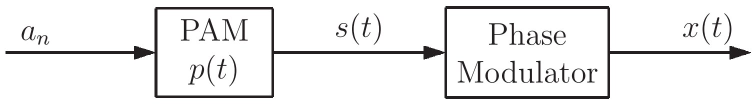
Lab 10: Phase and Hybrid Amplitude/Phase Shift Keying, Carrier Sync

1 Introduction

A sinusoid like $A\cos(2\pi ft+\theta)$ is characterized by its amplitude A, its frequency f and its phase θ. Thus, after having seen amplitude shift keying (ASK) and frequency shift keying (FSK) as digital modulation methods, it is quite natural to also consider phase shift keying (PSK) to transmit digital data. An advantage of both FSK and PSK is that they are insensitive to amplitude variations of the received signal. Because of its very nature, namely transmitting information through phase changes, a PSK signal needs to be received with a coherent receiver, as opposed to FSK and ASK that can also be received using non-coherent techniques. The bandwidth requirements for a given bitrate, however, are smaller for PSK than for ASK and FSK. In practice, the use of M-PSK is usually limited to $M\le 8$ distinct phases. For larger M a combination of ASK and PSK, termed hybrid APSK in this lab description, is generally used. One convenient way to implement hybrid APSK is to split up the digital data into in-phase and quadrature data sequences, using pulse amplitude modulation (PAM) for each and then feed the resulting continuous-time (CT) waveforms into the in-phase and quadrature channels of a QAM (quadrature amplitude modulation) transmitter.

1.1 Phase Shift Keying

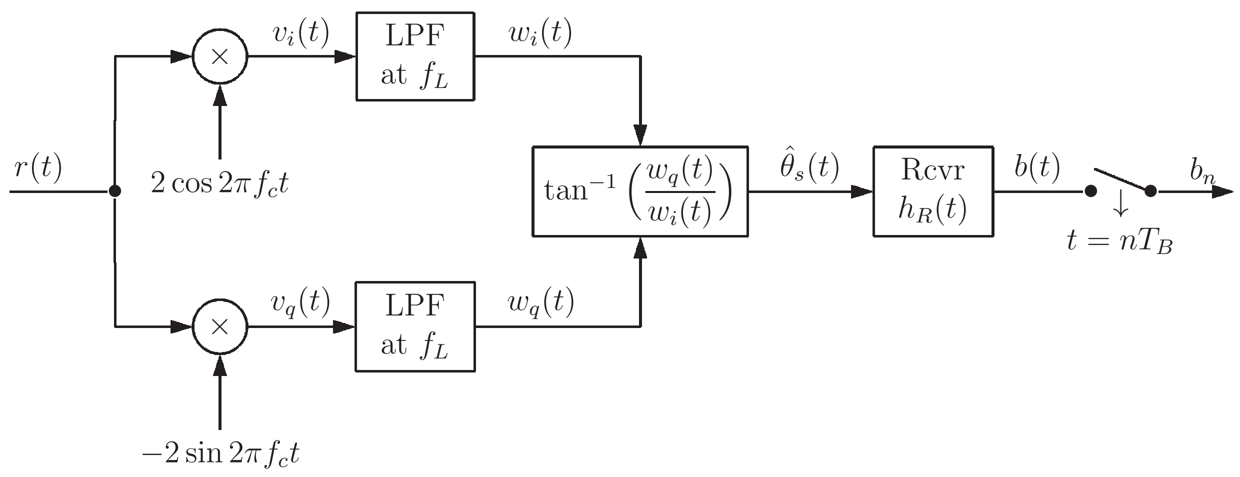
Let a_n denote a DT sequence with baud rate $F_B=1/T_B$. The following block diagram can be used to first convert this sequence into a CT PAM signal $s(t)$, which is then fed into a phase modulator (PM) to generate a PM signal $x(t)$ at some carrier frequency f_c.



Mathematically, the PM signal $x(t)$ can be expressed as

$$x(t)=A_c\cos\big(2\pi f_ct+\theta_s(t)\big)\,,\qquad\text{where}\quad \theta_s(t)=\Delta_\theta\,s(t)=\Delta_\theta\sum_{n=-\infty}^{\infty}a_n\,p(t-nT_B)\,,$$

where $p(t)$ is a PAM pulse, and where Δ_θ is a suitably chosen phase deviation constant. To demodulate a received PM signal $r(t)$ produced by the above circuit, the blockdiagram shown below can be used.



Note that, in order to cover the whole range from 0 to 2π (or $-\pi$ to $+\pi$) for $\theta(t)$, a four-quadrant version of $\tan^{-1}$, such as `atan2(wq,wi)` or `arctan2(wq,wi)`, has to be used. Next, assume that

$$r(t) = \cos\big(2\pi f_c t + \Delta_\theta\, s(t)\big)\,, \qquad \text{with} \quad s(t) = \sum_{n=-\infty}^{\infty} a_n\, p(t - nT_B)\,.$$

Then

$$v_i(t) = 2\,r(t)\,\cos(2\pi f_c t) = \cos\big(\Delta_\theta\, s(t)\big) + \cos\big(4\pi f_c t + \Delta_\theta\, s(t)\big)\,,$$
$$v_q(t) = -2\,r(t)\,\sin(2\pi f_c t) = \sin\big(\Delta_\theta\, s(t)\big) - \sin\big(4\pi f_c t + \Delta_\theta\, s(t)\big)\,.$$

After lowpass filtering and sampling at time $t = nT_B$ this becomes

$$w_i(t) = \cos\big(\Delta_\theta\, s(t)\big) \quad\Longrightarrow\quad w_i(nT_B) = w_i[n] = \cos\big(\Delta_\theta\, a_n\big)\,,$$
$$w_q(t) = \sin\big(\Delta_\theta\, s(t)\big) \quad\Longrightarrow\quad w_q(nT_B) = w_q[n] = \sin\big(\Delta_\theta\, a_n\big)\,,$$

where it is assumed that the PAM pulse $p(t)$ satisfies Nyquist's first criterion for no ISI, i.e.,

$$p(nT_B) = \begin{cases} 1\,, & n = 0\,, \\ 0\,, & \text{otherwise}\,. \end{cases} \quad\Longrightarrow\quad s(nT_B) = \sum_{k=-\infty}^{\infty} a_k\, p(nT_B - kT_B) = a_n\,.$$

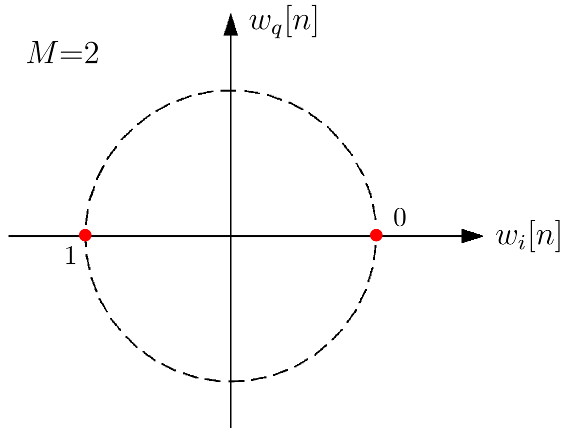
The quantities $w_i[n]$ and $w_q[n]$ can be interpreted as the real and imaginary parts, respectively, of a point on the unit circle at angle $\Delta_\theta\, a_n$ in the complex plane. If a_n can only take on M discrete values, e.g., $a_n \in \{0, 1, \ldots, M-1\}$, then the resulting M points can be spread equally around the unit circle, each representing the transmission of a different M-ary symbol. The following table shows the actual angle values used when

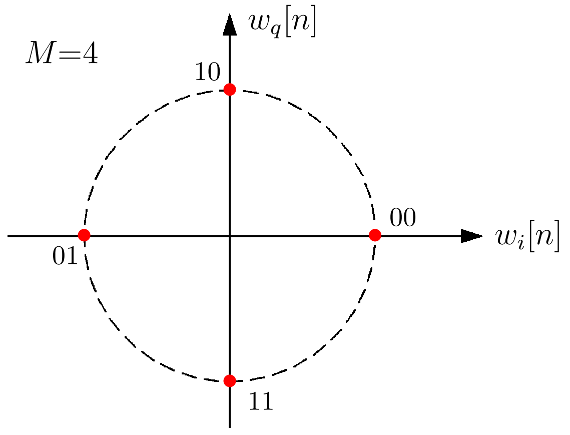
$$a_n \in \{0, 1, \ldots, M-1\}\,, \qquad \Delta_\theta = \frac{2\pi}{M}\,,$$

and M is a power of 2 in the range $2^1, \ldots, 2^4$.

M	Δ_θ	an	$\tan^{-1}\big(w_q[n]/w_i[n]\big)$
2	π	$\{0,1\}$	$\{0,\pi\}$
4	$\pi/2$	$\{0,1,2,3\}$	$\{0,\pi/2,\pi,3\pi/2\}$
8	$\pi/4$	$\{0,1,2,\ldots,7\}$	$\{0,\pi/4,\pi/2,3\pi/4,\ldots,7\pi/4\}$
16	$\pi/8$	$\{0,1,2,\ldots,15\}$	$\{0,\pi/8,\pi/4,3\pi/8,\ldots,15\pi/8\}$

Plotting $w_q[n]$ versus $w_i[n]$ for all M values that a_n can take on results in graphs similar to the ones shown below for $M=2$ and $M=4$.

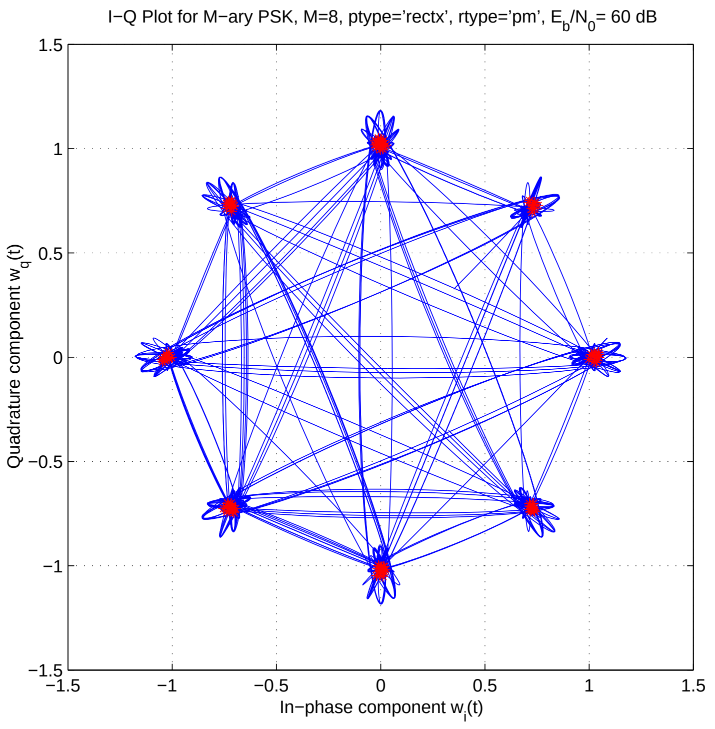




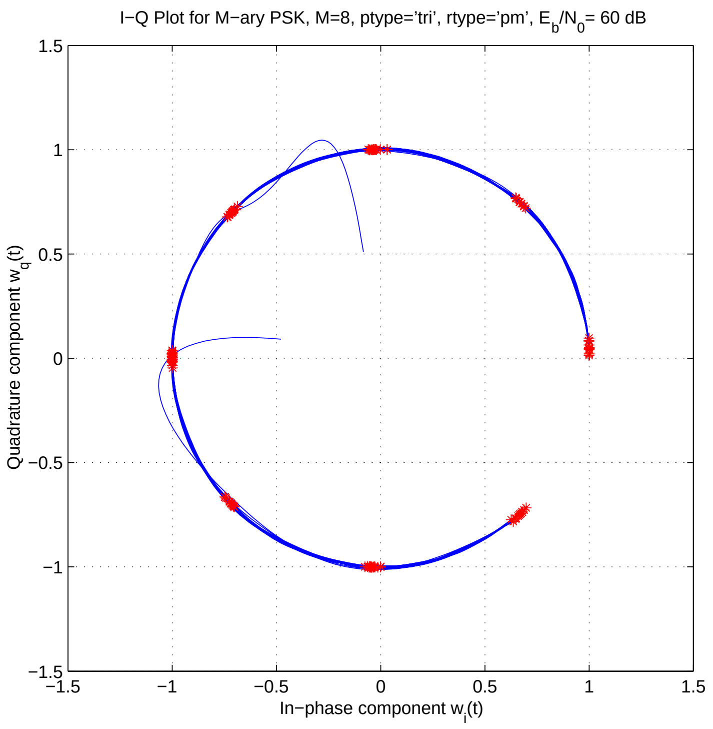
The graph which shows the locations of the M signal points and their geometrical relationships with respect to each other is called a **signal constellation**. For **M-ary phase shift keying (PSK)**, the signal constellation consists of M points spread equally around a (unit) circle in a 2-dimensional signal space. Such constellations are also called **polar signal constellations (PSC)**. If M is a power of 2, e.g., $M=2^m$, then each point in the signal constellation corresponds to a particular pattern of m transmitted bits. There are many ways in which bits can be associated with signal points, including Gray coding (so that only one bit changes between adjacent signal points). For the purposes of these notes it is assumed that M-ary symbols are integers in the range $0,1,\ldots M-1$ which correspond to their binary expansions, with the LSB written first. Binary representations for symbols in a_n are shown for $M=2^1\ldots2^4$ in the table below.

M	Values of a_n	Corresponding Binary Strings (LSB first)
2	$\{0,1\}$	$\{0,1\}$
4	$\{0,1,2,3\}$	$\{00,10,01,11\}$
8	$\{0,1,2,\ldots,7\}$	$\{000,100,010,110,001,101,011,111\}$
16	$\{0,\ldots,6,7,8,9,\ldots,15\}$	$\{0000,\ldots,0110,1110,0001,1001,\ldots,1111\}$

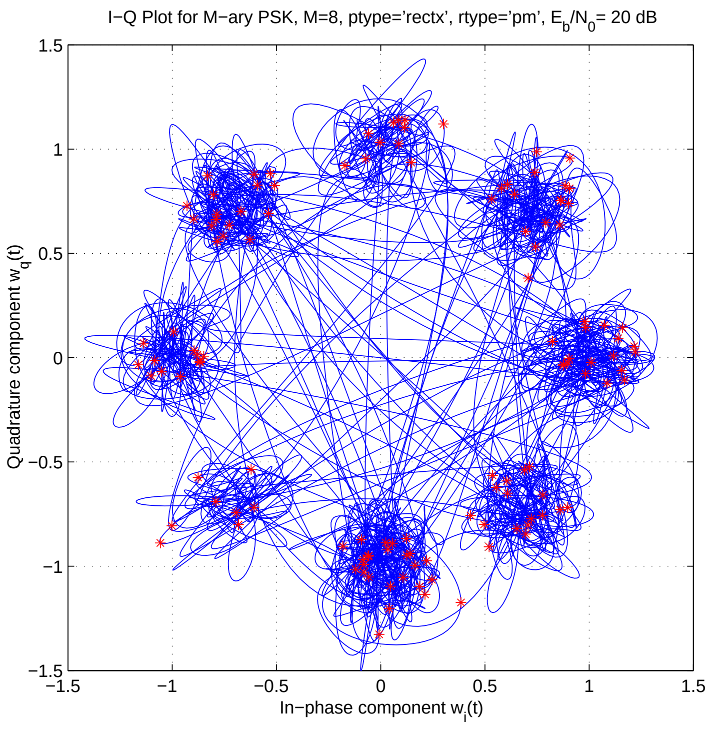
The signal constellations given previously only show the sampled in-phase and quadrature components $w_i[n]$ and $w_q[n]$ at the outputs of the lowpass filters in the PM receiver. A more general **I-Q plot**, which in addition also shows the transitions between the signal points, is obtained by also plotting $w_q(t)$ versus $w_i(t)$. The transition paths depend on how the PAM pulse $p(t)$ is chosen. The following graph shows an I-Q plot for 8-PSK with a rectangular pulse $p(t)$ of width $T_B = 1/F_B$ where F_B is the baud rate of the transmitted DT sequence a_n.



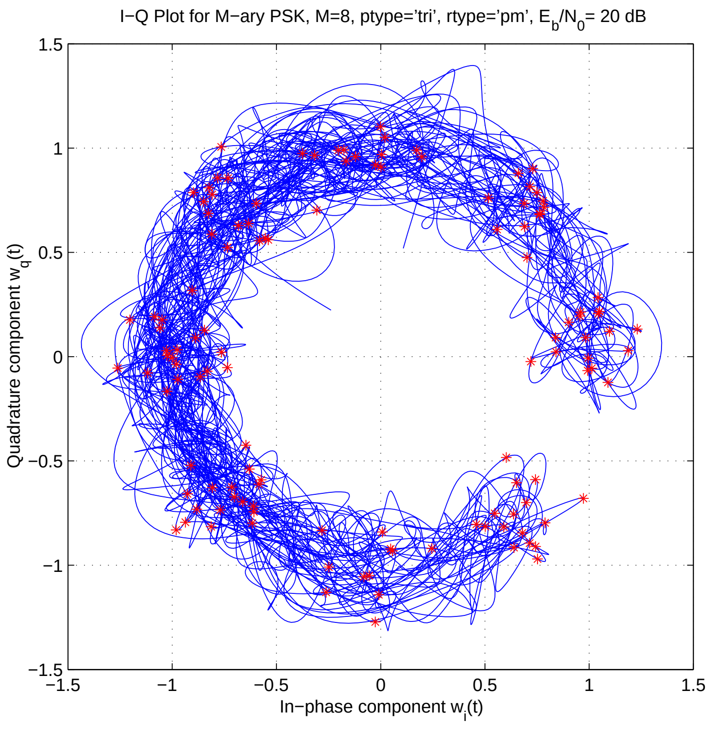
If $p(t)$ is rectangular as in the above plot, then the transitions between signal points are essentially straight lines. If $p(t)$ is triangular as shown in the next plot, then the phase changes occur more gradual and the transistions between signal points all occur on the circle on which the signal points are located. Note that this implies that the amplitude of the transmitted signal is constant (which is not the case for rectangular $p(t)$).



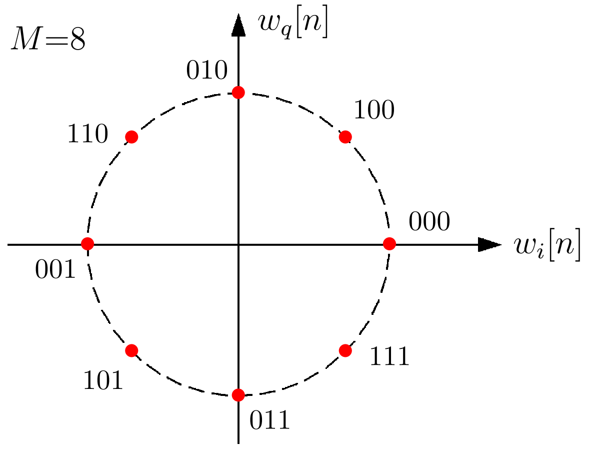
In a real communication system the received signal is usually noisy. In this case, the signal points (and the transitions between them) are scattered around their nominal locations. An example for $M = 8$, rectangular $p(t)$, and a signal-to-noise ratio (SNR) of $E_b/\mathcal{N}_0 = 20$ dB is shown in the following graph.

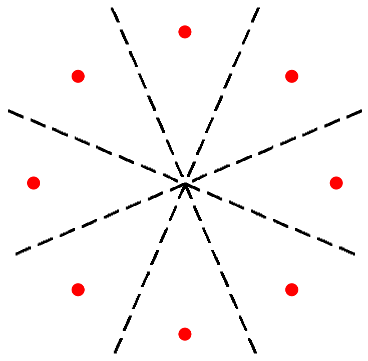


If $p(t)$ is triangular instead of rectangular, this changes as shown next.



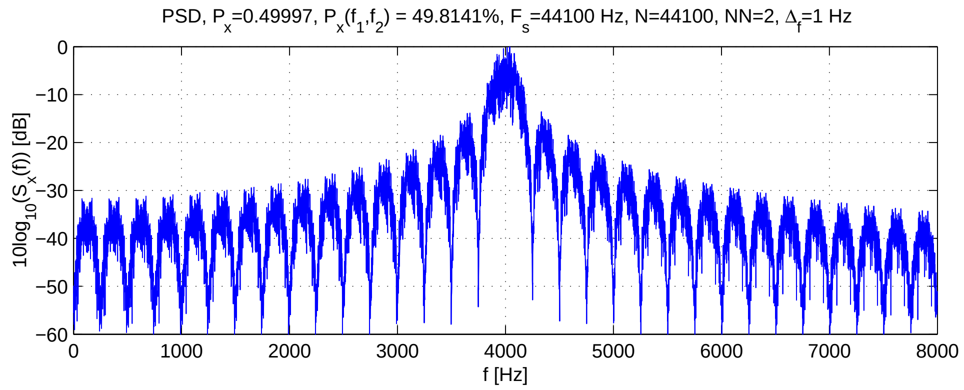
Assuming that the in-phase and quadrature noise components have equal power, and all M signal points are equally likely, the decision rule at the receiver is to assume that the most likely transmitted signal is the one that is closest in (Euclidean) distance to the received signal point. This leads to wedge-shaped **decision regions**, similar to cutting up a birthday cake into M equal slices. A signal constellation for 8-PSK is shown in the left figure below. The corrsponding **maximum likelihood (ML)** decision regions are shown in the graph on the right, with the dashed lines representing the boundaries between decision regions.



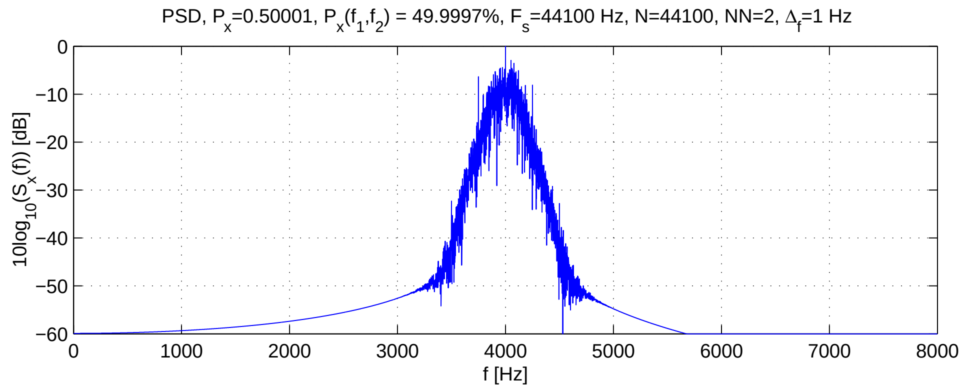


As the SNR decreases, some received signal points will eventually cross the decision region boundaries, which in turn leads to symbol and bit errors at the receiver output.

The next graph shows the PSD of a M-ary PSK signal with $M = 8$, $f_c = 4000$ Hz, rectangular $p(t)$, and a baud rate of $F_B = 250$. The main signal energy is concentrated in the band $f_c - F_B \ldots f_c + F_B$, but beyond that the signal energy decreases only relatively slowly, which leads to interference with other users in frequency division multiplexed systems.



In practice, the "tails" of such a spectrum either have to be removed because of FCC (Federal Communications Commission) regulations, or are removed by the transfer function of the transmission channel. In either case, just bandpass filtering the PSK signal leads to intersymbol interference (ISI) at the receiver, which can degrade the performance of the communication system, especially in the presence of noise. It is generally more desirable to control the PSD of the PSK signal by choosing a suitable PAM pulse $p(t)$, e.g., using a pulse with raised cosine in frequency (RCf) spectrum, or root RCf (RRCf) spectrum. If this is done by first converting a_n to a PAM signal $s(t)$, which then modulates a PM transmitter, the result is a **continuous phase modulation (CPM)** signal $x(t)$. The spectrum of such a M-ary PSK signal with $M = 8$, $f_c = 4000$ Hz, RCf $p(t)$, and a baud rate of $F_B = 250$ is shown in the following figure.



This can be demodulated using a phase detector, followed by a filter matched to $p(t)$. The block diagram of such a receiver was shown previously. However, it turns out that this kind of receiver is suboptimum if $p(t)$ is different from a rectangular pulse. To see why this is true, consider the **PM-based PSK** signal

$$x(t) = \cos\big(2\pi f_c t + \Delta_\theta\, s(t)\big)$$
$$= \underbrace{\cos\big(\Delta_\theta\, s(t)\big)}_{=\,w_i(t)} \cos 2\pi f_c t - \underbrace{\sin\big(\Delta_\theta\, s(t)\big)}_{=\,w_q(t)} \sin 2\pi f_c t\,,$$

where $s(t) = \sum_k a_k\, p(t-kT_B)$. Written out, the in-phase term is

$$w_i(t) = \cos\big(\Delta_\theta\, s(t)\big) = \cos\Big(\Delta_\theta \sum_{k=-\infty}^{\infty} a_k\, p(t-kT_B)\Big) \neq \sum_{k=-\infty}^{\infty} \cos(\Delta_\theta\, a_k)\, p(t-kT_B)\,,$$

and the quadrature term is

$$w_q(t) = \sin\big(\Delta_\theta\, s(t)\big) = \sin\Big(\Delta_\theta \sum_{k=-\infty}^{\infty} a_k\, p(t-kT_B)\Big) \neq \sum_{k=-\infty}^{\infty} \sin(\Delta_\theta\, a_k)\, p(t-kT_B)\,.$$

Because of the "$\neq$" statements above, PM-based PSK signals cannot be demodulated directly with matched filters in the phase detector. Thus, the SNR is not maximized before the (nonlinear) $\tan^{-1}$ function, which in turn leads to subpotimum performance in the presence of noise.

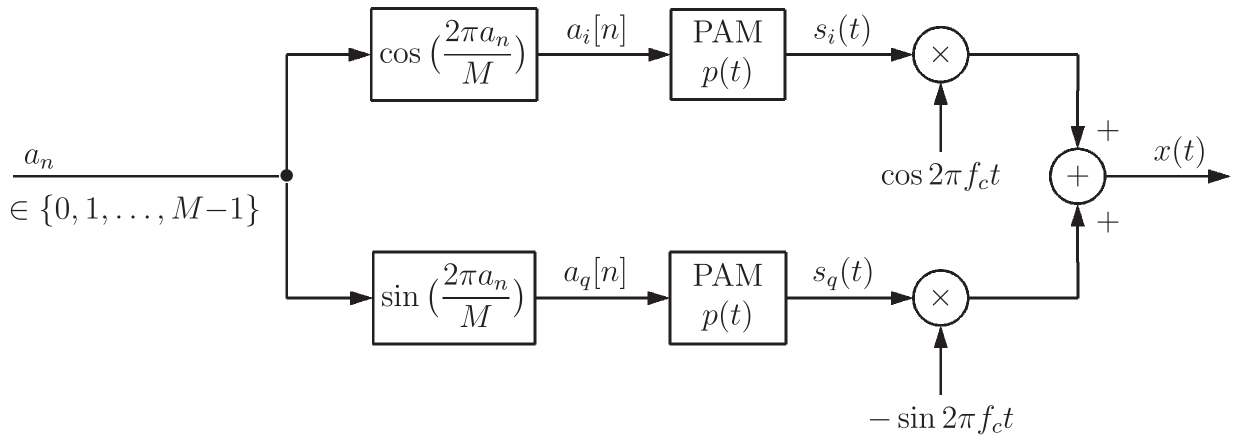
Another approach that is very often used in practice is **QAM-based PSK** which splits the DT sequence a_n up into an in-phase component $a_i[n]$ and a quadrature component $a_q[n]$ by setting

$$a_i[n] = \cos\big(\Delta_\theta\, a_n\big)\,, \quad \text{and} \quad a_q[n] = \sin\big(\Delta_\theta\, a_n\big)\,.$$

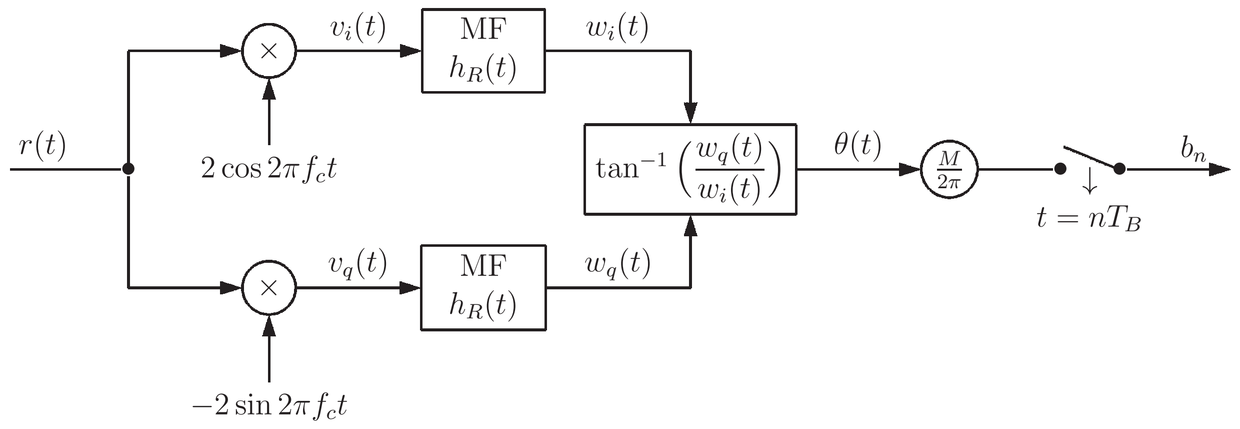
Each of these two signals is then converted separately into a PAM signal, using the same pulse $p(t)$ in most cases. The two PAM signals are then fed into a quadrature amplitude modulator (QAM) to form a PSK signal $x(t)$ of the form

$$x(t) = \sum_{n=-\infty}^{\infty} a_i[n]\, p(t-nT_B)\, \cos 2\pi f_c t - \sum_{n=-\infty}^{\infty} a_q[n]\, p(t-nT_B)\, \sin 2\pi f_c t\,.$$

The blockdiagram of such a transmitter, with $a_n \in \{0,1,\ldots,M-1\}$ and $\Delta_\theta = 2\pi/M$, is shown below.



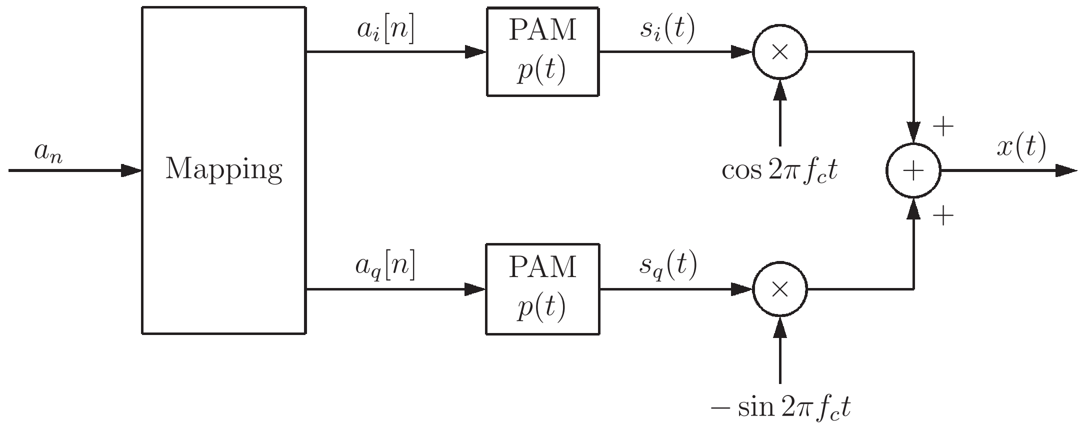
The advantage of using separate in-phase and quadrature components is that now a QAM receiver like the one shown next can be used, where the LPF's are replaced by matched filters (MF), matched to $p(t)$ (assuming an ideal transmission channel).



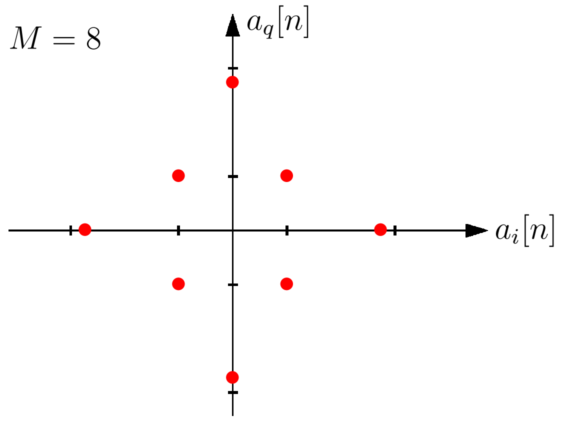
To make scatter plots and I-Q plots from this receiver, just simply plot $w_q(nT_B)$ versus $w_i(nT_B)$ or $w_q(t)$ versus $w_i(t)$, in the same way as done before with the PM receiver.

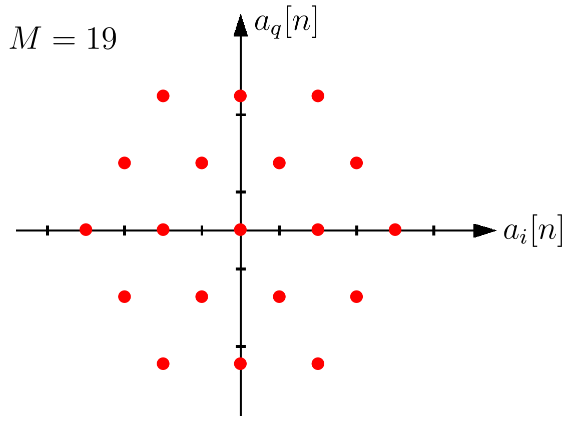
1.2 Hybrid Amplitude/Phase Shift Keying

Hybrid M-ary amplitude/phase shift keying (APSK) is a logical extension of QAM-based M-ary PSK. For APSK both the amplitude and the phase of the carrier can be modulated simultaneously, resulting in QAM signal constellations. Using the blockdiagram below, a M-ary DT sequence a_n is split up into a M_i-ary in-phase sequence $a_i[n]$ and a M_q-ary quadrature sequence $a_q[n]$, where $M_i \times M_q = M$. The two sequences are then converted individually to waveforms using PAM with a pulse $p(t)$ and combined into a single QAM signal $x(t)$ at carrier frequency f_c using a regular QAM modulator.

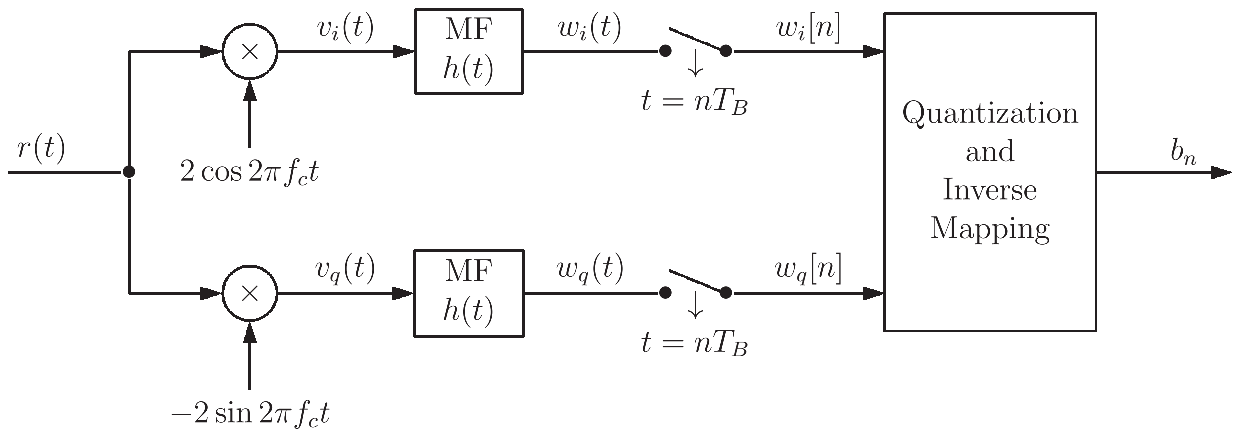


The signal constellation is obtained by plotting $a_q[n]$ versus $a_i[n]$. The geometry of the signal constellation is determined by the mapping from a_n to $a_i[n]$ and $a_q[n]$. In principle, any 2-dimensional constellation, such as the two examples for $M = 8$ and $M = 19$ shown below, can be used as QAM signal constellation for hybrid APSK.

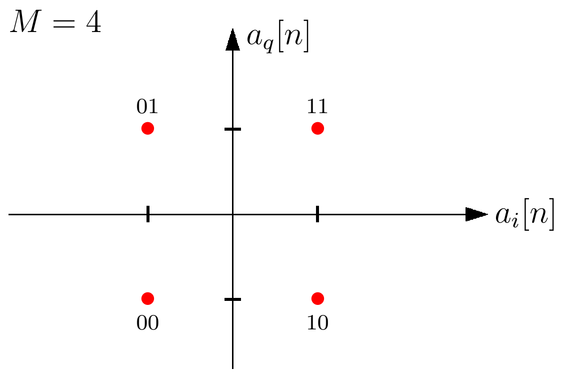


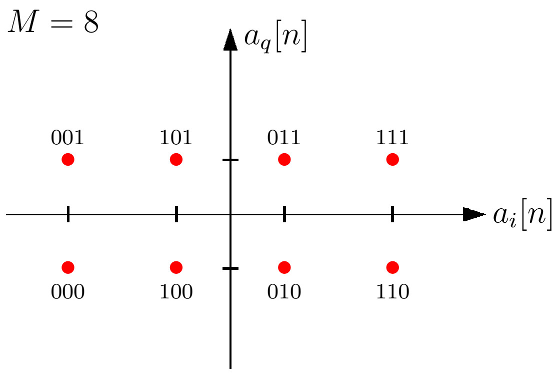


However, since in general a receiver as shown in the following blockdiagram is used, rectangular or **cartesian signal constellations (CSC)** are preferred because they lead to simpler receiver implementation.

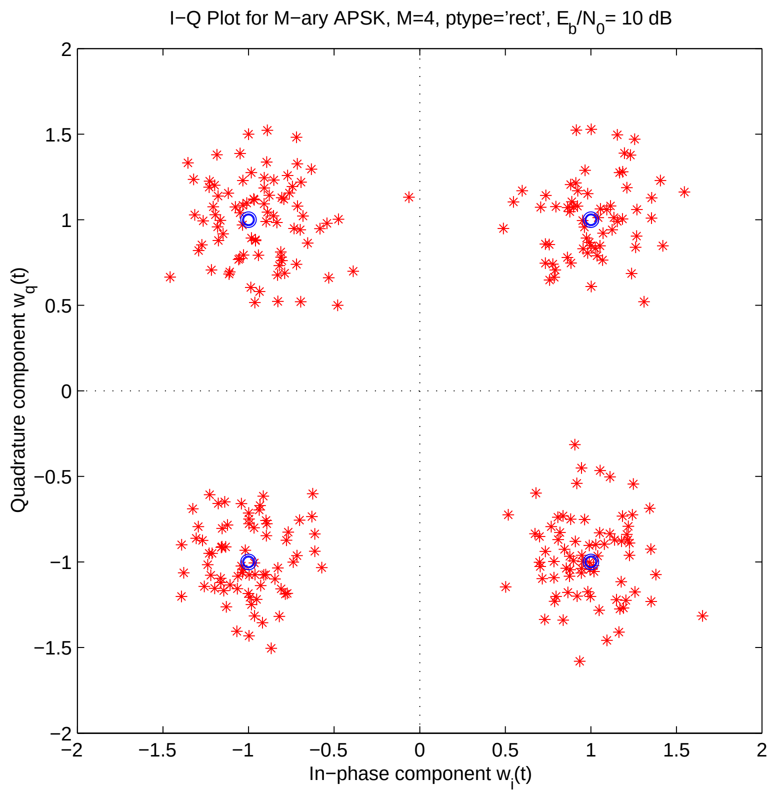


CSC constellations for $M = 4$ and $M = 8$ are shown in the next two figures. For $M = 4$ it is quite natural to use one bit per symbol for $a_i[n]$ and the other one for $a_q[n]$. For $M = 8$ it is less clear how three bits should be split up between the in-phase and quadrature data sequences. A somewhat arbitrary choice when m in $M = 2^m$ is odd, is to use $m_i = \lceil m/2 \rceil$ bits for the in-phase sequence and $m_q = \lfloor m/2 \rfloor$ bits for the quadrature sequence. The bit assignments for each signal point were made as follows. The first m_i out of every m bits are assigned to the in-phase sequence $a_i[n]$, with the LSB coming first (in the leftmost position). The remaining m_q out of every m bits are similarly assigned to the quadrature sequence $a_q[n]$, again with the LSB coming first.

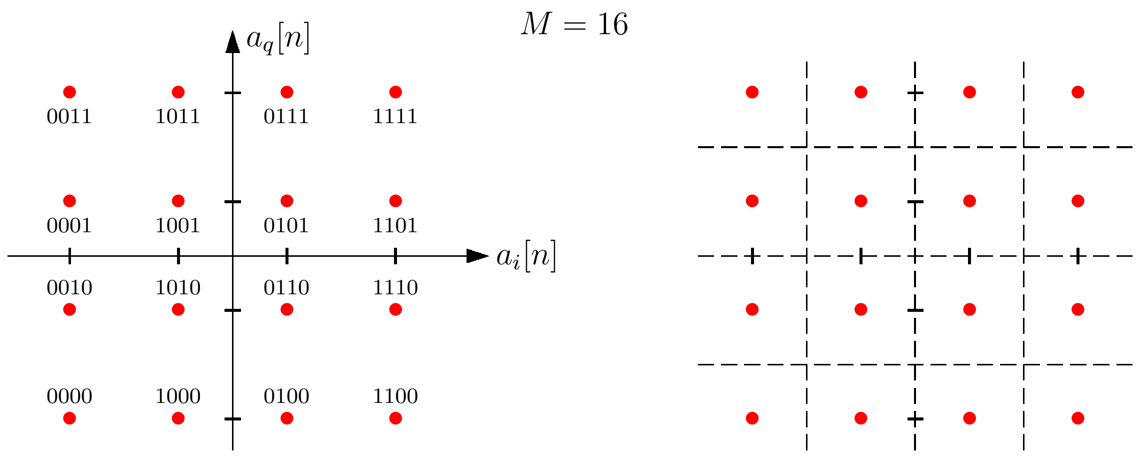




Apart from the usual problem of synchronizing the receiver with the transmitter, the main problem of the receiver is to associate the received noisy sample pairs $(w_i[n], w_q[n])$ with the most likely transmitted signal point $(a_i[n], a_q[n])$. A maximum liklelihood (ML) receiver measures the Euclidean distance between $(w_i[n], w_q[n])$ and all possible $(a_i[n], a_q[n])$, and outputs that value of b_n that corresponds to the signal point which is closest to the received point. The graph below shows some received signal points for a noisy QAM signal when a $M = 4$ CSC constellation is used.



The horizontal and vertical dashed lines represent the decision boundaries for a ML receiver. Because they are orthogonal, decisions on the in-phase and quadrature components can be made independently without loss of optimality, which simplifies the decision and quantization process significantly. The next two figures show the bit assignments and the decision boundaries for a $M = 16$ QAM-CSC signal.



WiFi signals use similar constellations with up to about 8 bits per symbol (256-QAM).

1.3 Carrier Synchronization

In addition to extracting symbol rate information from the received signal $r(t)$, it is also necessary to extract carrier synchronization information at a PSK or APSK receiver. In a digital receiver the starting point is to convert the received real bandpass signal to a complex-valued lowpass signal, so that the sampling rate for further processing can be minimized.

Assume that the receiver uses $\cos(2\pi f_c t)$ for the carrier reference, but the transmitter actually used $\cos(2\pi f_c t + \theta_e(t))$, where $\theta_e(t)$ is a time-varying phase error, as carrier. In complex notation the received signal is of the form

$$r(t) = \mathrm{Re}\{s_L(t)\, e^{j(2\pi f_c t + \theta_e(t))}\} = \frac{s_L(t)\, e^{j(2\pi f_c t+\theta_e(t))} + s_L^*(t)\, e^{-j(2\pi f_c t+\theta_e(t))}}{2} ,$$

where $s_L(t)$ is a complex-valued baseband PAM signal. For CPM M-PSK with data d_n

$$r(t) = \cos\big(2\pi f_c t + \theta_e(t) + \Delta_\theta\, s(t)\big) ,$$

and thus

$$s_L(t) = \cos(\Delta_\theta\, s(t)) + j\, \sin(\Delta_\theta\, s(t)) = e^{j\Delta_\theta\, s(t)} , \qquad \Delta_\theta = \frac{2\pi}{M} ,$$

where

$$s(t) = \sum_{n=-\infty}^{\infty} d_n\, p(t - nT_B) , \qquad d_n \in \{0, 1, \ldots, M-1\} ,$$

and $p(t)$ is a real-valued PAM pulse. For QAM with in-phase data $d_i[n]$ and quadrature data $d_q[n]$

$$r(t) = s_i(t)\, \cos(2\pi f_c t + \theta_e(t)) - s_q(t)\, \sin(2\pi f_c t + \theta_e(t)) ,$$

and therefore

$$s_L(t) = s_i(t) + j\, s_q(t) ,$$

with

$$s_i(t) = \sum_{n=-\infty}^{\infty} d_i[n]\, p(t - nT_B)\,, \qquad \text{and} \qquad s_q(t) = \sum_{n=-\infty}^{\infty} d_q[n]\, p(t - nT_B)\,.$$

For QAM-based M-PSK with data sequence $d_n \in \{0, 1, \ldots, M-1\}$

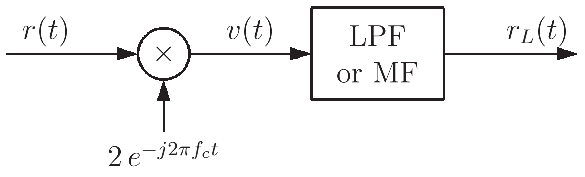
$$d_i[n] = \cos\Big(\frac{2\pi d_n}{M}\Big)\,, \qquad \text{and} \qquad d_q[n] = \sin\Big(\frac{2\pi d_n}{M}\Big)\,,$$

and thus

$$s_L(t) = \sum_{n=-\infty}^{\infty} \big(d_i[n] + j\, d_q[n]\big)\, p(t - nT_B) = \sum_{n=-\infty}^{\infty} e^{j2\pi d_n/M}\, p(t - nT_B)\,,$$

where $p(t)$ is a real-valued PAM pulse.

The following block diagram can be used to convert the received real bandpass signal $r(t)$ to a complex-valued lowpass signal $r_L(t)$.



After multiplication by the complex exponential

$$\begin{aligned} v(t) = 2\, r(t)\, e^{-j2\pi f_c t} &= \big[s_L(t)\, e^{j(2\pi f_c t + \theta_e(t))} + s_L^*(t)\, e^{-j(2\pi f_c t + \theta_e(t))}\big]\, e^{-j2\pi f_c t} \\ &= s_L(t)\, e^{j\theta_e(t)} + s_L^*(t)\, e^{-j(4\pi f_c t + \theta_e(t))}\,, \end{aligned}$$

and thus, after lowpass filtering,

$$r_L(t) = s_L(t)\, e^{j\theta_e(t)}\,.$$

Thus, for CPM-based M-PSK,

$$r_L(t) = \exp\big(j \sum_{n=-\infty}^{\infty} \frac{2\pi\, d_n}{M}\, p(t - nT_B)\big)\, e^{j\theta_e(t)}\,, \qquad d_n \in \{0, 1, \ldots, M-1\}\,,$$

and, for QAM-based M-PSK,

$$r_L(t) = \sum_{n=-\infty}^{\infty} e^{j2\pi d_n/M}\, p(t - nT_B)\, e^{j\theta_e(t)}\,, \qquad d_n \in \{0, 1, \ldots, M-1\}\,.$$

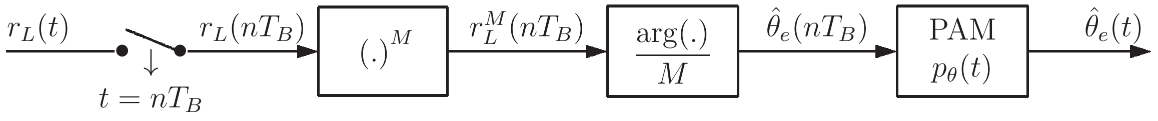
If there is no ISI and $p(nT_B) = \delta_n$, then the samples at $t = nT_B$ are in both cases

$$r_L(nT_B) = e^{j2\pi d_n/M}\, e^{j\theta_e(nT_B)}\,.$$

Since d_n takes on integer values, the first (data-dependent) term can be eliminated by raising $r_L(nT_B)$ to the M-th power, i.e.,

$$r_L^M(nT_B) = e^{jM\theta_e(nT_B)} .$$

This leads to the following blockdiagram for estimating first $\hat{\theta}_e(nT_B)$ and then (assuming $\theta_e(t)$ is bandlimited to $F_B/2$) $\hat{\theta}_e(t)$ by using PAM with a sinc pulse $p_\theta(t)$.



Note that for this to work it is crucial that the sampling times $t = nT_B$ are estimated accurately. The see how this can be done for QAM-based signals, start from

$$r_L(t) = \big(s_i(t) + j\, s_q(t)\big)\, e^{j\theta_e(t)} ,$$

where both $s_i(t)$ and $s_q(t)$ are real-valued PAM signals. Computing the magnitude squared yields

$$|r_L(t)|^2 = s_i^2(t) + s_q^2(t) .$$

Except for PAM signals with rectangular $p(t)$, this has a spectral component at F_B which can be filtered out and used to synchronize the sampling circuit at the receiver. For CPM-based signals, start from

$$r_L(t) = e^{j\Delta_\theta s(t)}\, e^{j\theta_e(t)} ,$$

where $s(t)$ is a real-valued PAM signal. In this case $|r_L(t)|^2$ will not have a useable spectral component at F_B. However, the first derivative with respect to t is

$$\frac{dr_L(t)}{dt} = j\,\frac{d}{dt}\big[\Delta_\theta s(t) + \theta_e(t)\big]\, e^{j\Delta_\theta s(t)}\, e^{j\theta_e(t)} .$$

Taking the magnitude squared yields

$$\Big|\frac{dr_L(t)}{dt}\Big|^2 = \Big[\Delta_\theta \frac{ds(t)}{dt} + \frac{d\theta_e(t)}{dt}\Big]^2 ,$$

which in most cases contains a spectral component at F_B. Note that, if $\theta_e(t)$ is a slowly varying phase error, then $d\theta_e(t)/dt \approx 0$.

1.4 PSK and APSK in GNU Rasio

– To be completed –

2 Lab Experiments

E1. M-ary PSK. (a) For the Python module `keyfun.py`, write a function called `pskxmtr` which generates an M-PSK signal from an M-ary data sequence d_n (taking on values in $\{0, 1, \ldots, M-1\}$) with baud rate F_B. Depending on the value of `xtype`, the function should generate either PM-based or QAM-based PSK signals. The header for the `pskxmtr` function is given below.

```
def pskxmtr(M,sig_dn,Fs,ptype,pparms,xtype,fcparms):
    """
    M-ary Phase Shift Keying (PSK) Transmitter for
    PM-based ('pm') and QAM-based ('qam') PSK Signals
    >>>>> sig_xt,sig_st =
                pskxmtr(M,sig_dn,Fs,ptype,pparms,xtype,fcparms) <<<<<
    where  sig_xt: waveform from class sigWave
           sig_xt.signal():   transmitted PSK signal, sampling rate Fs
           sig_xt.timeAxis(): time axis for x(t), starts at t=-TB/2
           sig_st: waveform from class sigWave
           sig_st.signal():   Baseband PAM signal s(t) for 'pm'
           sig_st.signal():   st = sit + 1j*sqt  for 'qam'
           M:       number of distinct symbol values in d[n]
           xtype:   Transmitter type from set {'pm','qam'}
           sig_dn: sequence from class sigSequ
           sig_dn.signal() = [dn]
           dn:      M-ary (0,1,..,M-1) N-symbol DT input sequence d_n
           sig_dn.get_FB(): baud rate of d_n, TB=1/FB
           Fs:      sampling rate of x(t)
           ptype:   pulse type from set
                    {'man','rcf','rect','rrcf','sinc','tri'}
           pparms = []         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'}
           fcparms = [fc, thetac]
           fc:      carrier frequency
           thetac:  carrier phase in deg (0: cos, -90: sin)
    """
```

Use $F_s = 44100$ Hz, $f_c = 4000$ Hz, $\theta_c = 0°$, and $F_B = 300$ Hz to generate M-PSK signals for $M = 2, 4$ with rectangular $p(t)$ and RCf $p(t)$ with $\alpha = 0.5$ and $k \approx 10$. Use random M-ary data to produce a PSK signal of length at least 1 sec. Look at a characteristic piece of $x(t)$ in the time domain, and look at the PSD of $x(t)$. Do this for both `xtype='pm'` and `xtype='qam'` and compare the results. Then look at the PSDs of $x^i(t)$, $i = 2$ for $M = 2$, and $i = 2, 4$ for $M = 4$, again for both `xtype='pm'` and `xtype='qam'`. Look for spectral lines that tell you something about the baud rate F_B and/or the carrier frequency f_c.

(b) In the `keyfun.py` module, complete the following PSK receiver function, called `pskrcvr`. Its purpose is to receive and demodulate the M-ary PSK signals that are generated by the `pskxmtr` function. Note that there are two receiver modes, `rtype='pm'` and `rtype='qam'`, corresponding to the transmitter modes with the same names.

```
def pskrcvr(M,sig_rt,rtype,fcparms,FBparms,ptype,pparms):
    """
    M-ary Phase Shift Keying (PSK) Receiver for
    PM-based ('pm') and QAM-based ('qam') Reception of PSK signals
    >>>>> sig_bn,sig_wt,ixn =
                pskrcvr(M,sig_rt,rtype,fcparms,FBparms,ptype,pparms) <<<<<
    where  sig_bn: sequence from class sigSequ
           sig_bn.signal(): received DT sequence b[n]
           sig_wt: waveform from class sigWave
           sig_wt.signal(): wt = wit + 1j*wqt,  LPF outputs for ('pm')
           sig_wt.signal(): wt = wit + 1j*wqt,  MF outputs for ('qam')
           wit:      in-phase filter output
           wqt:      quadrature filter output
           ixn:      sampling time indexes for b(t)->b[n], w(t)->w[n]
           M:        number of distinct PSK phases
           sig_rt: waveform from class sigWave
           sig_rt.signal():   received (noisy) PSK signal r(t)
           sig_rt.timeAxis(): time axis for r(t)
           rtype:    receiver type from list {'pm','qam'}
           fcparms = [fc, thetac]
           fc:       carrier frequency
           thetac:   carrier phase in deg (0: cos, -90: sin)
           FBparms = [FB, dly]
           FB:      baud rate of PAM signal, TB=1/FB
           dly:     sampling delay for w(t)->w[n], 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 = []  for {'man','rect','tri'}
           pparms = [k, alpha]  for {'rcf','rrcf'}
           pparms = [k, beta]   for {'sinc'}
           k:       "tail" truncation parameter for {'rcf','rrcf','sinc'}
                    (truncates at -k*TB and k*TB)
           alpha:   Rolloff parameter for {'rcf','rrcf'}, 0<=alpha<=1
           beta:    Kaiser window parameter for {'sinc'}
    """
```

Use pskxmtr to generate M-ary PSK signals of duration $\approx$ 1 sec for $M = 2, 4, 8$ with rectangular $p(t)$ and RCf $p(t)$ with $\alpha = 0.5$ and $k \approx 10$. Choose $F_s = 44100$ Hz, $f_c = 4000$ Hz, $\theta_c = 0°$, and $F_B = 300$ Hz as in part (a) and generate PSK signals for both xtype='pm' and xtype='qam'. With the help of pskrcvr in rtype='pm' mode (use this mode even if the signal was generated with xtype='qam'), make I-Q plots that show both $w_q(t)$ versus $w_i(t)$ and $w_q[n]$ versus $w_i[n]$ to verify the correctness of your Matlab PSK modulator and demodulator functions. Characterize the differences between the PM-based and the QAM-based PSK signals.

(c) Generate a 4-PSK signal $x(t)$ using pskxmtr with xtype='pm', and a rectangular $p(t)$. Let $F_s = 44100$ Hz, $f_c = 4000$ Hz, $\theta_c = 0°$, $F_B = 300$ baud, and use random 4-ary data to generate a signal of length $\approx$ 1 sec. Use trapfilt (with $k \approx 40$ and $\alpha \approx 0.05$) to bandpass filter $x(t)$ with a passband from $f_c - 2F_B$ to $f_c + 2F_B$. The output from trapfilt is the received signal $r(t)$. Analyze $r(t)$ in the time and frequency domains, and generate an *I*-*Q* plot which also shows the (ideal) signal points at the sampling times $t = nT_B$. Describe the effect of the BPF on the PSK signal.

(d) Repeat (c), but this time let $p(t)$ be an RCf pulse with $\alpha = 0.5$ and $k \approx 10$. What conclusions can you draw from comparing the results in (c) and (d)?

E2. Hybrid ASK/PSK. (a) Here is the header for a Python function (in keyfun.py called apskxmtr which is used to generate M-ary APSK CSC (amplitude/phase shift keying, cartesian signal constellation) signals for $M = 2^m$. If m is even, then the signal points are arranged on a square grid, otherwise a rectangular grid is used with the longer egde in the in-phase direction. The nominal signal constellation points are at the intersections of ± 1, ± 3, etc, in the in-phase and quadrature directions.

```
def apskxmtr(M,sig_dn,Fs,ptype,pparms,fcparms):
    """
    M-ary (M=2**m) Hybrid Amplitude/Phase Shift Keying (APSK)
    Transmitter for Cartesian Signal Constellations (CSC)
    >>>>> sig_xt,sig_st = apskxmtr(M,sig_dn,Fs,ptype,pparms,fcparms) <<<<<
    where  sig_xt: waveform from class sigWave
           sig_xt.signal():   transmitted APSK signal, sampling rate Fs
           sig_xt.timeAxis(): time axis for x(t), starts at t=-TB/2
           sig_st: waveform from class sigWave
           sig_st.signal(): st = sit + 1j*sqt  baseband PAM signal
           M=2**m:  number of distinct symbol values in d[n]
           m even:  Mi=Mq=2**(m/2) (square constellation)
           m odd:   Mi=2**((m+1)/2), Mq=2**((m-1)/2)
           sig_dn: sequence from class sigSequ
           sig_dn.signal() = [dn]
           dn:      M-ary (0,1,..,M-1) N-symbol DT input sequence d_n
                    first ceil(m/2) bits go to in-phase component,
                    last floor(m/2) bits go to quadrature component
           sig_dn.get_FB(): baud rate of d_n, TB=1/FB
           Fs:      sampling rate of x(t)
           ptype:   pulse type from set
                    {'man','rcf','rect','rrcf','sinc','tri'}
           pparms = []         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'}
           fcparms = [fc, thetac]
           fc:      carrier frequency
           thetac:  carrier phase in deg (0: cos, -90: sin)
    """
```

Look at the APSK signals generated by `apskxmtr` with parameters $F_s = 44100$ Hz, $f_c = 4000$ Hz, $\theta_c = 0°$, $F_B = 300$ baud, for $M = 4, 8, 16$ and a rectangular pulse $p(t)$ of width T_B. Use random binary data to produce $x(t)$ of length ≈ 1 sec and examine the PSD of $x(t)$ and a short piece of $x(t)$ itself in the time domain.

(b) The goal of the function `apskrcvr` whose header is shown below is to demodulate M-ary APSK CSC signals for $M = 2^m$ and to convert them back to the most likely M-ary sequence. It is assumed that the received signal `r` is scaled properly so that the nominal signal points are at the intersections of ± 1, ± 3, etc, of the demodulated in-phase and quadrature components.

```
def apskrcvr(M,sig_rt,fcparms,FBparms,ptype,pparms):
    """
    M-ary (M=2**m) Hybrid Amplitude/Phase Shift Keying (APSK)
    Receiver for Cartesian Signal Constellations (CSC)
    >>>>> sig_bn,sig_wt,ixn =
                    apskrcvr(M,sig_rt,fcparms,FBparms,ptype,pparms) <<<<<
    where  sig_bn: sequence from class sigSequ
           sig_bn.signal(): received DT sequence b[n]
           sig_wt: waveform from class sigWave
           sig_wt.signal(): wt = wit + 1j*wqt,  matched filter outputs
           wit:      in-phase filter output
           wqt:      quadrature filter output
           ixn:      sampling time indexes for b(t)->b[n], w(t)->w[n]
           M=2**m:  number of signal points
           m even:  Mi=Mq=2**(m/2) (square constellation)
           m odd:   Mi=2**((m+1)/2), Mq=2**((m-1)/2)
           sig_rt: waveform from class sigWave
           sig_rt.signal():   received (noisy) PSK signal r(t)
           sig_rt.timeAxis(): time axis for r(t)
           fcparms = [fc, thetac]
           fc:       carrier frequency
           thetac:   carrier phase in deg (0: cos, -90: sin)
           FBparms = [FB, dly]
           FB:      baud rate of PAM signal, TB=1/FB
           dly:     sampling delay for w(t)->w[n], 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 = []  for {'man','rect','tri'}
           pparms = [k, alpha]  for {'rcf','rrcf'}
           pparms = [k, beta]   for {'sinc'}
           k:       "tail" truncation parameter for {'rcf','rrcf','sinc'}
                    (truncates at -k*TB and k*TB)
           alpha:   Rolloff parameter for {'rcf','rrcf'}, 0<=alpha<=1
           beta:    Kaiser window parameter for {'sinc'}
    """
```

Test `apskrcvr` together with `apskxmtr` using parameters $F_s = 44100$ Hz, $f_c = 4000$ Hz, $\theta_c = 0°$, $F_B = 300$ baud, and a rectangular $p(t)$ of width T_B. In particular, make signal constellation and I-Q plots for $M = 4, 8, 16$ and check that they look right.

(c) Generate an APSK CSC signal of length ≈ 1 sec for $M = 8$ with the same parameters (F_s, f_c, F_B, and $p(t)$) as in (b). How does the I-Q plot change if you set $\theta_c = 30°$ and $\theta_c = 90°$ at the transmitter (leaving $\theta_c = 0°$ for the receiver)?

E3. Analysis and Demodulation of (A)PSK Signals. (Experiment for ECEN 5002, optional for ECEN 4652) **(a)** The function `bin2m` (to be added to the `ascfun.py` module) shown below converts a binary ($\{0,1\}$) string `dn` into an M-ary ($\{0,1,\ldots M-1\}$) string `an` for $M = 2^m$. That is, each symbol in `an` corresponds to m bits in `dn`, using an LSB-first conversion.

```
def bin2m(dn, m):
    """
    Binary (LSB first) to M-ary symbol conversion for M=2**m
    >>>>> an = bin2m(dn, m) <<<<<
    where  an   M-ary {0,1,...,M-1} output sequence
           dn   binary {0,1} input sequence
           m    number of bits per M-ary symbol
    """
    L = m*ceil(len(dn)/float(m))    # Make L multiple of m
    dn = hstack((dn,zeros(L-len(dn))))  # Pad dn with zeros
    B = array(reshape(dn,(L/float(m),m)),int)
              # Rows of B are bits of symbols (left bit is LSB)
    p2 = np.power(2,arange(m))      # Powers of 2, smallest first
    an = dot(B,p2)                  # M-ary sequence
    return an
```

To convert a string of M-ary symbols ($M = 2^m$) back to a binary string, the `m2bin` function shown next can be used.

```
def m2bin(an, m):
    """
    M-ary to binary (LSB first) symbol conversion for M=2**m
    >>>>> dn = m2bin(an, m) <<<<<
    where  dn   binary {0,1} output sequence
           an   M-ary {0,1,...,M-1} input sequence
           m    number of bits per M-ary symbol
    """
    p2 = np.power(2.0,arange(0,-m,-1))
                                   # Powers of 2, largest first
    B = mod(floor(outer(an,p2)),2)
              # Rows of B are bits of symbols (left bit is LSB)
    dn = array(reshape(B,(1,m*len(an))),int)
                                   # Binary output sequence
    return dn[0]
```

To generate an M-ary (A)PSK signal from an ASCII text string, first use `asc2bin` to make a binary string, and then `bin2m` to make an M-ary string that is then used as input for `(a)pskxmtr`. To receive an M-ary (A)PSK signal that contains ASCII text, first demodulate using `(a)pskrcvr`, then use `m2bin`, followed by `bin2asc` to recover the text message. The wav files `qbf4psk.wav`, `qbf8psk.wav`, `qbf4apsk.wav`, `qbf8apsk.wav`, and `qbf16apsk.wav` are test files for $M = 4, 8$ PSK and $M = 4, 8, 16$ APSK, containing the text "`The quick brown fox ....` The parameters that were used to make the files are $F_s = 44100$ Hz, $f_c = 4000$ Hz, $\theta_c = 0°$, $F_B = 300$ baud, and $p(t)$ a rectangular pulse of width $T_B = 1/F_B$. Check that you can correctly extract the test sentence from each of the wav files. **Hints:** Remember that the time axis starts at $-T_B/2$. Also, the transmitted signals $x(t)$ had to be scaled to fit in the ± 1 amplitude range needed for wav files. To find the correct factors for undoing the scaling, look at eye diagrams or scatter plots of the signals received from the wav files.

(b) Analyze and demodulate the signals in the files `apsksig1001.wav`, `apsksig1002.wav`, `apsksig1003.wav`, and `apsksig1004.wav`. All signals are (A)PSK transmissions which contain English ASCII text and use either a rectangular $p(t)$ of width T_B or a RRCf $p(t)$ with $\alpha = 0.5$. When demodulating these signals, remember that the time axis starts at $t = -T_B/2$, and make sure you choose the right scaling for the received signal so that it fits the decision boundaries of your quantizer and inverse mapping. Look at eye diagrams or scatter plots to determine the correct scaling factors.