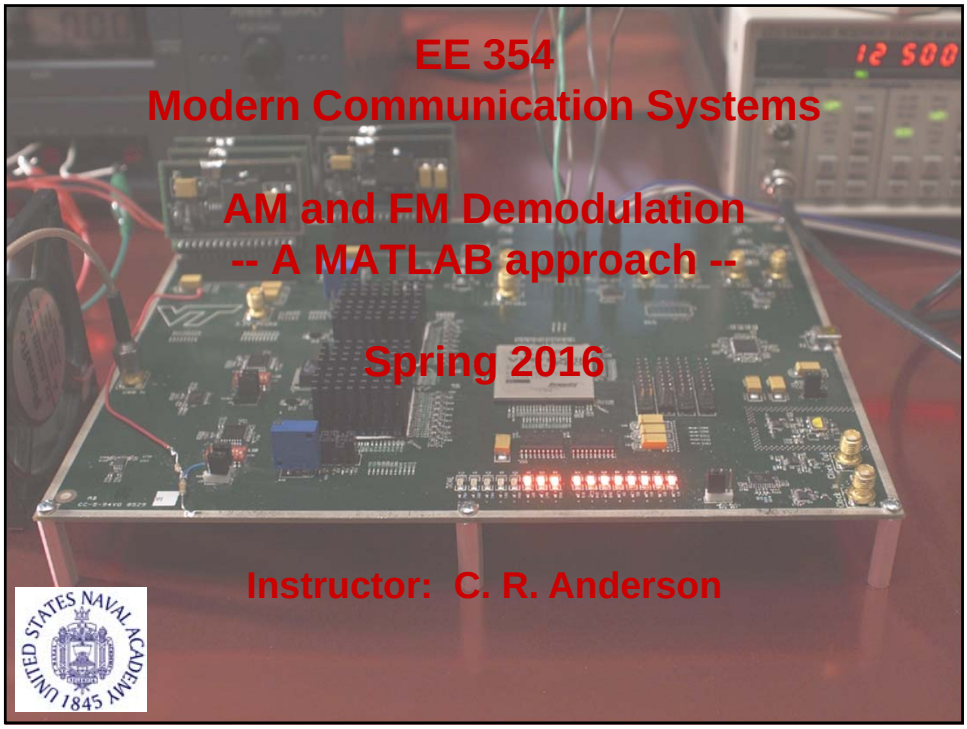


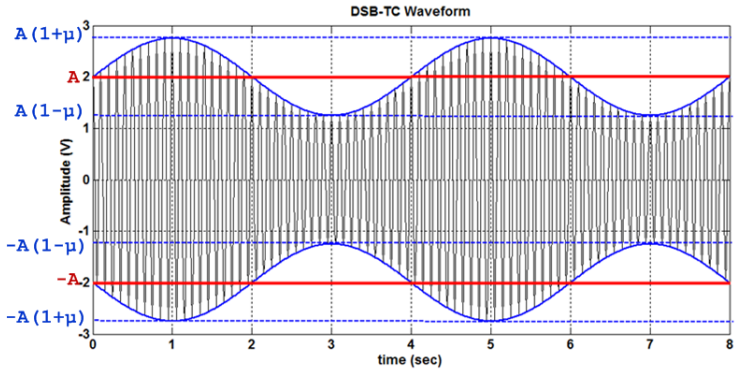
EE 354
Modern Communication Systems

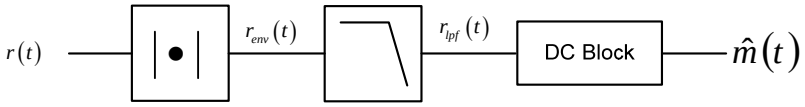
AM and FM Demodulation
-- A MATLAB approach --

Spring 2016

Instructor: C. R. Anderson

Recovering the Message: Envelope Detector





Observe: **If we can trace out the Envelope of the AM signal, we can effectively recover the underlying information signal.**

2

Envelope Detector Mathematically

Received Signal: $r(t) = A_c\left[1 + a_{\text{mod}} m_n(t)\right]\cos(2\pi f_c t)$

Absolute Value Operation: $r_{env}(t) = A_c\left|\left[1 + a_{\text{mod}} m_n(t)\right]\cos(2\pi f_c t)\right|$

If $1 + a_{\text{mod}} m_n(t)$ is constrained to be always positive (i.e., $0 \le a_{\text{mod}} \le 1$)

$$r_{env}(t) = A_c\left[1 + a_{\text{mod}} m_n(t)\right]\left|\cos(2\pi f_c t)\right|$$

Note: The full-wave rectified cosine can be expanded in a Fourier Series

$$r_{env}(t) = A_c\left[1 + a_{\text{mod}} m_n(t)\right]\left[a_0 + a_1\cos(2\pi(2f_c)t) + a_2\cos(2\pi(4f_c)t) + \ldots\right]$$

LPF: Block everything except the a_0 term.

$$r_{env}(t) = a_0 A_c\left(1 + a_{\text{mod}} m_n(t)\right)$$

DC Block does the rest: $\hat{m}(t) = a_0 a_{\text{mod}} m_n(t)$

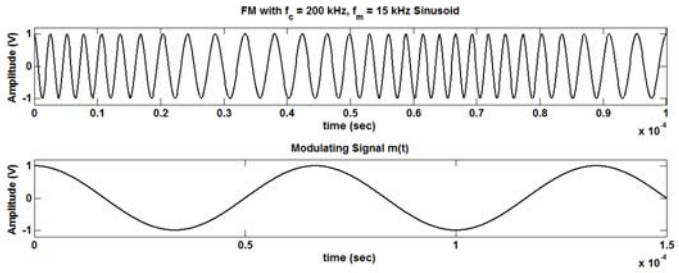
3

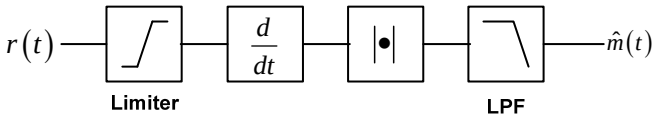
AM Demodulation in Software

```
fs_rf = 1e6;   % sampling frequency of the AM signal
fco = 0.02e6;  % cutoff frequency for the low pass filter
% Will need to downsample the recovered signal to an audio-
% level sampling frequency for output to sound card.
downsample_rate = floor(fs_rf./44.1e3);

% Perform AM Demod using Envelope Detection.
r_env = abs(am_samples); % Envelope Detector
r_lpf = filter_audio(x_env,fco,fs_rf);  % Low Pass Filter
% Downsample to audio sampling frequency
fs_audio = fs_rf./downsample_rate;
m_hat = downsample(r_lpf, downsample_rate);

% Perform AM Demod using Square-Law Detector
r_sq = am_samples.^2; % Square-Law Detector
% Low Pass Filter and Square Root operation
m_hat = sqrt(filter_audio(r_sq,fco,fs_rf));
% Downsample to audio sampling frequency
fs_audio = fs_rf./downsample_rate;
x_bb = downsample(m_hat, downsample_rate);

% Output the demodulated signal to the sound card
sound(m_hat, fs_audio, 16);
```

4

FM Demodulation: Implement the Instantaneous Frequency Equation





$$r(t) = A_c \cos\left(2\pi f_c t + 2\pi k_f \int_0^t m(\tau)\, d\tau\right) \qquad f_i = \frac{1}{2\pi}\frac{d\theta(t)}{dt} \qquad \hat{m}(t) = 2\pi A_c k_f m(t)$$

Note that there's another way to do this…

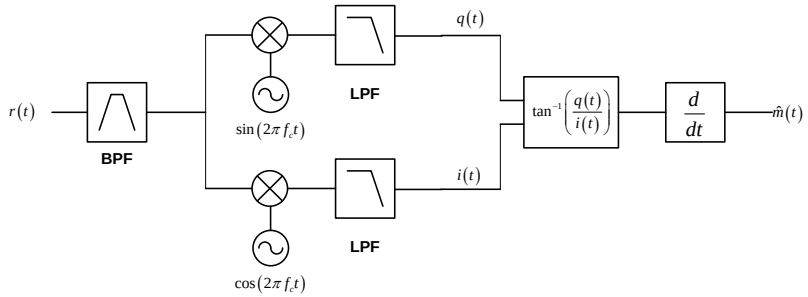
5

FM Demodulation in Software

```
fs_rf = 1e6;   % sampling frequency of the FM signal
fco = 0.02e6;  % cutoff frequency for the low pass filter
% Will need to downsample the recovered signal to an audio-
% level sampling frequency for output to sound card.
downsample_rate = floor(fs_rf./44.1e3);

% Perform FM Demod using Instantaneous Frequency Method.
% Note: A hard limiter would go here to reduce noise

diff_sfm = diff(sfm); % Take Derivative of FM signal

% Matlab 'diff' reduces length by 1 sample, need to add a
% sample back in its place.
diff_sfm = [0 diff_sfm];
sfm_abs = abs(diff_sfm); % Envelope Detector
m_demod = filter_audio(sfm_abs,fco,fs_rf);  % Low Pass Filter

% Downsample to audio sampling frequency
fs_audio = fs_rf./downsample_rate;
x_bb = downsample(x_demod, downsample_rate);

% Output the demodulated signal to the sound card
sound(x_bb, fs_audio, 16);
```

6

FM Arctangent Demodulator



$$i(t) = \tfrac{1}{2}A\cos(\theta(t))$$
$$q(t) = \tfrac{1}{2}A\sin(\theta(t))$$

$$\hat{m}(t) = \tan^{-1}\left(\frac{q(t)}{i(t)}\right) = \tan^{-1}\left(\frac{\frac{1}{2}A\sin(\theta(t))}{\frac{1}{2}A\cos(\theta(t))}\right) = \tan^{-1}\left(\tan(\theta(t))\right) = \theta(t)$$

$Note:\ \theta(t) = 2\pi k_f \int_0^t m(\tau)\,d\tau$

Need a 4-Quadrant ArcTan or Phase Unwrap.

"Nobody does this---I'm shocked!" --- fred harris

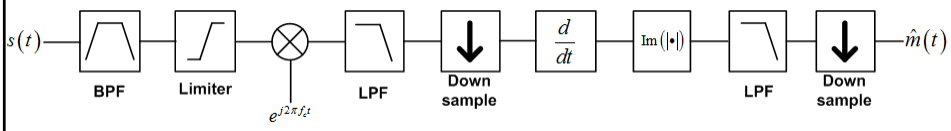
7

FM ArcTan Demodulation in Software

```
fs_rf = 1e6;   % sampling frequency of the FM signal
fco = 0.02e6;  % cutoff frequency for the low pass filter
% Will need to downsample the recovered signal to an audio-
% level sampling frequency for output to sound card.
downsample_rate = floor(fs_rf./44.1e3);

% Perform FM Demod using ArcTangent Method.
% Note: A hard limiter would go here to reduce noise
% Create the Inphase and Quadrature Signals
q_sig = sfm.*sin(2*pi*fc*t);
i_sig = sfm.*cos(2*pi*fc*t);

% Perform 4-Quadrant Arctangent
arctan_sfm = unwrap(atan2(q_sig, i_sig));

% Matlab 'arctan2' and 'diff' reduces length by 1 sample, need to add a
% sample back in its place.
arctan_sfm = [0 arctan_sfm];
diff_sfm = diff(sfm); % Take Derivative of Arctangented signal
diff_sfm = [0 diff_sfm];
m_demod = filter_audio(sfm_abs,fco,fs_rf);  % Low Pass Filter
```

8

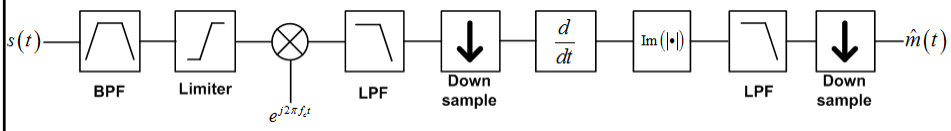
FM Complex Exponential Demodulator




```
fs_rf = 1e6;   % sampling frequency of the FM signal
fco = 0.02e6;  % cutoff frequency for the low pass filter
% Will need to downsample the recovered signal to an audio-
% level sampling frequency for output to sound card.

fs_soundcard = 44.1e3;
fc = XXXXXX;   % Center frequency of the FM Signal
% Decimate to sampling rate ~ just greater than the FM BW
dec_rate_rf = floor(fs_rf./(fs_soundcard.*XXXX));
fs_bb = fs_rf./dec_rate_rf;
dec_rate_bb = ceil(fs_bb./fs_soundcard);
fs_audio = fs_rf./(dec_rate_rf.*dec_rate_bb);

% Bandpass Filter
x_fm = filter_IF(sfm,flow,fhigh,fs_rf);
```

9

FM Complex Exponential Demodulator

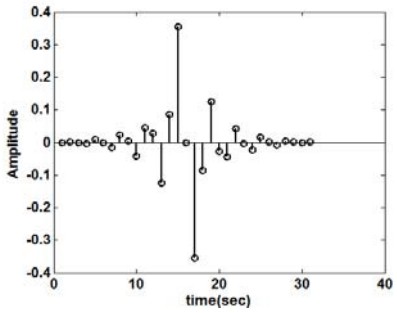


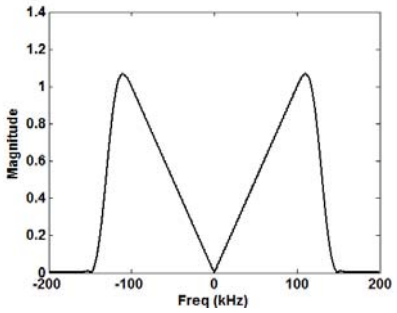

```
% Downconvert to baseband and filter
x_bb = filter_bb((x_fm.*exp(-j.*2.*pi.*fc.*time)),frf_co,fs_rf);
x_bb_ds = downsample(x_bb,dec_rate_rf); % Downsample, first stage
x_limit = x_bb_ds./abs(x_bb_ds); % Limiter (could move earlier)

% Generates an "Ideal" differentiator using an FIR Filter
hd = firls(30,[0 100000 0.8.*fs_bb fs_bb]/fs_bb, [0 1 0 0],'differentiator');
x_diff = imag(conv(x_limit,hd,'same').*conj(x_limit));

% Now, just extract the audio signal
x_demod = filter_bb(x_diff,fco,fs_bb);
t_bb = downsample(time, dec_rate_rf*dec_rate_bb);
x_bb = downsample(x_demod, dec_rate_bb);

soundsc(x_bb,fs_audio,16)
```

10

Note: FIR Filter as a Differentiator

```
hd = firls(30,[0 100000 0.8.*fs_bb fs_bb]/fs_bb, [0 1 0 0],'differentiator');
% Note:  This is a digital version of a CT Differentiator
% F[d/dt] = j*2*pi*fc*t  → Linear frequency response upto the cutoff freq.

% Plot the filter, and its frequency response

subplot(121);
stem(hd);
Subplot(122);
f = [-fs_bb./10:fs_bb./10-1];
plot(f,abs(fftshift(fft([hd zeros(1,2.*(fs_bb./10)-31)]))));
```





11