Lab 6: PAM Receiver with Matched Filter and Symbol Timing Extraction

1 Introduction

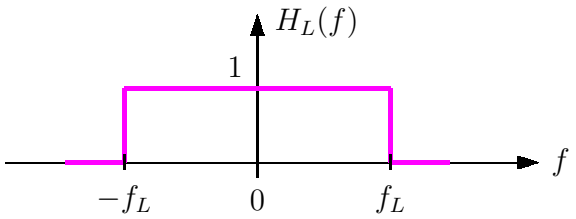
Communication without noise would be trivial. You could take the text of a whole encyclopedia, encode it in ASCII, and make a long binary string by concatenating the resulting bits. To transmit this string as a single symbol you would interpret it as the binary representation of a number which determines the amplitude of a voltage or current pulse sent by the transmitter. The receiver would then convert the received amplitude back into a number whose binary representation is the received bit string. If the bit string is 7 bits long (from one ASCII character), then the receiver needs to distinguish 128 different amplitude levels, and the scheme actually works quite well. But if a whole text results in 1 million bits, say, then the receiver needs to be able to distinguish $2^{1,000,000} \approx 10^{300,000}$ levels, which is impossible in the presence of the slightest amount of noise. Thus, there is no other choice than to transmit a long text as a sequence of smaller numbers and, to minimize the probability of error, to build the receiver front-end such that it maximizes the signal-to-noise ratio (SNR) for each reception. An associated problem that comes from transmitting sequences of numbers rather than a single number is that the receiver must synchronize itself to the transmitter. In practice this means in most cases that symbol timing information must be extracted from the received signal waveform itself.

1.1 Trapezoidal Lowpass Filter

Lowpass filters (LPF) are used for many different tasks in communication systems. Among them are the separation of a desired signal from noise and interference and symbol timing extraction at the receiver.

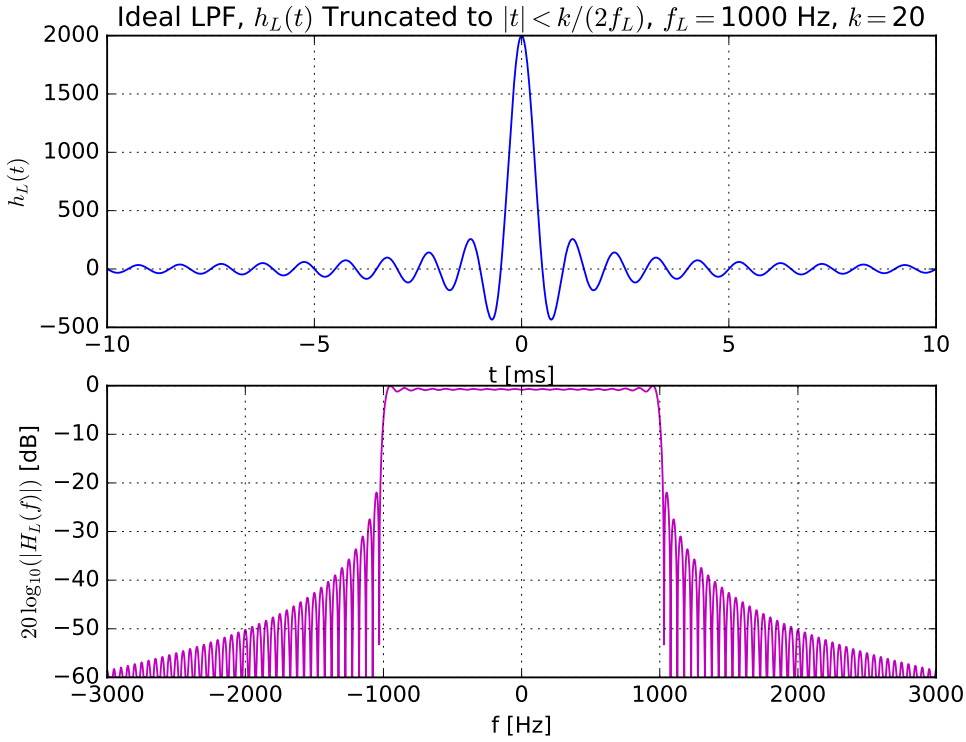
The impulse response $h_L(t)$ and the frequency response $H_L(f)$ of an ideal LPF with cutoff frequency f_L are shown in the figure below.

$$h_L(t) = \frac{\sin(2\pi f_L t)}{\pi t} \qquad \Longleftrightarrow$$

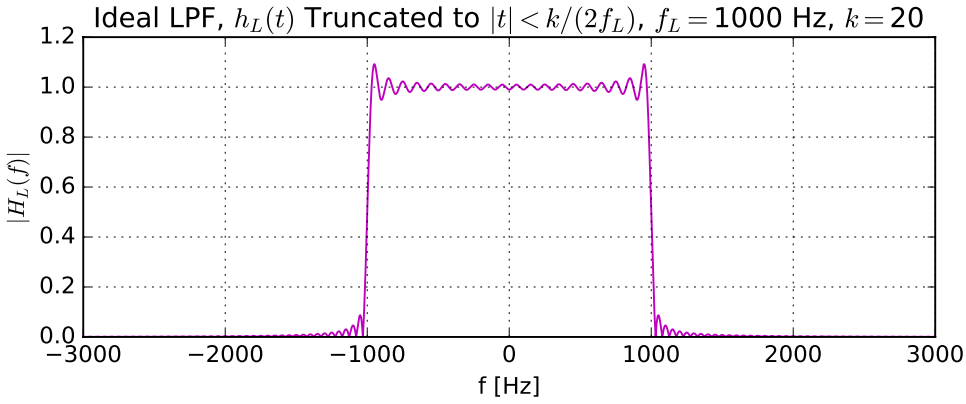


For practical applications the problem with the ideal LPF is that $h_L(t)$ needs to be truncated, e.g., to the time interval $-k/(2f_L) \leq t \leq k/(2f_L)$ for some integer k, which leads to

substantial sidelobes in the frequency response whenever k is finite. The following plots of $h_L(t)$ and $20\log_{10}(|H_L(f)|)$ (in dB) show an example for $f_L = 1000$ Hz and $k = 20$.

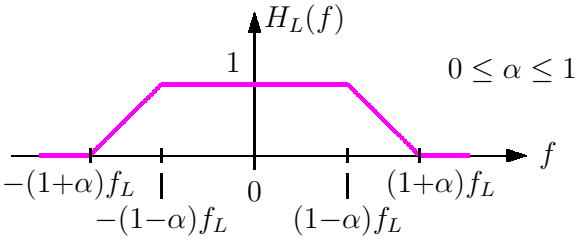


If the magnitude of the frequency response $|H_L(f)|$ is displayed on a linear scale, we see a ripple in both the passband and the stopband of the filter as a result of the truncation of $h_L(t)$.

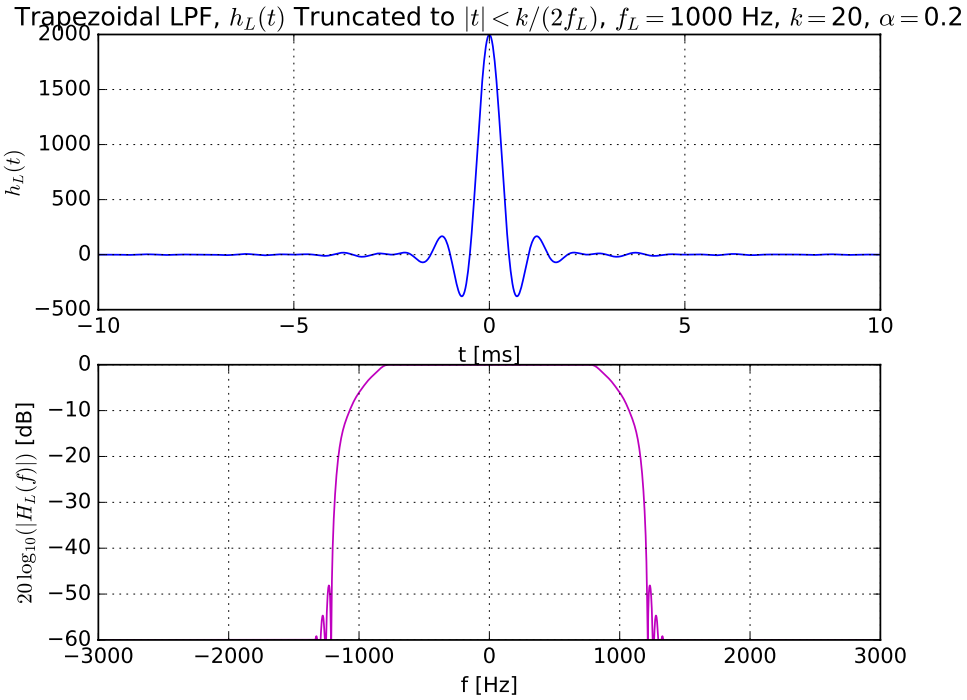


By making the transition from the passband to the stopband less abrupt, the situation can be improved considerably. One possibility is to use an LPF with a trapezoidal frequency response as shown next.

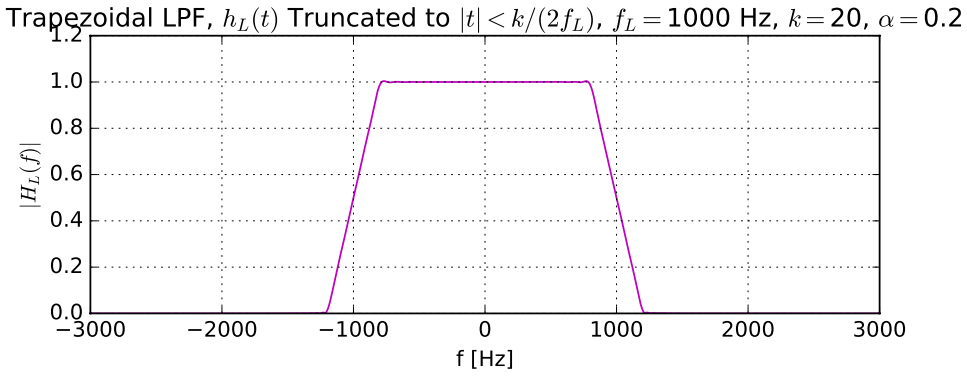
$$h_L(t) = \frac{\sin(2\pi f_L t)}{\pi t}\,\frac{\sin(2\pi\alpha f_L t)}{2\pi\alpha f_L t} \quad \Longleftrightarrow$$



As the parameter α is varied from 0 to 1, the frequency response goes from an ideal LPF to a $H(f)$ with triangular shape. An example with $f_L = 1000$ Hz, $k = 20$, and $\alpha = 0.2$ is shown below.

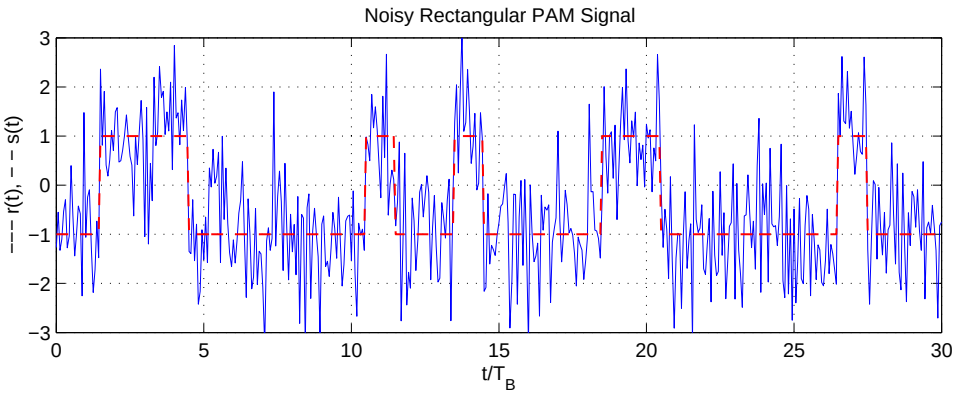


An additional advantage of a LPF with trapezoidal frequency response is that its transition region around f_L can be used as a frequency discriminator, i.e., as a device that converts linearly from frequency to amplitude. A linear plot of the magnitude of the frequency response of the same filter that was used as an example above is plotted in the next figure.

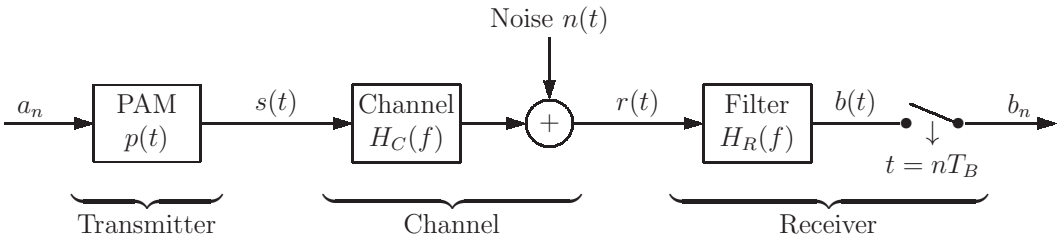


1.2 Filtering PAM Signals

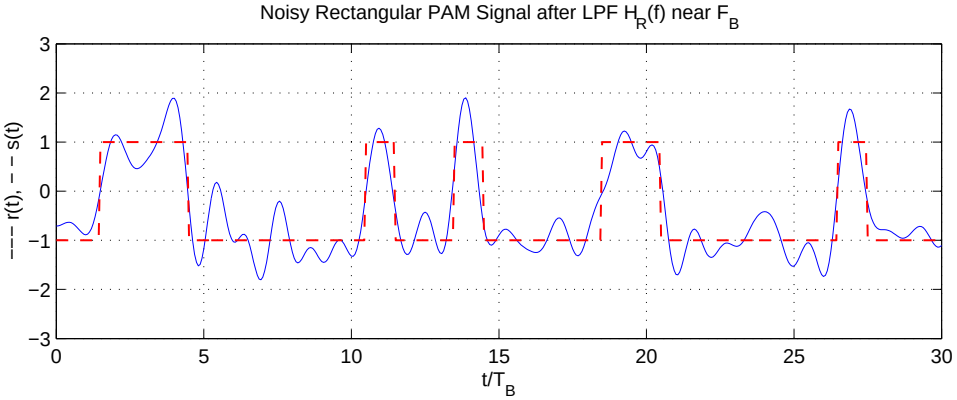
Suppose you received a noisy rectangular PAM signal $r(t)$ with baud rate $F_B = 1/T_B$, like the one shown in the graph below.



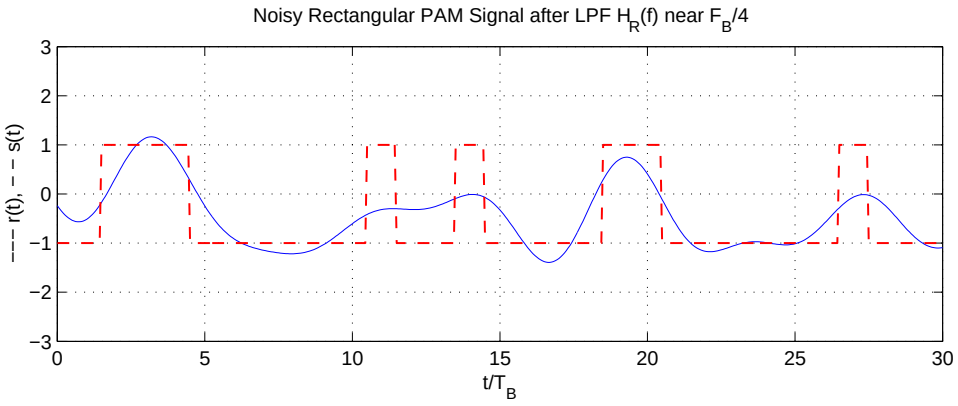
Before sampling $r(t)$ at times $t = nT_B$, it certainly is a good idea to perform some kind of averaging operation. This is the function of the receiver filter $h_R(t) \Leftrightarrow H_R(f)$ shown in the following blockdiagram of a general PAM communication system.



If $H_R(f)$ is a LPF with cutoff frequency near F_B, then the output $b(t)$ of the receiver filter from the same $r(t)$ as shown above looks as follows.



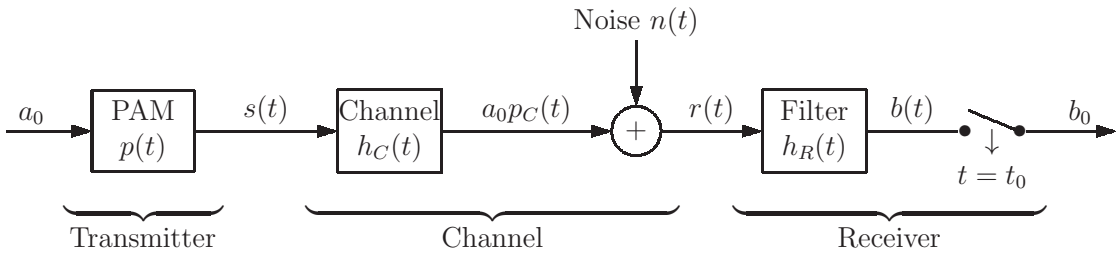
This is quite an improvement, and it is tempting to see if more filtering would lead to further improvements. The following graph shows $r(t)$ after lowpass filtering near $F_B/4$.



That does remove more noise, but at the price of loosing signal energy and introducing intersymbol interference (ISI). Thus, a compromise needs to be made between rejecting as much noise as possible, while keeping most of the signal energy and avoiding ISI. A properly chosen LPF is a good initial choice, but to maximize the signal-to-noise ratio (SNR) at the output of the receiver filter, a matched filter, as described in the next section, needs to be used.

1.3 Matched Filter

Suppose a transmitter sends the PAM signal $s(t) = a_0\, p(t)$ over a channel with unit impulse response $h_C(t) \Leftrightarrow H_C(f)$. It is assumed that $p(t)$ is a deterministic pulse, but a_0 is a random variable. In the absence of noise the received signal is thus $a_0\, p(t) * h_C(t) = a_0\, p_C(t)$, where $p_C(t) = p(t) * h_C(t)$. If an additive noise model is used for the channel, then the received signal is $r(t) = a_0\, p_C(t) + n(t)$, where $n(t)$ is the noise signal. The question now is how to design a receiver that estimates a_0 as accurately as possible. The following block diagram depicts the situation graphically.



Note that, in order to avoid the issue of ISI, this model uses the so called "one shot" approach, i.e., only the single sample a_0 is transmitted. If the overall pulse $q(t) = p_C(t) * h_R(t)$ satisfies Nyquist's first criterion, then the results from the "one shot" approach directly generalize to the transmission of sequences $a_0, a_1, a_2, \ldots$ of symbols.

Assume that the random variable a_0 and the noise $n(t)$ are uncorrelated and that $n(t)$ is a wide-sense stationary (WSS) CT random process with zero mean, i.e., $E[n(t)] = 0$ and

autocorrelation function $R_n(t, t-\tau) = E[n(t)\, n^*(t-\tau)] = R_n(\tau)$, all t. The power spectral density (PSD) $S_n(f)$ of the noise is the FT of the correlation function $R_n(\tau)$, i.e.,

$$S_n(f) = \int_{-\infty}^{\infty} R_n(\tau) e^{-j2\pi f\tau}\, d\tau\ .$$

For **white noise** $R_n(\tau) = (\mathcal{N}_0/2)\delta(\tau) \Leftrightarrow S_n(f) = \mathcal{N}_0/2$, all f, i.e., white noise has a flat (2-sided) PSD.

One optimality criterion for selecting the receiver filter response $h_R(t)$ is the maximization of the signal-to-noise ratio (SNR) of b_0 after sampling at $t = t_0$. If $r(t) = a_0\, p_C(t) + n(t)$, then

$$b(t) = h_R(t) * \big(a_0\, p_C(t) + n(t)\big) = a_0 \int_{-\infty}^{\infty} H_R(f)\, P_C(f)\, e^{j2\pi ft}\, df + \int_{-\infty}^{\infty} h_R(\mu)\, n(t-\mu)\, d\mu\ .$$

Therefore

$$\begin{aligned} E[b(t)] &= E[a_0] \int_{-\infty}^{\infty} H_R(f)\, P_C(f)\, e^{j2\pi ft}\, df + \int_{-\infty}^{\infty} h_R(\mu)\, \underbrace{E[n(t-\mu)]}_{=0}\, d\mu \\ &= E[a_0] \int_{-\infty}^{\infty} H_R(f)\, P_C(f)\, e^{j2\pi ft}\, df\ . \end{aligned}$$

In the absence of noise (i.e., $n(t) = 0$)

$$\begin{aligned} E[|b(t)|^2] &= E\Big[\big(a_0 \int_{-\infty}^{\infty} H_R(f)\, P_C(f)\, e^{j2\pi ft}\, df\big)\big(a_0 \int_{-\infty}^{\infty} H_R(\nu)\, P_C(\nu)\, e^{j2\pi\nu t}\, d\nu\big)^*\Big] \\ &= E[|a_0|^2] \Big| \int_{-\infty}^{\infty} H_R(f)\, P_C(f)\, e^{j2\pi ft}\, df \Big|^2\ , \end{aligned}$$

and thus the signal power at $t = t_0$ is

$$E[|b_0|^2] = E[|b(t_0)|^2] = E[|a_0|^2] \Big| \int_{-\infty}^{\infty} H_R(f)\, P_C(f)\, e^{j2\pi ft_0}\, df \Big|^2\ .$$

The autocorrelation function $R_b(t_1, t_2)$ at the output of the receiver filter when $r(t) = n(t)$

(noise only, no pulse present or $a_0 = 0$) is obtained as follows

$$
\begin{aligned}
R_b(t_1,t_2) = E[b(t_1)\,b^*(t_2)] &= E\big[\big(h_R(t_1) * n(t_1)\big)\,\big(h_R(t_2) * n(t_2)\big)^*\big] \\
&= E\Big[\int_{-\infty}^{\infty} h_R(\mu)\,n(t_1-\mu)\,d\mu \int_{-\infty}^{\infty} h_R^*(\nu)\,n^*(t_2-\nu)\,d\nu\Big] \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_R(\mu)\,h_R^*(\nu) \quad \underbrace{E[n(t_1-\mu)\,n^*(t_2-\nu)]}_{= R_n(t_1-t_2-\mu+\nu) = \int S_n(f)\,e^{j2\pi f(t_1-t_2-\mu+\nu)}df} \quad d\mu\,d\nu \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_R(\mu)\,h_R^*(\nu)\,S_n(f)\,e^{j2\pi f(t_1-t_2-\mu+\nu)}\,df\,d\mu\,d\nu \\
&= \int_{-\infty}^{\infty} S_n(f)\int_{-\infty}^{\infty} h_R(\mu)\,e^{-j2\pi f\mu}\,d\mu \int_{-\infty}^{\infty} h_R^*(\nu)\,e^{j2\pi f\nu}\,d\nu\,e^{j2\pi f(t_1-t_2)}\,df \\
&= \int_{-\infty}^{\infty} S_n(f)\,|H_R(f)|^2\,e^{j2\pi f(t_1-t_2)}\,df = R_b(t_1-t_2)\,.
\end{aligned}
$$

The variance of the noise at time t at the output of the filter $h_R(t)$ is therefore

$$\sigma_b^2 = R_b(0) = \int_{-\infty}^{\infty} S_n(f)\,|H_R(f)|^2\,df\,.$$

Assuming white noise with $S_n(f) = \mathcal{N}_0/2$ for all f, this simplifies to

$$\sigma_b^2 = R_b(0) = \frac{\mathcal{N}_0}{2}\int_{-\infty}^{\infty} |H_R(f)|^2\,df\,.$$

The SNR after sampling at $t = t_0$ in the receiver can now be expressed as

$$\frac{S}{N} = \frac{E[|b_0|^2]}{\sigma_b^2} = \frac{2E[|a_0|^2]}{\mathcal{N}_0}\,\frac{\Big|\int_{-\infty}^{\infty} H_R(f)\,P_C(f)\,e^{j2\pi f t_0}\,df\Big|^2}{\int_{-\infty}^{\infty}|H_R(f)|^2\,df}\,.$$

To maximize S/N by choice of $h_R(t) \Leftrightarrow H_R(f)$, the following theorem comes in handy.

Theorem: Schwartz Inequality. Let $U(f)$ and $V(f)$ be real- or complex-valued and have finite energy. Then

$$\Big|\int_{-\infty}^{\infty} U(f)\,V(f)\,df\Big|^2 \le \int_{-\infty}^{\infty}|U(f)|^2\,df\int_{-\infty}^{\infty}|V(f)|^2\,df\,,$$

with equality iff $V^*(f) = A\,U(f)$ for some real- or complex-valued constant A. □

In the context of the PAM receiver filter this implies that

$$\Big|\int_{-\infty}^{\infty} H_R(f)\,P_C(f)\,e^{j2\pi f t_0}\,df\Big|^2 \le \int_{-\infty}^{\infty}|H_R(f)|^2\,df\int_{-\infty}^{\infty}\underbrace{|P_C(f)\,e^{j2\pi f t_0}|^2}_{=|P_C(f)|^2}\,df\,,$$

and thus

$$\frac{S}{N} \le \frac{2E[|a_0|^2]}{\mathcal{N}_0} \frac{\int_{-\infty}^{\infty} |H_R(f)|^2\,df \int_{-\infty}^{\infty} |P_C(f)|^2\,df}{\int_{-\infty}^{\infty} |H_R(f)|^2\,df} = \frac{2E[|a_0|^2]}{\mathcal{N}_0} \int_{-\infty}^{\infty} |P_C(f)|^2\,df\,,$$

with equality iff

$$P_C^*(f)\,e^{-j2\pi f t_0} = A\,H_R(f) \qquad \Longrightarrow \qquad H_R(f) = \frac{P_C^*(f)\,e^{-j2\pi f t_0}}{A}\,.$$

Substituting this in the equation for $E[b(t)]$ at $t = t_0$ yields

$$E[b_0] = \frac{E[a_0]}{A} \int_{-\infty}^{\infty} |P_C(f)|^2\,df \qquad \Longrightarrow \qquad A = \int_{-\infty}^{\infty} |P_C(f)|^2\,df = \int_{-\infty}^{\infty} |p_C(t)|^2\,dt\,,$$

for $E[b_0] = E[a_0]$. Altogether this proves the following

Theorem: Matched Filter. Let $r(t) = a_0\,p_C(t) + n(t)$ be a received PAM signal with random amplitude a_0, known pulse $p_C(t)$, and additive white noise $n(t)$ with PSD $S_n(f) = \mathcal{N}_0/2$ for all f. Assume that a_0 and $n(t)$ are uncorrelated. Then the filter

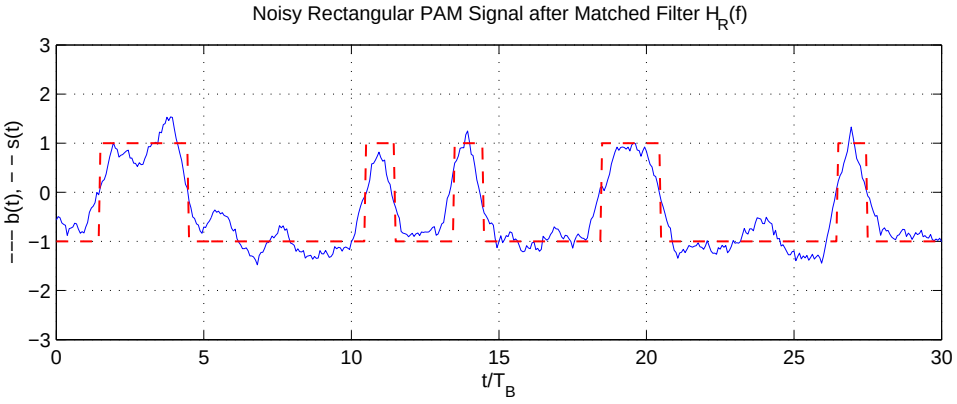
$$h_R(t) = \frac{p_C^*(t_0 - t)}{\int_{-\infty}^{\infty} |p_C(\mu)|^2\,d\mu} \qquad \Longleftrightarrow \qquad H_R(f) = \frac{P_C^*(f)\,e^{-j2\pi f t_0}}{\int_{-\infty}^{\infty} |P_C(\nu)|^2\,d\nu}$$

which is called a matched filter, maximizes the SNR at its output when sampled at $t = t_0$. The expected output value (at $t = t_0$) is $E[a_0]$, with SNR

$$\frac{S}{N} = \frac{2\,E[|a_0|^2]}{\mathcal{N}_0} \int_{-\infty}^{\infty} |p_C(\mu)|^2\,d\mu = \frac{2\,E[|a_0|^2]}{\mathcal{N}_0} \int_{-\infty}^{\infty} |P_C(\nu)|^2\,d\nu\,.$$

□

The plot below shows the output of the matched filter for the same noisy rectantgular PAM signal $r(t)$ that was used as an example in a the section.



The improvement over simple lowpass filtering is clearly visible.

1.4 Root Raised Cosine in Frequency Pulse

The overall pulse $q(t)$ in a PAM communication system with matched filter receiver is

$$q(t) = p_C(t) * h_R(t) \qquad \Longleftrightarrow \qquad Q(f) = P_C(f)\, H_R(f)$$

Assuming $t_0 = 0$ and white noise with $S_n(f) = \mathcal{N}_0/2$, yields

$$Q(f) = \frac{|P_C(f)|^2}{\int_{-\infty}^{\infty} |P_C(\nu)|^2\, d\nu}\,.$$

Thus, to satisfy Nyquist's 1'st criterion for no ISI after the matched filter, $P_C(f)$ has to satisfy

$$\sum_{k=-\infty}^{\infty} |P_C(f - kF_B)|^2 = K\,, \qquad \text{all } f\,,$$

for some constant K. Since the RCf pulse is bandlimited and behaves well in the time domain, it is a natural starting point for finding a bandlimited $P_C(f)$ that satisfies the above formula. Recall that that the RCf pulse with parameter $0 \le \alpha \le 1$ is defined as

$$p(t) = \frac{\sin(\pi t/T_B)}{\pi t/T_B}\,\frac{\cos(\pi\alpha t/T_B)}{1-(2\alpha t/T_B)^2}\,,$$

with Fourier transform

$$P(f) = \begin{cases} T_B\,, & |f| \le \dfrac{1-\alpha}{2T_B}\,, \\ \dfrac{T_B}{2}\left[1+\cos\Big(\dfrac{\pi T_B}{\alpha}\big(|f| - \dfrac{1-\alpha}{2T_B}\big)\Big)\right], & \dfrac{1-\alpha}{2T_B} \le |f| \le \dfrac{1+\alpha}{2T_B}\,, \\ 0\,, & |f| \ge \dfrac{1+\alpha}{2T_B}\,. \end{cases}$$

To obtain the square root of this $P(f)$, use the trigonometric identity

$$\cos^2\alpha = \frac{1}{2}(1+\cos(2\alpha)) \quad \Longrightarrow \quad \sqrt{\frac{1+\cos\beta}{2}} = \cos\big(\frac{\beta}{2}\big)\,.$$

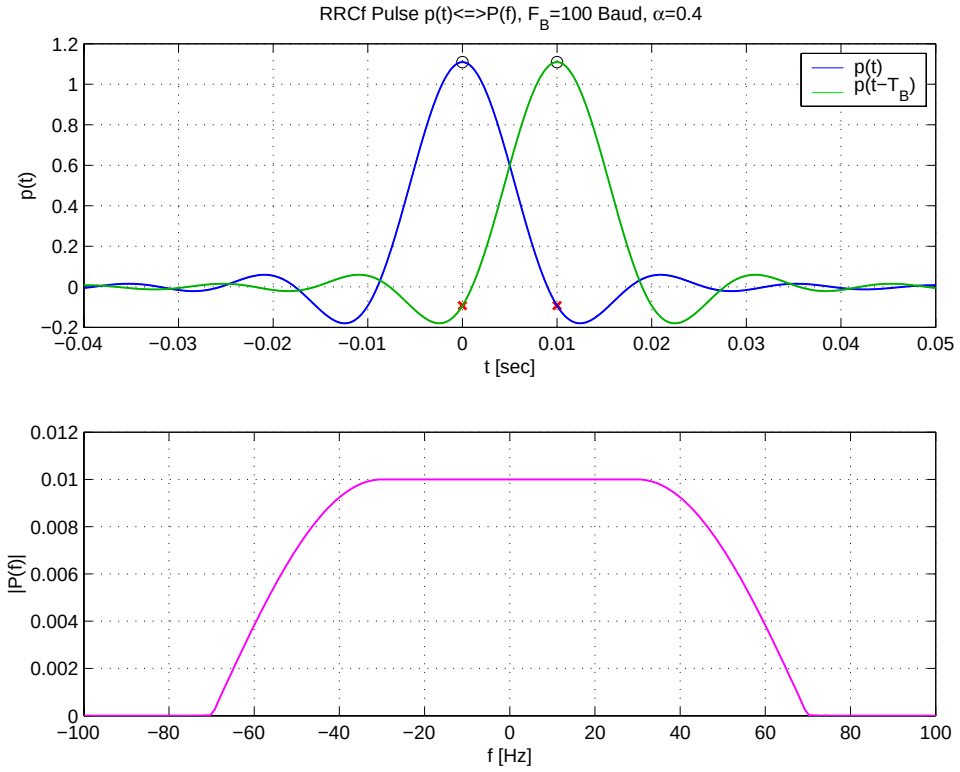
In addition, multiply the resulting spectrum by $\sqrt{T_B}$ (to keep the dc component unchanged) so that

$$P_C(f) = \begin{cases} T_B\,\cos\Big(\dfrac{\pi T_B}{2\alpha}\big(|f| - \dfrac{1-\alpha}{2T_B}\big)\Big), & \dfrac{1-\alpha}{2T_B} \le |f| \le \dfrac{1+\alpha}{2T_B}\,, \\ T_B\,, & |f| \le \dfrac{1-\alpha}{2T_B}\,, \\ 0\,, & \text{otherwise}\,. \end{cases}$$

which is proportional to the **square root of the RCf spectrum**. In the time domain this corresponds to

$$p_C(t) = \begin{cases} \dfrac{T_B}{\pi} \dfrac{\sin\big((1-\alpha)\pi t/T_B\big) + (4\alpha t/T_B)\,\cos\big((1+\alpha)\pi t/T_B\big)}{\big(1-(4\alpha t/T_B)^2\big)\,t}\,, & t \neq 0, \pm\dfrac{T_B}{4\alpha}\,, \\ 1-\alpha+\dfrac{4\alpha}{\pi}\,, & t = 0\,, \\ \dfrac{\alpha}{\sqrt{2}}\left[\big(1+\dfrac{2}{\pi}\big)\,\sin\big(\dfrac{\pi}{4\alpha}\big) + \big(1-\dfrac{2}{\pi}\big)\,\cos\big(\dfrac{\pi}{4\alpha}\big)\right], & t = \pm\dfrac{T_B}{4\alpha}\,, \end{cases}$$

where $0 \leq \alpha \leq 1$. Because of its spectral shape and origin, this pulse will be called the **root raised cosine in frequency (RRCf)** pulse. An example of a RRCf pulse $p(t)$ and $p(t-T_B)$ and its Fourier transform $|P(f)|$ for $\alpha = 0.4$ is shown in the graphs below.

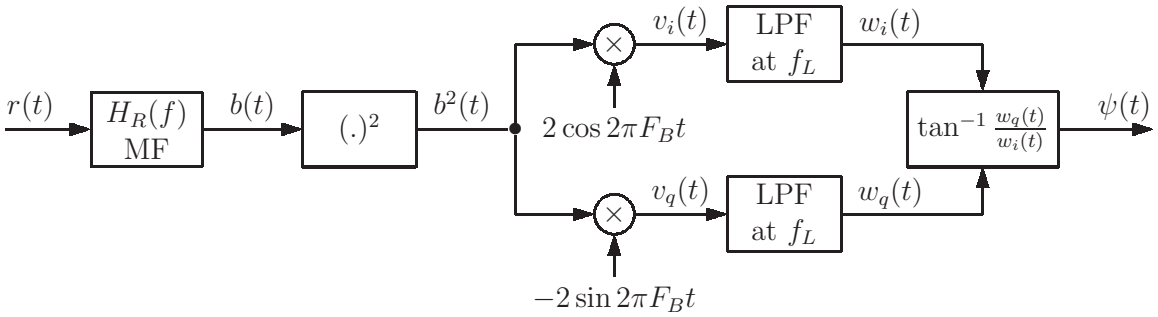


Note that this $p(t)$ creates intersymbol interference, as indicated by the red crosses at the sampling times (black circles), during transmission. After matched filtering at the receiver, however, this intersymbol interference disappears.

1.5 Symbol Timing Synchronization

As shown previously, a spectral component at the baud rate F_B can be extracted from most practical PAM signals by squaring the received signal $r(t)$. To remove as much noise as

possible, it is best to use a matched filter (MF) $H_R(f)$ first and then to square its output $b(t)$ as shown in the following block diagram.



The reason why symbol timing synchronization at the receiver is necessary is that the baud rate F_B used (initially) at the receiver differs from the baud rate $F_B + f_e$ used at the transmitter by the frequency error f_e. In addition, there may also be a phase error θ_e. Reliable reception of a PAM signal is only possible if $f_e = 0$ and θ_e is sufficiently small.

Let F_B be the baud rate used by the receiver and let

$$b_B(t) = A\,\cos\big(2\pi(f_B + f_e)t + \theta_e\big)\,,$$

be the term in $b^2(t)$ that contains the spectral component at the baud rate $F_B + f_e$ used by the transmitter. To estimate f_e and θ_e, $b^2(t)$ is multiplied by $2\cos 2\pi F_B t$ and by $-2\sin 2\pi F_B t$ to obtain

$$\begin{aligned} v_i(t) &\approx 2b_B(t)\cos 2\pi f_B t = A\big[\cos(2\pi f_e t + \theta_e) + \cos(2\pi(2F_B + f_e)t + \theta_e)\big]\,,\\ v_q(t) &\approx 2b_B(t)\cos 2\pi f_B t = A\big[\sin(2\pi f_e t + \theta_e) - \sin(2\pi(2F_B + f_e)t + \theta_e)\big]\,, \end{aligned}$$

where the approximation comes from the fact that $b^2(t)$ also contains other terms than the (strong) spectral component in $b_B(t)$. After lowpass filtering with cutoff frequency $f_L < F_B$

$$\begin{aligned} w_i(t) &\approx A\cos(2\pi f_e t + \theta_e)\,,\\ w_q(t) &\approx A\sin(2\pi f_e t + \theta_e)\,. \end{aligned}$$

Thus after taking the inverse tangent of $w_q(t)/w_i(t)$ (resolved modulo 2π) and unwrapping the phase at jumps by $\pm 2\pi$,

$$\psi(t) \approx 2\pi f_e t + \theta_e\,.$$

This is the (estimated) error between the symbol timing at the transmitter and the symbol timing at the receiver. To obtain a pulse train $c(t)$ for sampling the received signal $b(t)$ at the (estimated) best time instants, compute

$$g(t) = \frac{d}{dt}\big\{\,\mathrm{sgn}\,\big[\sin\big(2\pi F_B t + \psi(t)\big)\big]\big\}\,,$$

where sgn(.) is the signum function ($\mathrm{sgn}(x) = -1$ if $x < 0$, $\mathrm{sgn}(x) = +1$ otherwise), and then set

$$c(t) = \frac{g(t) + |g(t)|}{4}\,.$$

Thus, $c(t)$ has impulses of area 1 at the positive zero crossings of $\sin\big(2\pi F_B t + \psi(t)\big)$.

2 Lab Experiments

E1. LPF with Trapezoidal Frequency Response. (a) The unit impulse response of an LPF with trapezoidal frequency response and -6dB frequency f_L is

$$h(t) = \frac{\sin(2\pi f_L t)}{\pi t} \frac{\sin(2\pi \alpha f_L t)}{2\pi \alpha f_L t} , \qquad 0 \le \alpha \le 1 .$$

The linear "rolloff" region of this LPF extends from $(1-\alpha)f_L$ to $(1+\alpha)f_L$. Make a Python module called `filtfun` and start the module with a function called `trapfilt`. This fcuntion should generate a truncated version $h_T(t)$ of the impulse response given above and use it to filter an input signal $x(t)$ and produce a delay-compensated output signal $y(t)$. More specifically

$$h_T(t) = \begin{cases} h(t) , & -k/(2f_L) \le t \le k/(2f_L) , \\ 0 , & \text{otherwise} , \end{cases}$$

for some integer k and

$$\angle H_T(f) = 0 , \quad -f_L \le f \le f_L .$$

Note that delay compensation is important, for instance, when processing PAM signals, so that the optimum sampling times are not shifted by the lowpass filter. The header and a skeleton for the `trapfilt` function are shown below.

```
# File: filtfun.py
# Module for filter functions
from pylab import *
from scipy.signal import butter, lfilter
import ecen4652 as ecen

def trapfilt0(sig_xt, fL, k, alfa):
    """
    Delay compensated FIR LPF/BPF filter with trapezoidal
    frequency response.
    >>>>> sig_yt, n = trapfilt(sig_xt, fL, k, alfa) <<<<<
    where  sig_yt: waveform from class sigWave
           sig_yt.signal():  filter output y(t), samp rate Fs
           n:    filter order
           sig_xt: waveform from class sigWave
           sig_xt.signal():  filter input x(t), samp rate Fs
           sig_xt.get_Fs():  sampling rate for x(t), y(t)
           fL:   LPF cutoff frequency (-6 dB) in Hz
           k:    h(t) is truncated to |t| <= k/(2*fL)
           alfa: frequency rolloff parameter, linear rolloff
                 over range (1-alfa)fL <= |f| <= (1+alfa)fL
    """
    xt = sig_xt.signal()       # Input signal
    Fs = sig_xt.get_Fs()       # Sampling rate
    ixk = round(Fs*k/float(2*fL))  # Tail cutoff index
    tth = arange(-ixk,ixk+1)/float(Fs)  # Time axis for h(t)
    n = len(tth)-1             # Filter order
    # ***** Generate impulse response ht here *****
    yt = lfilter(ht, 1, hstack((xt, zeros(ixk))))/float(Fs)
                               # Compute filter output y(t)
    yt = yt[ixk:]              # Filter delay compensation
    return ecen.sigWave(yt, Fs, sig_xt.get_t0()), n
                               # Return y(t) and filter order
```

(b) To test `trapfilt`, let $F_s = 16000$ Hz, use a time axis from -0.5 to 0.5 sec, and generate a unit impulse at $t = 0$. Note that this has to be (an approximation to) a CT unit impulse (with area 1 under the impulse). Use this unit impulse as input for `trapfilt` with $f_L = 1000$ Hz, $k = 20$ and $\alpha = 0.2$. Then use your `showft` function to plot the magnitude and the phase of the unit impulse response. Display the magnitude both linear on an absolute scale, and relative in dB. Compare your graphs to the ones given in the introduction. In particular, verify that the passband has a magnitude of 1 and that the transition from passband to stopband is linear. Verify also that the filter is properly delay-compensated (explain how you did this). How do the results change when you set $\alpha = 0$?

(c) The file `pamsig601.wav` contains a noisy polar binary PAM signal. Look at the PSD of this signal and then use `trapfilt` to lowpass filter the signal with $f_L \approx 1000$ Hz, $k \approx 10$, and $\alpha \approx 0.2$. Determine the baud rate F_B (look at the PSD of the lowpass filtered signal

squared), and then use this information to display the eye diagram (use a width of 3 T_B and superimpose about 100 traces) of the lowpass filtered PAM signal. Change f_L to find the value that gives you the largest eye opening. How is this f_L related to F_B?

(d) Implementation and Measurement of 'trapfilt' in GNU Radio. Use the header specification shown below to start a new Python module `filtspecs.py` with a function called `trapfilt_taps` that computes the taps for a FIR LPF with trapezoidal frequency response.

```
# File: filtspecs.py
# Module for filter specfifications
# FIR: Determine filter taps
# IIR: Determine numerator (b) and denominator (a)
# polynomial coefficients
from pylab import *
from scipy.signal import butter

def trapfilt_taps(N, phiL, alfa):
    """
    Returns taps for order N FIR LPF with trapezoidal frequency
    response, normalized cutoff frequency phiL = fL/Fs, and rolloff
    parameter alfa.
    >>>>> hLn = trapfilt_taps(N, phiL, alfa) <<<<<
    where  N:    filter order
           phiL: normalized cutoff frequency (-6 dB)
           alfa: frequency rolloff parameter, linear rolloff
                 over range (1-alfa)phiL <= |f| <= (1+alfa)phiL
    """
```

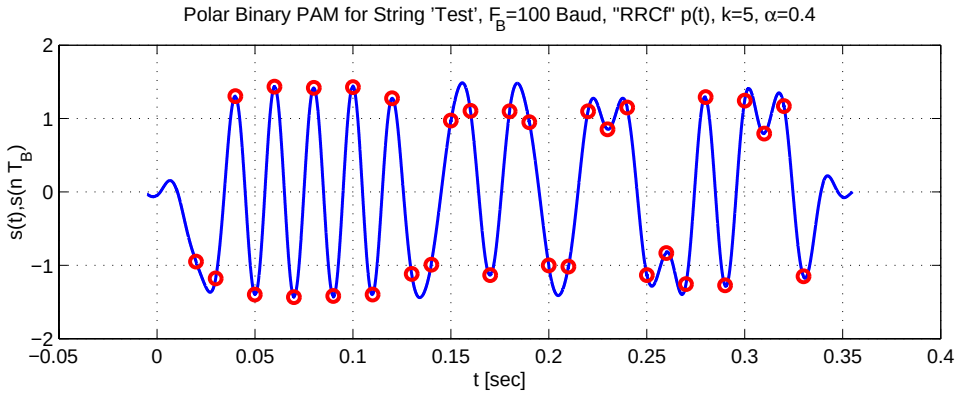
Place the `filtspecs.py` module in one of the directories in your `PYTHONPATH` and generate a GNU Radio flowgraph that can be used to test the `trapfilt` taps in a decimating or interpolating FIR filter block. Describe your measurement procedure and how the results that you measured verify the correct implementation of the LPF with a trapezoidal frequency response in GNU Radio.

E2. PAM Transmitter/Receiver. (a) Update your `pam11` function to include the **RRCf** (root raised cosine in frequency) pulse which is defined as

$$p(t) = \begin{cases} \dfrac{T_B}{\pi} \dfrac{\sin\big((1-\alpha)\pi t/T_B\big) + (4\alpha t/T_B)\cos\big((1+\alpha)\pi t/T_B\big)}{\big(1-(4\alpha t/T_B)^2\big)\,t}\,, & t \neq 0, \pm\dfrac{T_B}{4\alpha}\,, \\ 1-\alpha+\dfrac{4\alpha}{\pi}\,, & t = 0\,, \\ \dfrac{\alpha}{\sqrt{2}}\Big[\big(1+\dfrac{2}{\pi}\big)\sin\big(\dfrac{\pi}{4\alpha}\big) + \big(1-\dfrac{2}{\pi}\big)\cos\big(\dfrac{\pi}{4\alpha}\big)\Big]\,, & t = \pm\dfrac{T_B}{4\alpha}\,, \end{cases}$$

where $0 \leq \alpha \leq 1$. Call the new function `pam12` (PAM, V1.2). The `ptype` for the RRCf pulse is `'rrcf'` and the parameter format is `pparms = [k alpha]`, with the same meanings as

for the RCf pulse. An example of polar binary PAM using an RRCf pulse with $\alpha = 0.4$ and "tail length" $k = 4$ is shown in the following graph.



Note that the RRCf pulse has ISI at the sampling time instants. After filtering with a matched filter, however, the ISI disappears.

(b) For the module `pamfun.py`, write a Python function called `pamrcvr10` (PAM receiver V1.0) which implements a receiver, with matched filter and sampler, for (noisy) PAM signals. Here is the header for this function:

***** need to adapt for use with `sigWave` and `sigSequ` class signals *****

```
def pamrcvr10(tt, rt, FBparms, ptype, pparms=[]):
    """
    Pulse amplitude modulation receiver with matched filter:
    r(t) -> b(t) -> bn.
    V1.0 for 'man', 'rcf', 'rect', 'rrcf', 'sinc', and 'tri'
    pulse types.
    >>>>> bn, bt, ixn = pamrcvr10(tt, rt, FBparms, ptype, pparms) <<<<<
    where  tt:    time axis for r(t)
           rt:    received (noisy) PAM signal r(t)
           FBparms: = [FB, dly]
           FB:    Baud rate of PAM signal, TB=1/FB
           dly:   sampling delay for b(t) -> b_n as a fraction of TB
                  sampling times are t=n*TB+t0 where t0 = dly*TB
           ptype: pulse type from list
                  ('man','rcf','rect','rrcf','sinc','tri')
           pparms not used for 'man','rect','tri'
           pparms = [k, alpha]  for 'rcf','rrcf'
           pparms = [k, beta]  for 'sinc'
           k:     "tail" truncation parameter for 'rcf','rrcf','sinc'
                  (truncates p(t) to -k*TB <= t < k*TB)
           alpha: rolloff parameter for ('rcf','rrcf'), 0<=alpha<=1
           beta:  Kaiser window parameter for 'sinc'
           bn:    received DT sequence after sampling at t=n*TB+t0
           bt:    received PAM signal b(t) at output of matched filter
           ixn:   indexes where b(t) is sampled to obtain b_n
    """
```

The following commands implement `pamrcvr10` for rectangular $p(t)$.

```
    FB = FBparms[0]
    t0 = FBparms[1]
    Fs = (len(tt)-1)/(tt[-1]-tt[0])
    # ***** Set up matched filter response h_R(t) *****
    TB = 1/float(FB)           # Time per symbol
    ptype = ptype.lower()      # Convert ptype to lowercase
                               # Set left/right limits for p(t)
    if (ptype=='rect'):        # Rectangular or Manchester p(t)
          kL = -0.5; kR = -kL
    else:
        kL = -1.0; kR = -kL    # Default left/right limits
    ixpL = ceil(Fs*kL*TB)      # Left index for p(t) time axis
    ixpR = ceil(Fs*kR*TB)      # Right index for p(t) time axis
    ttp = arange(ixpL,ixpR)/float(Fs)  # Time axis for p(t)
    pt = zeros(len(ttp))       # Initialize pulse p(t)
    if (ptype=='rect'):        # Rectangular p(t)
        ix = where(logical_and(ttp>=kL*TB, ttp<kR*TB))[0]
        pt[ix] = ones(len(ix))
    else:
        print("ptype '%s' is not recognized" % ptype)
    hRt = pt[::-1]             # h_R(t) = p(-t)
    hRt = Fs/sum(np.power(pt,2.0))*hRt  # h_R(t) normalized
    # ***** Filter r(t) with matched filter h_R(t)
    bt = convolve(rt,hRt)/float(Fs)  # Matched filter b(t)=r(t)*h_R(t)
    bt = bt[-ixpL-1:len(tt)-ixpL-1]  # Trim b(t)
    # ***** Sample b(t) at t=n*TB+t0 to obtain b_n *****
    N = ceil(FB*(tt[-1]-tt[0]))    # Number of symbols
    ixn = array(around((arange(N)+0.5+t0)*Fs/float(FB)),int64)
                               # Sampling indexes
    ix = where(logical_and(ixn>=0,ixn<len(tt)))[0]
    ixn = ixn[ix]              # Trim to existing indexes
    bn = bt[ixn]               # DT sequence sampled at t=n*TB+t0
    return bn, bt, ixn
```

Complete this function by adding the remaining pulse types. Note that the generation of $p(t)$ in `pamrcvr10` is essentially identical to the $p(t)$ generation in `pam12`.

(c) Reception of noisy PAM signals. The wav-files `pamsig601.wav`, `pamsig602.wav`, and `pamsig603.wav` contain received PAM signals $r(t)$ with additive white Gaussian noise. The data in the PAM signals is ASCII text, 8 bits/character, LSB-first, transmitted as polar binary with logical $0 \rightarrow -A$ and logical $1 \rightarrow +A$ where A is an amplitude level that was chosen so that the wav-files (with the noise added) have amplitude less than 1. Use `showpsd` in combination with `trapfilt` and time domain plots to analyze the PAM signals sufficiently so that you can use your `pamrcvr10` function to recover the ASCII text in each case.

E3. Matched Filters and ISI. (Experiment for ECEN 5002, optional for ECEN 4652) Devise a test using `pam12.m`, `pamrcvr10.m`, and `showeye.m` to determine for which of the

pulse types (`'man', 'rcf', 'rect', 'rrcf', 'sinc', 'tri'`) there is ISI (intersymbol interference) either in the received PAM signal $r(t)$ and/or in the signal $b(t)$ after the matched filter. If there is ISI, estimate (in percent of the full size) by how much the "eye" in the eye diagram gets closed because of ISI.

