-Law Detector (cont.)

- Assume that  $m(t)$  has no spectral components below 50 Hz and remove the dc offset at the output of the square root box by a simple highpass filter of the form

$$G(z) = \frac{1+c}{2} \frac{1-z^{-1}}{1-cz^{-1}}$$

where  $c$  is a constant slightly less than 1 chosen so that the lowest frequency components of  $m(t)$  are negligibly distorted. Notice that this filter has an exact null at 0 frequency and is 1 at half the sampling rate. Plot the amplitude response of this filter for various  $c$  to select an appropriate value. You could also use the program IIR.EXE to design this filter.

## **Experiment 5.2.1**

### **Observing the Square-Law Detector Output with No Input Noise**

- Attach the signal generator to the ADC input and the oscilloscope to the DAC output.
- Set the signal generator to create an AM wave with a sinusoidal modulating signal with a frequency between 100 and 400 Hz. Use a 3 kHz carrier frequency.
- Sketch or capture and plot the AM input signal and the output of your demodulator.

## **Experiment 5.2.2**

### **Observing the Square-Law Detector Output with Input Noise**

Experiment demodulating signals corrupted by additive, zero mean, Gaussian noise.

## Experiment 5.2.2 Demodulating with Input Noise (cont.)

- The lab does not have hardware continuous-time noise generators. You will have to simulate the noise in the DSP using the method described in Slides 5-31 through 5-35 and add the simulated noise sample to the input signal sample.
- Use the same sinusoidally modulated AM signal as before.
- Start with a large signal-to-noise power ratio (SNR) and decrease it until the demodulator output is very noisy and barely resembles the message sinusoid. The degradation will increase relatively smoothly over a range of SNR and you will have to use your judgement as to what “barely resembles” means. Listening to the noisy output may help. Estimate the SNR in dB at this point.

# Generating Gaussian Random Numbers

Pairs of independent, zero mean, Gaussian random numbers can be generated by the following steps:

1. **Generate Random Numbers Uniform Over  $[0,1)$**

- **rand**(void) generates integers uniformly distributed over  $[0, \text{RAND\_MAX}]$  where  $\text{RAND\_MAX} = 32767$ .
- **srand**(unsigned int seed) sets the value of the random number generator seed so that subsequent calls of **rand** produce a new sequence of pseudorandom numbers. **srand** does not return a value.
- If **rand** is called before **srand** is called, a seed value of 1 is used.

## Gaussian RV's (cont. 1)

### Sample Code for Generating Uniform [0,1) RV's

```
float v;  
v = (float) rand() / (RAND_MAX + 1);
```

### 2. Converting a Uniform to a Rayleigh Random Variable

A Rayleigh random variable,  $R$ , has the pdf

$$f_R(r) = \frac{r}{\sigma^2} e^{-\frac{r^2}{2\sigma^2}} u(r)$$

and cumulative distribution function

$$F_R(r) = \left[ 1 - e^{-\frac{r^2}{2\sigma^2}} \right] u(r)$$

## Gaussian RV's (cont. 2)

Let  $v = \left[1 - e^{-\frac{r^2}{2\sigma^2}}\right] u(r)$ . Then, for  $0 \leq v < 1$ , the inverse cdf is

$$r = F_R^{-1}(v) = \sqrt{-2\sigma^2 \log_e(1 - v)}$$

Now let  $V$  be a random variable uniform over  $[0,1)$ . Then

$$R = \sqrt{-2\sigma^2 \log_e(1 - V)}$$

is a Rayleigh random variable.

**Warning:** The Code Composer C compiler has two natural logarithm functions. One is for “float” and the other for “double” inputs and outputs. Strange program behavior and failure can occur if you use the wrong ones!

The “float” function prototype is:

float **logf**(float x).

The “double” function prototype is:

double **log**(double x)

### Gaussian RV's (cont. 3)

#### 3. Transforming a Rayleigh RV into an Independent Pair of Gaussian RV's

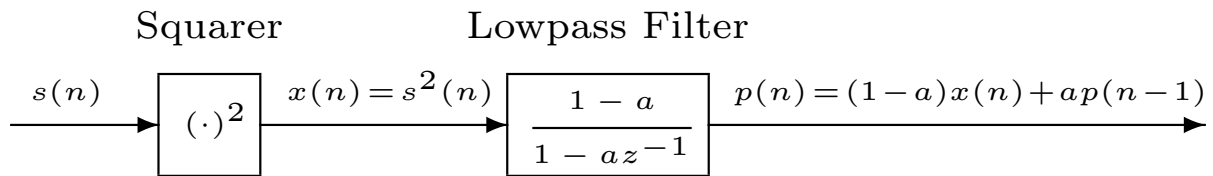
Let  $\Theta$  be a random variable uniformly distributed over  $[0, 2\pi)$  and independent of  $V$ . Then

$$X = R \cos \Theta \quad \text{and} \quad Y = R \sin \Theta$$

are two independent Gaussian random variables, each with zero mean and variance  $\sigma^2$ . That is, they each have the pdf

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

## Measuring the Signal Power



### A Simple Power Meter

The average noise power is  $\sigma^2$ . Estimate the average power of the AM signal by making the power meter shown above. Choose  $a$  close to but slightly less than 1, for example, 0.99. Make a loop to run the power meter for several thousand samples. Put a break point after the loop.

Examine the final value of  $p(n)$  with Code Composer when the program halts at the break point.

## Experiment 5.3

### Making an Envelope Detector Using the Hilbert Transform

- Write a program for the DSK to implement the envelope detector shown on Slide 5-16. Again, assume a carrier frequency of 3 kHz and a baseband message bandlimited to 400 Hz. In addition to the components shown in the figure, add the same highpass filter you used in the square-law detector after the square root box to remove the dc component of the detector output.
- You can design the Hilbert transform filter with the program REMEZ87.EXE.
  - Use an odd number,  $N$ , of filter taps.
  - Good results can be achieved with REMEZ by using just one band with a lower cutoff frequency  $f_1$  and upper cutoff frequency  $f_2$  chosen to pass the AM signal.

## Experiment 5.3

### Hilbert Transform Demodulator (cont. 1)

Choosing the band to be centered in the Nyquist band also seems to improve the filter amplitude response generated by REMEZ87. To center the band, choose the upper cutoff frequency to be

$$f_2 = 0.5f_s - f_1$$

where  $f_s$  is the sampling frequency.

- Enter 1 for the magnitude of the Hilbert transform in the band and 1 for the weight factor.
- Select  $N$  so that the amplitude response of the filter is quite flat over the signal passband and ripples caused by incomplete cancellation of the  $2f_c$  components are essentially invisible in the demodulated output.

## Experiment 5.3

### Hilbert Transform Demodulator (cont. 2)

- You can also design the Hilbert transform filter with the program WINDOW.EXE. Try using the Hamming and Kaiser windows. You can make a tradeoff between the transition bandwidth and the out-of-band attenuation with the Kaiser window.
- The resulting FIR filter will have a delay equal to the delay from the input to the center tap of the filter,  $T(N - 1)/2$ . Therefore,  $s(nT)$  must be delayed by this amount to match the delay in  $\hat{s}(nT)$ . This can be easily accomplished by taking  $s(nT)$  from the point in the delay-line of the Hilbert transform filter at its center tap.
- Test your envelope detector using the same steps as you did for the square-law detector.