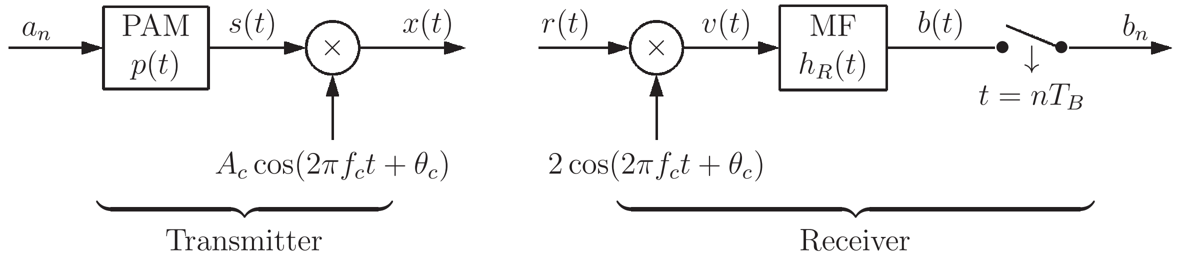
Lab 9: Amplitude, Frequency, and Phase Shift Keying

1 Introduction

Amplitude modulation (AM) can easily be used for the transmission of digital data if the (analog) message signal $m(t)$ is replaced by a (digitally) pulse amplitude modulated (PAM) signal $s(t)$. The simplest method to tansmit binary data is on-off keying (OOK), whereby the transmitter is off for sending 0's, and a carrier at frequency f_c is transmitted for sending 1's (or vice versa). If the carrier oscillator at the transmitter is just turned on and off, then the phase for each 1 is random and a non-coherent receiver must be used. If phase coherence is maintained at the transmitter, then either coherent or non-coherent reception is possible. Instead of multiplying the carrier oscillator by 0 or 1 for binary data, it can be multiplied by -1 or $+1$, thereby producing phase changes by $\pm 180°$ at the carrier frequency f_c. This method is called binary phase shift keying (BPSK) and requires a coherent receiver. An extension of this is phase shift keying (PSK) with four phases, e.g., using 0°, $\pm 90°$ and 180°, for a 4-ary PAM signal. This is called quaternary phase shift keying (QPSK). Instead of using a single carrier frequency f_c for transmitting 0's and 1's, two distinct frequencies, e.g., f_{c0} and f_{c1} can be used to transmit 0's and 1's, respectively. The resulting signal is a binary frequency shift keying (BFSK) signal. It can be easily demodulated with a non-coherent receiver. If phase coherence is maintained at the transmitter, it can also be demodulated (with a smaller probability of error) using a coherent receiver. All three methods, amplitude shift keying (ASK), phase shift keying (PSK), and frequency shift keying (FSK) can be extended in a straightforward manner from the binary to the M-ary case by using M amplitudes, M phases, or M carrier frequencies, respectively.

1.1 Amplitude Shift Keying

The name "Amplitude Shift Keying" (ASK) refers to a digital bandpass modulation method, whereby the amplitude of a carrier frequency takes on different discrete values (at the sampling time instants), depending on the transmitted DT sequence a_n. The blockdiagram of a coherent ASK communication system looks as follows.



At the transmitter a PAM signal

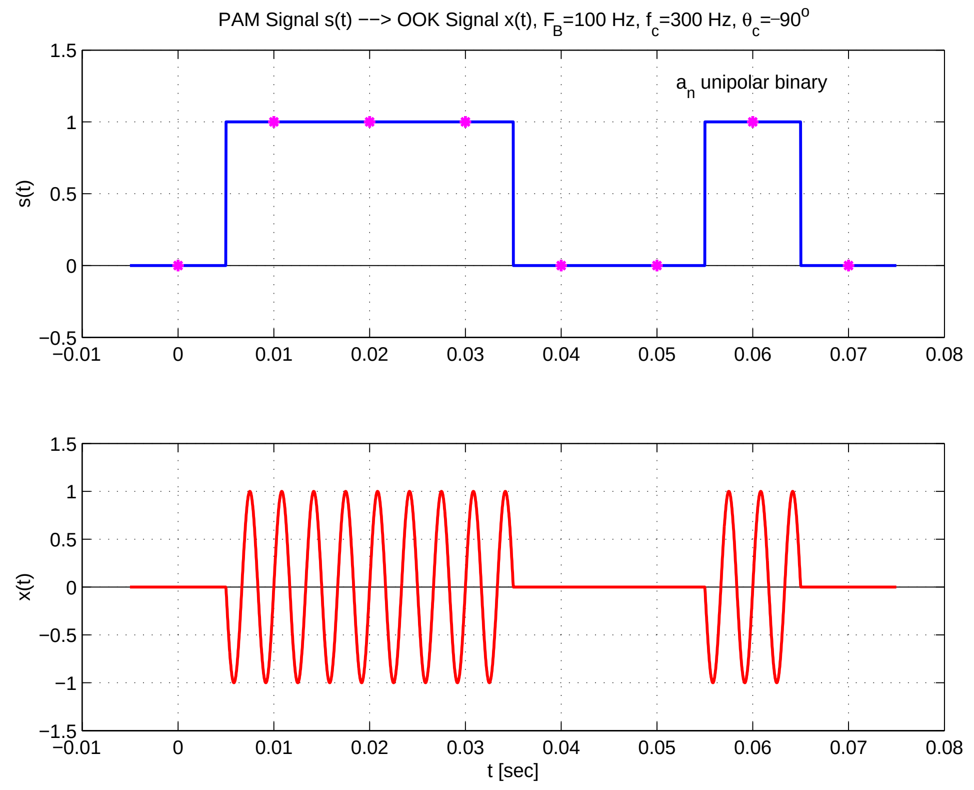
$$s(t) = \sum_{n=-\infty}^{\infty} a_n \, p(t - nT_B) \,,$$

with baud rate $F_B = 1/T_B$ and pulse $p(t)$ is generated from the (digital) DT sequence a_n. The CT signal $s(t)$ then modulates a carrier with frequency f_c and phase θ_c, so that the transmitted signal is

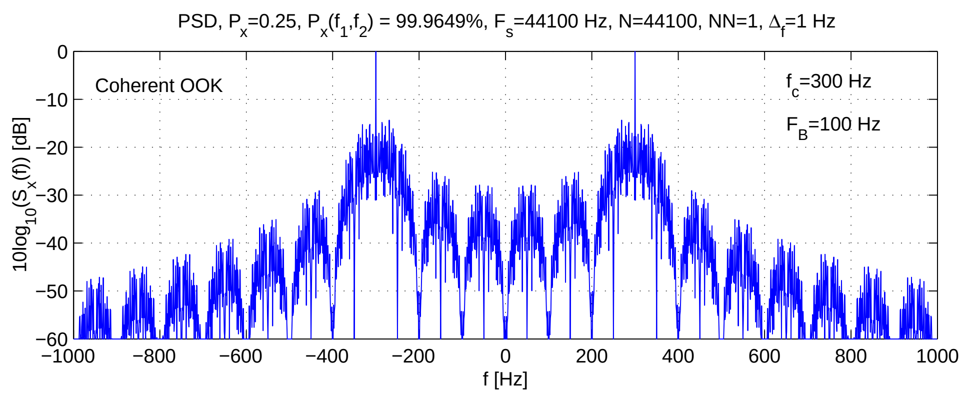
$$x(t) = A_c \sum_{n=-\infty}^{\infty} a_n \, p(t - nT_B) \, \cos(2\pi f_c t + \theta_c) \,.$$

At the receiver the signal $r(t)$ is demodulated using a local oscillator that replicates the carrier signal $\cos(2\pi f_c t + \theta_c)$ with exact frequency and phase synchronization. After that, a regular baseband PAM receiver with matched filter (MF), matched to $p(t) * h_{CL}(t)$, where $h_{CL}(t)$ is the lowpass-equivalent channel response (i.e., $h_C(t)$ shifted down in frequency by f_c), is used.

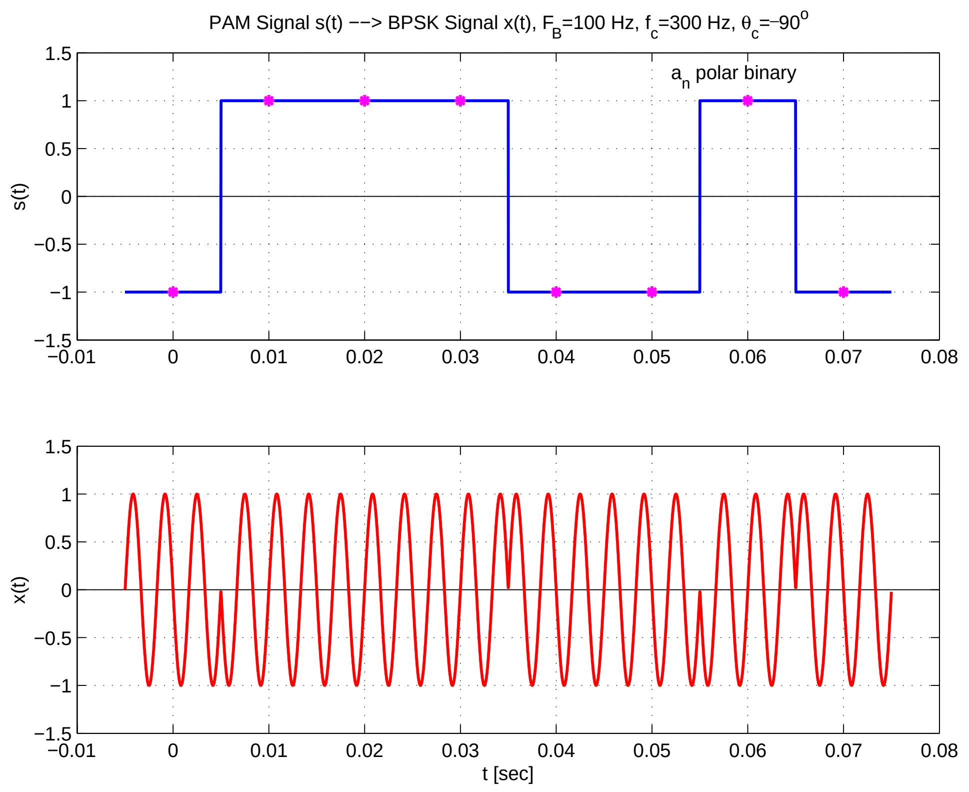
The simplest case of ASK is **on-off keying (OOK)**, which is obtained when a_n is a unipolar binary sequence (i.e., $a_n \in \{0, 1\}$). The following two graphs show $s(t)$ and $x(t)$ for $\{a_n\} = \{0, 1, 1, 1, 0, 0, 1, 0\}$, $F_B = 100$ Hz, rectangular $p(t)$, $f_c = 300$ Hz, and $\theta_c = -90°$.



The PSD of a coherent OOK signal (i.e., an OOK signal for which θ_c does not change over time) with rectangular $p(t)$, $f_c = 300$ Hz, and $F_B = 100$ Hz is shown in the next graph.



If a_n is a polar binary sequence, e.g., $\{a_n\} = \{-1,+1,+1,+1,-1,-1,+1,-1\}$, then $s(t)$ and $x(t)$ look as follows.

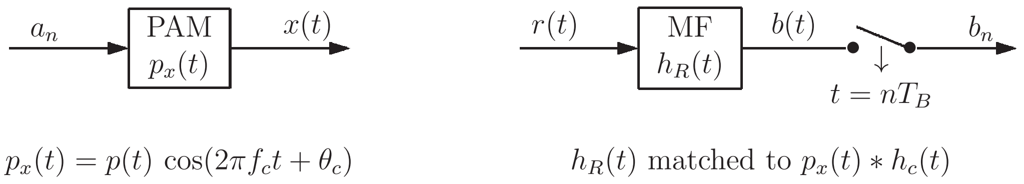


Assuming a rectangular $p(t)$, the carrier is multiplied by either $+1$ or -1, depending on the data to be transmitted. This leads to phase changes by 180° and because of it this version of ASK is usually called **binary phase shift keying (BPSK)**. Note that for BPSK it is crucial that the transmitter and receiver are phase-synchronized, i.e., BPSK requires a coherent receiver.

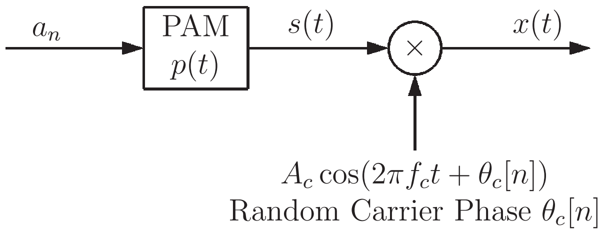
Rather than generating a baseband PAM signal first and then using amplitude modulation to make it a bandpass signal at f_c, it is possible to directly generate a bandpass PAM signal using a pulse $p_x(t)$ that is already shifted to f_c in the frequency domain. Mathematically, this can be expressed as

$$x(t) = \sum_{n=-\infty}^{\infty} a_n \, p_x(t - nT_B) \, , \qquad \text{where} \quad p_x(t) = p(t) \, \cos(2\pi f_c t + \theta_c) \, .$$

At the receiver one can then use a MF that is directly matched to $p_x(t) * h_c(t)$ where $h_c(t)$ is the impulse response of the channel. The blockdiagram of an ASK transmission system that uses $p_x(t)$ directly at f_c is shown below.


a_n
PAM
$p_x(t)$
$x(t)$
$p_x(t) = p(t)\,\cos(2\pi f_c t + \theta_c)$
$r(t)$
MF
$h_R(t)$
$b(t)$
$t = nT_B$
b_n
$h_R(t)$ matched to $p_x(t) * h_c(t)$

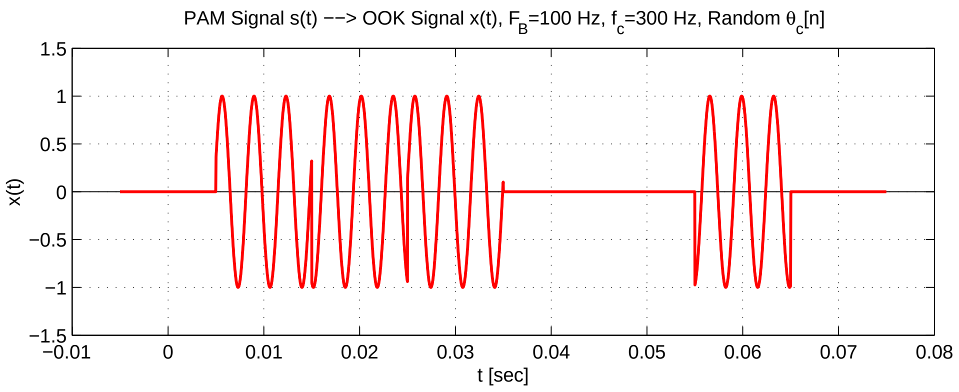
The simplest method to achieve wireless data transmission is to turn an oscillator with frequency f_c on or off, depending on whether a one or a zero is transmitted. Using the morse code for on-off keying, this is how Marconi demonstrated in 1901 that transatlantic wireless communication was possible. What is likely to happen, however, if the oscillator at f_c is turned on and off, is that the carrier phase θ_c takes on random values between 0 and 2π. A model for this is shown in the following blockdiagram.


a_n
PAM
$p(t)$
$s(t)$
$\times$
$x(t)$
$A_c \cos(2\pi f_c t + \theta_c[n])$
Random Carrier Phase $\theta_c[n]$

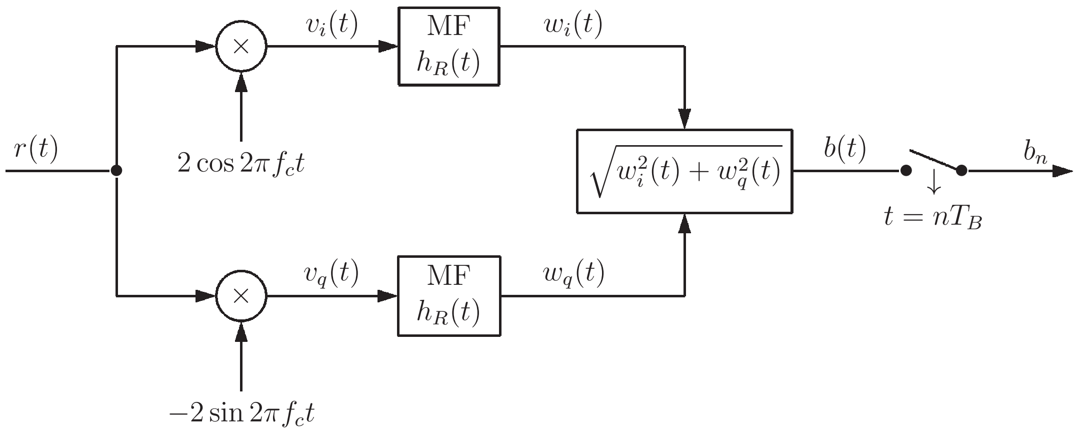
Here it is assumed that for each new symbol a_n a random carrier phase value $\theta_c[n]$ is chosen (even if the oscillator is not turned off between two consecutive ones). Thus

$$\begin{aligned} x(t) &= A_c \sum_{n=-\infty}^{\infty} a_n \, p(t - nT_B) \, \cos(2\pi f_c t + \theta_c[n]) \\ &= A_c \Big[\sum_{n=-\infty}^{\infty} a_n \cos\theta_c[n] \, p(t{-}nT_B) \, \cos 2\pi f_c t - \sum_{n=-\infty}^{\infty} a_n \sin\theta_c[n] \, p(t{-}nT_B) \, \sin 2\pi f_c t \Big] \, , \end{aligned}$$

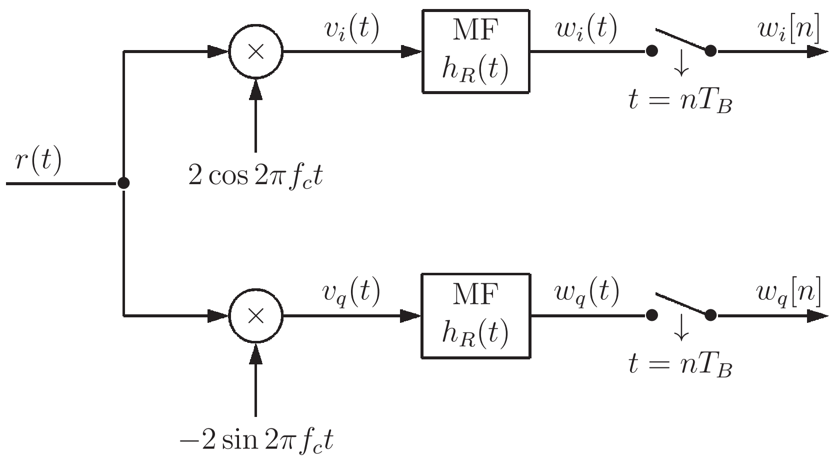
where $\theta_c[n] \in [0, 2\pi)$. Because the transmitted carrier phase of such a signal is not constant, the resulting ASK signal will be called **non-coherent ASK**. An example when $\{a_n\} = \{0, 1, 1, 1, 0, 0, 1, 0\}$, $F_B = 100$ baud, and $f_c = 300$ Hz is shown in the next graph.



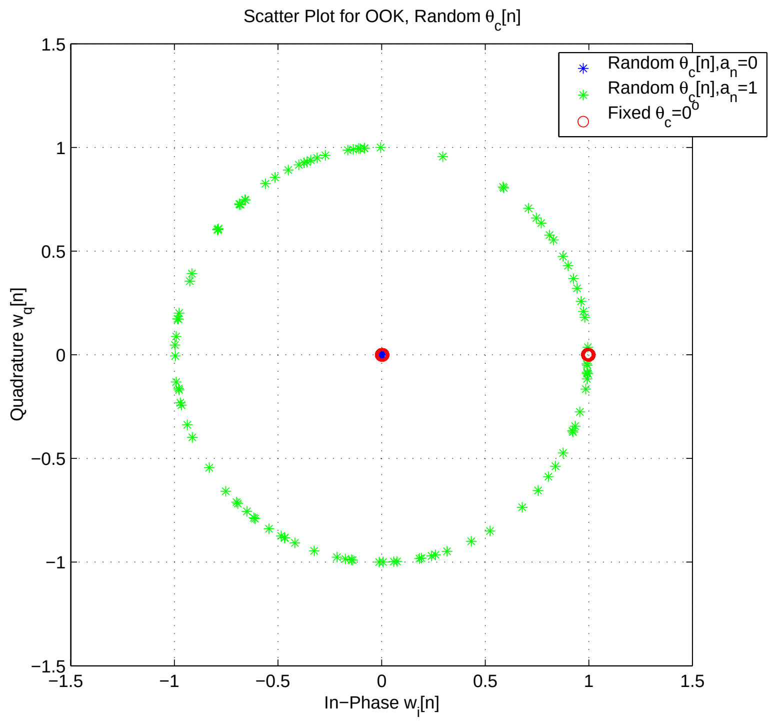
To demodulate a signal with (unknown) random phase, a **matched filter envelope detector (MFED)**, such as the one shown in the following blockdiagram, can be used.



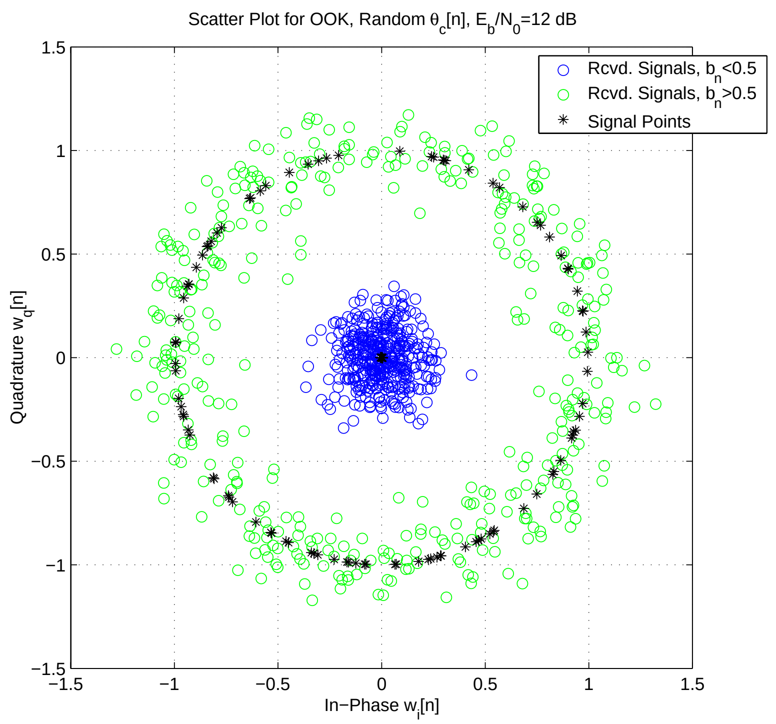
In the process of computing $b(t) = \sqrt{w_i^2(t) + w_q^2(t)}$, the dependence on the carrier phase (and in fact the dependence on the exact carrier frequency f_c) drops out. The interesting observation to be made is that, because of the orthogonality of $\cos 2\pi f_c t$ and $\sin 2\pi f_c t$, the in-phase component $w_i(t)$ and the quadrature component $w_q(t)$ span a 2-dimensional space. Upon sampling at times $t = nT_B$, as shown in the blockdiagram below, this becomes a **2-dimensional signal space** spanned by $w_i[n]$ (real part in complex plane) and $w_q[n]$ (imaginary part in complex plane).



Thus, coherent ASK uses a one-dimensional signal space and noncoherent ASK uses a two-dimensional signal space. This is shown for OOK in the following figure which is called a **scatter plot**.



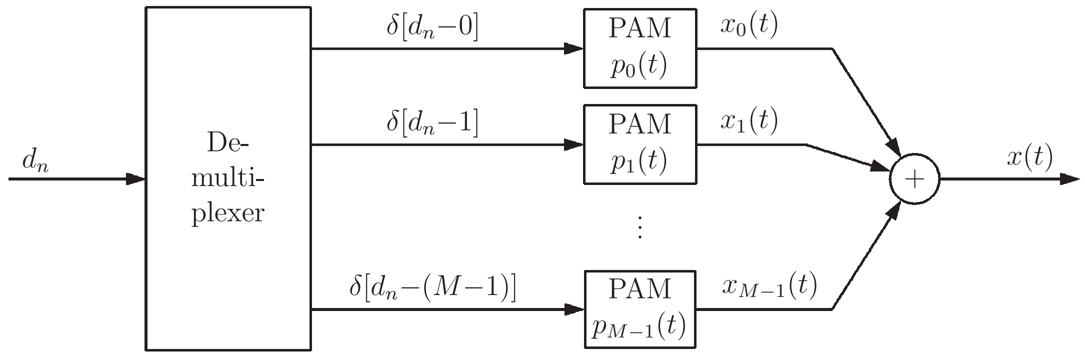
Note that for noncoherent ASK the signal "points" are actually points on concentric circles. If the received signal is noisy, then the noise manifests itself as deviations of these points from the concentric circles, as shown for OOK with SNR $E_b/\mathcal{N}_0 = 12$ dB in the scatter plot below.



The black stars are the signal points of the noncoherent ASK signal at the transmitter. The blue and green circles are the received noisy signal points with blue indicating a decision for a received 0 and green indicating a decision for a received 1. As the SNR decreases, the received signal points deviate more and more from their nominal positions at the origin or on the circle with radius 1, and the probability of symbol error, $P_s(\mathcal{E})$, approaches 0.5.

1.2 Frequency Shift Keying

Rather than using a digital DT sequence to change the amplitude of a carrier, the frequency of the carrier can be changed by the data sequence to be transmitted. The resulting digital bandpass signaling method is called **frequency shift keying (FSK)**. One way to implement M-ary FSK is shown in the following block diagram.



An M-ary DT sequence d_n, with $d_n \in \{0, 1, \ldots, M-1\}$, is first demultiplexed into M binary sequences, the first one having 1's in all positions where d_n is zero and 0's everywhere else, the second one having 1's in all positions where d_n is one and 0's everywhere else, etc. The M sequences are then converted to OOK signals of the form

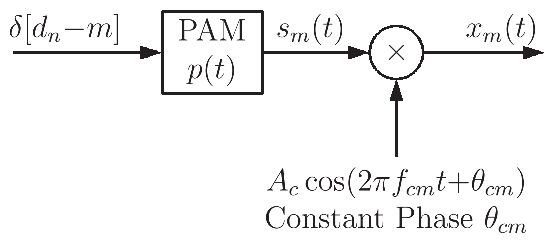
$$x_m(t) = \sum_{n=-\infty}^{\infty} \delta[d_n - m]\, p_m(t - nT_B) \,, \quad m = 0, 1, \ldots, M{-}1 \,, \quad \delta[k] = \begin{cases} 1 \,, & k = 0 \,, \\ 0 \,, & \text{otherwise} \,, \end{cases}$$

and added up to produce the FSK signal $x(t)$ as

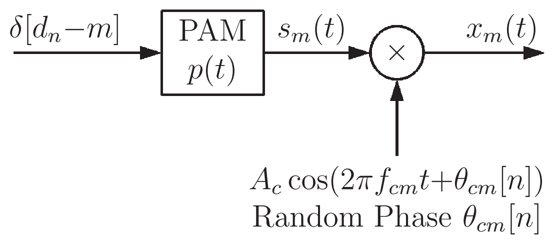
$$x(t) = \sum_{m=0}^{M-1} x_m(t) = \sum_{m=0}^{M-1} \sum_{n=-\infty}^{\infty} \delta[d_n{-}m]\, p_m(t - nT_B) \,.$$

Depending on whether the individual OOK signals are coherent or non-coherent, one can distinguish between **coherent FSK** and **non-coherent FSK**. The blockdiagrams below show how to implement the individual bandpass PAM functions $p_m(t)$, $m = 0, 1, \ldots, M{-}1$, in each case.

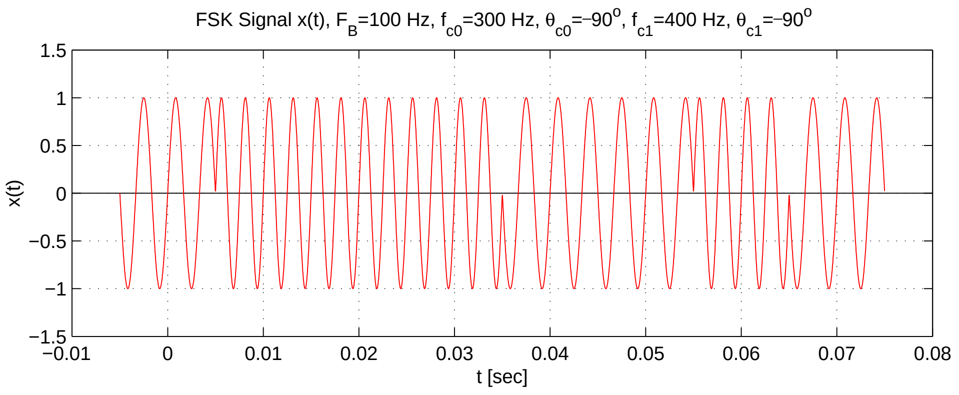
PAM $p_m(t)$ for Coherent FSK

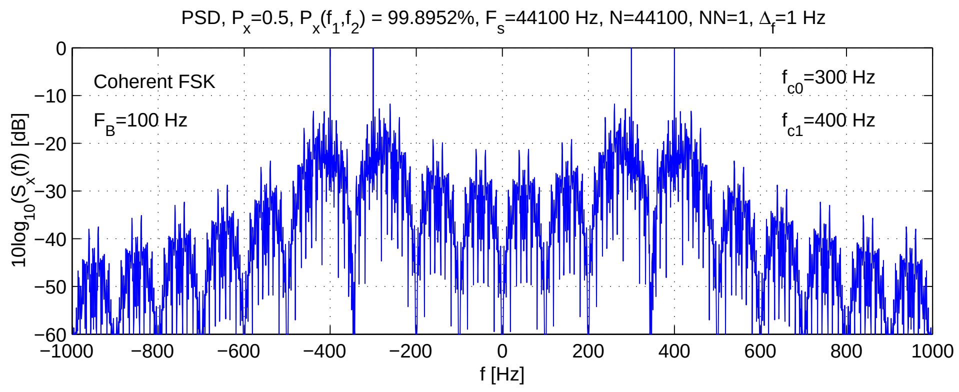

$\delta[d_n{-}m]$
PAM $p(t)$
$s_m(t)$
$\times$
$x_m(t)$
$A_c \cos(2\pi f_{cm} t{+}\theta_{cm})$
Constant Phase θ_{cm}

PAM $p_m(t)$ for Non-Coherent FSK

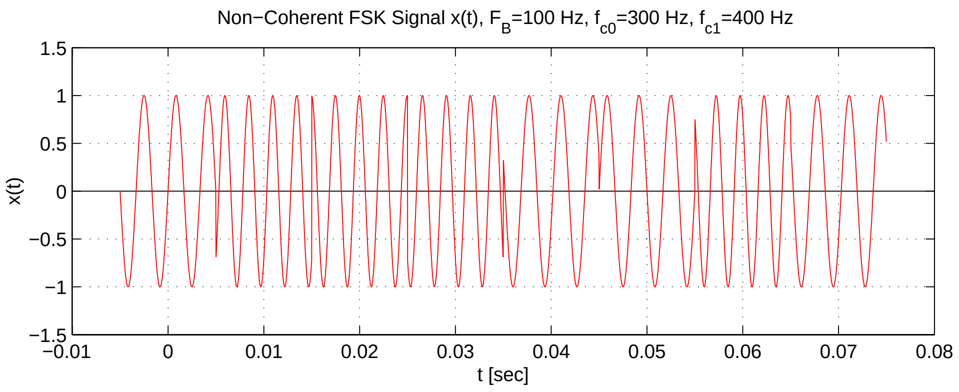

$\delta[d_n{-}m]$
PAM $p(t)$
$s_m(t)$
$\times$
$x_m(t)$
$A_c \cos(2\pi f_{cm} t{+}\theta_{cm}[n])$
Random Phase $\theta_{cm}[n]$

Thus, for coherent FSK $p_m(t) = A_c\, p(t)\, \cos(2\pi f_{cm} t{+}\theta_{cm})$, and for non-coherent FSK $p_m(t) = A_c\, p(t)\, \cos(2\pi f_{cm} t + \theta_{cm}[n])$. Note that, when the FSK signal is made up from the outputs of individual oscillators, $x(t)$ may contain phase jumps at the symbol boundaries, as can be seen in the following graph that shows an example of coherent binary FSK with rectangular $p(t)$, $F_B = 100$ baud, $f_{c1} = 300$ Hz, $\theta_{c1} = -90°$, $f_{c2} = 400$ Hz, $\theta_{c2} = -90°$.

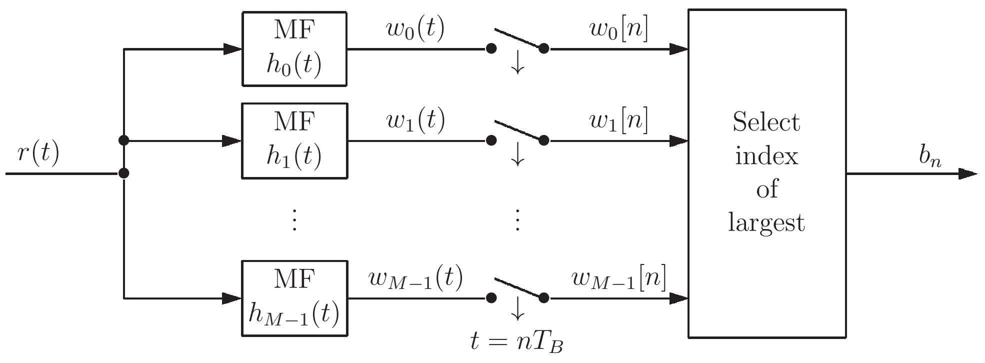




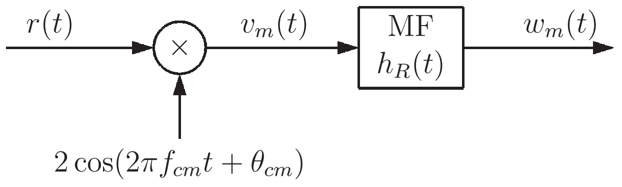
The upper graph shows $x(t)$ in the time domain for $\{d_n\} = \{0, 1, 1, 1, 0, 0, 1, 0\}$, and the lower graph shows the PSD $S_x(f)$ when d_n is a random binary signal with equally likely 0's and 1's. The next graph shows $x(t)$ for the same data and parameters, but using a non-coherent FSK transmitter.



The block diagram below shows the basic structure of a receiver for M-ary FSK.



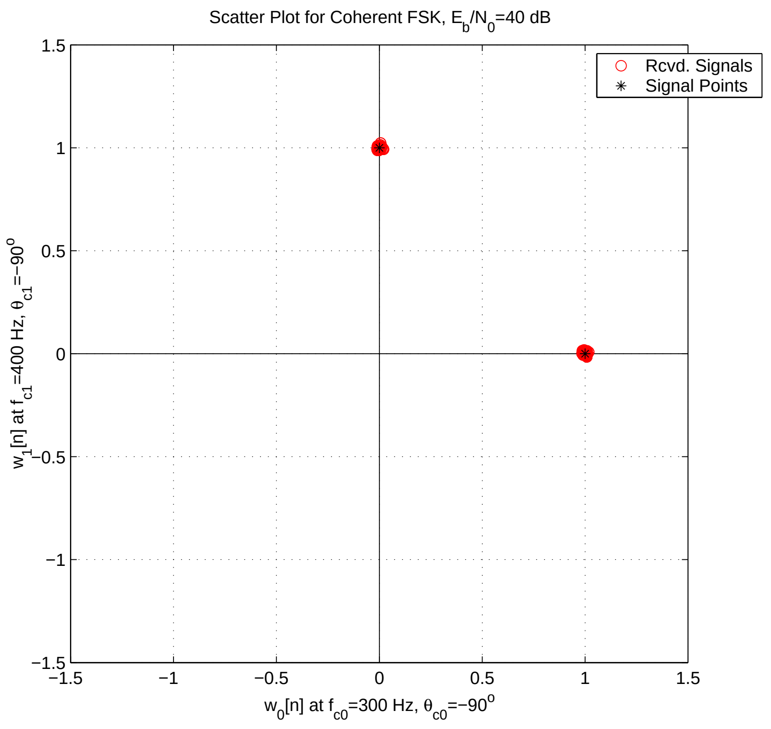
There is a matched filter at each of the M FSK frequencies $f_{c0}, \ldots, f_{cM-1}$. Coherent FSK can be received with coherent matched filters. One way to implement a coherent MF when f_{cm} and θ_{cm} are fixed and known is shown in the following blockdiagram.



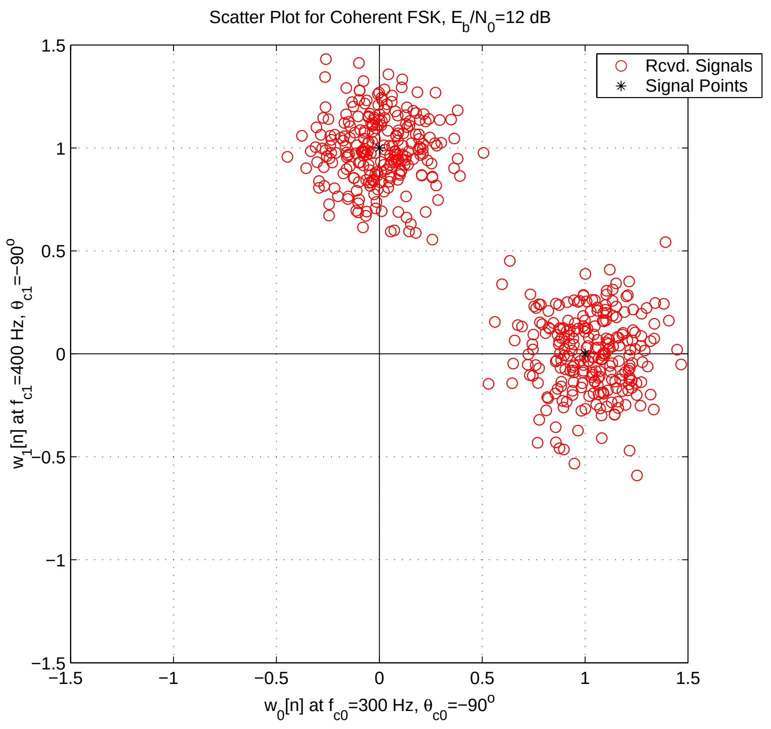
An interesting question is how close together any two adjacent frequencies, say $f_{c\ell}$ and $f_{cm} = f_{c\ell} + \Delta_f$, should be chosen. By making Δ_f small, the total bandwidth used by M-ary FSK can be reduced, but if Δ_f is too small, then the normalized distance between symbols using different frequencies becomes too small and thus the probability of error increases. As a general criterion, one usually tries to make all M symbols in M-ary FSK **orthogonal**, i.e., assuming $p(t)$ is a rectangular pulse of width T_B, one selects Δ_f in $f_{cm} = f_{c\ell} + \Delta_f$ such that

$$\int_{(n-1/2)T_B}^{(n+1/2)T_B} \cos(2\pi f_{c\ell}t + \theta_{c\ell})\, \cos(2\pi f_{cm}t + \theta_{cm})\, dt = 0\ , \qquad \text{for all integers } n.$$

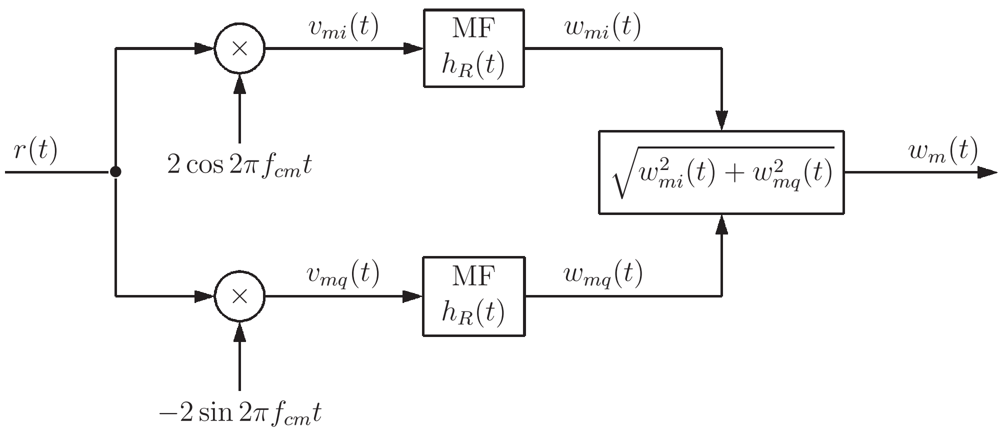
The scatter plot in the following figure shows $w_0[n]$ versus $w_1[n]$ for coherent binary FSK satisfying the orthogonality condition.



For the graph above a FSK signal with high SNR (40 dB) was used so that the nominal signal points (black *) and the received signal points (red o) coincide. The graph below shows how this changes when the SNR is reduced to $E_b/\mathcal{N}_0 = 12$ dB.

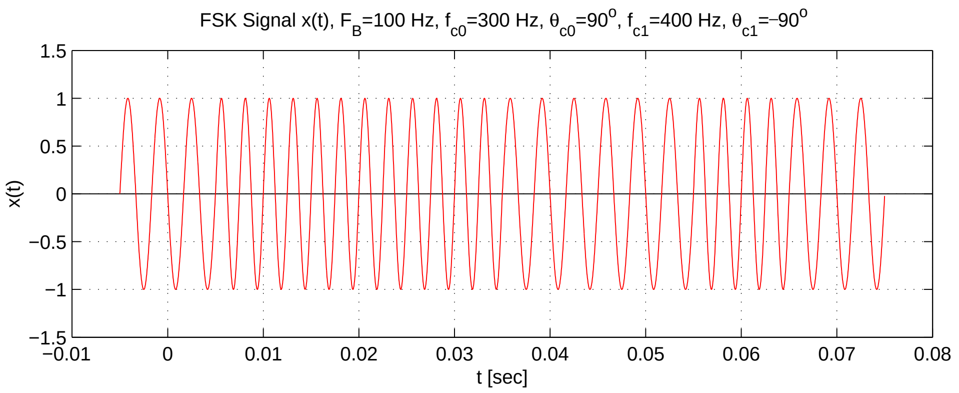


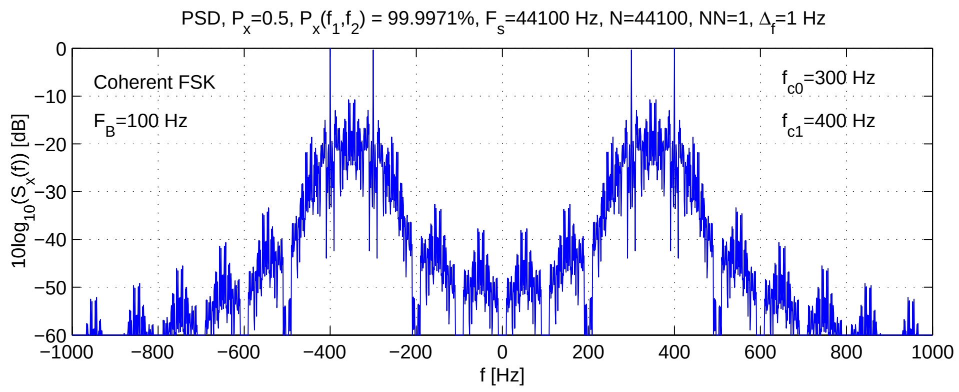
If the carrier phase is random and/or unknown to the receiver, then matched filter envelope detectors (MFED) of the form shown in the blockdiagram below need to be used.



In this case, a scatter plot of binary FSK would have to be drawn in 4-dimensional space. Looking at just $w_0i[n]$ versus $w_0q[n]$, or $w_1i[n]$ versus $w_1q[n]$ yields signal "points" that either lie at the origin or on a circle, centered at the origin.

One more version of FSK that is actually preferred in practice is **continuous phase FSK (CPFSK)**. This can be obtained from coherent FSK by choosing the frequencies f_{cm} and the phases θ_{cm} in such a way that phase jumps are avoided at the symbol boundaries. An example of binary CPFSK and its PSD is shown in the following two graphs.



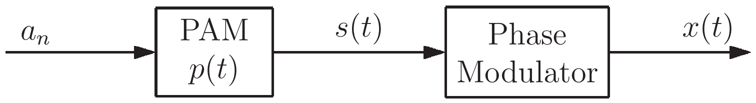


As can be seen in the lower graph, the absence of phase jumps in $x(t)$ yields a PSD $S_x(f)$ that decreases rapidly for frequencies below f_{c0} and above f_{cM-1} (in the M-ary case), which reduces interference between communication systems that use adjacent bands.

1.3 Phase Shift Keying

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 FSK.

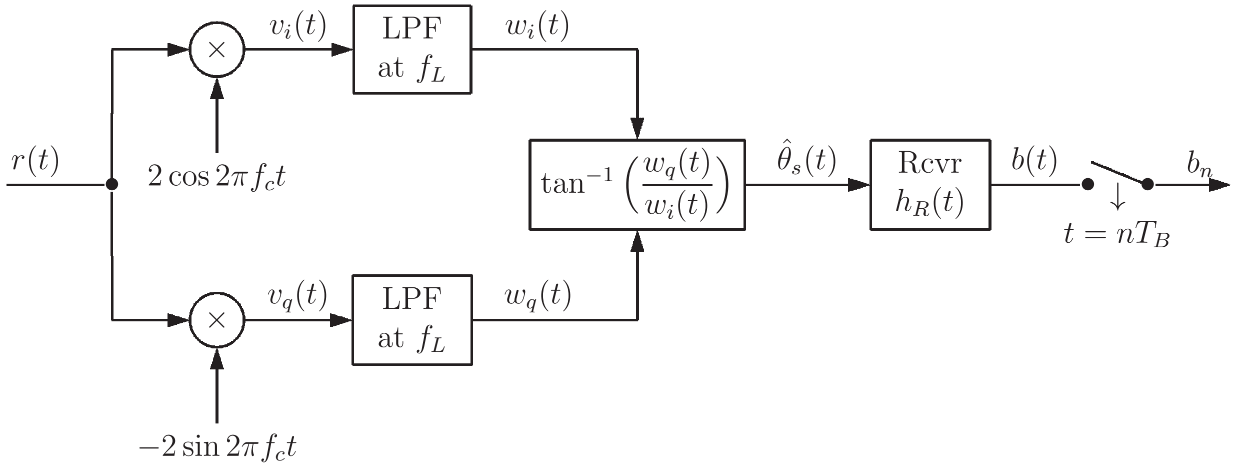
Let a_n denote a DT sequence with baud rate $F_B = 1/T_B$. The following block diagram can be used to first convert this sequence into a CT PAM signal $s(t)$, which is then fed into a phase modulator (PM) to generate a PM signal $x(t)$ at some carrier frequency f_c.



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

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

$p(t)$ is a PAM pulse, and 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(wqt,wit)`, 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

$$\begin{aligned} v_i(t) &= 2\,r(t)\,\cos(2\pi f_c t) = \cos\big(\Delta_\theta\, s(t)\big) + \cos\big(4\pi f_c t + \Delta_\theta\, s(t)\big)\,, \\ v_q(t) &= -2\,r(t)\,\sin(2\pi f_c t) = \sin\big(\Delta_\theta\, s(t)\big) - \sin\big(4\pi f_c t + \Delta_\theta\, s(t)\big)\,. \end{aligned}$$

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

$$\begin{aligned} w_i(t) &= \cos\big(\Delta_\theta\, s(t)\big) \quad \Longrightarrow \quad w_i(nT_B) = w_i[n] = \cos\big(\Delta_\theta\, a_n\big)\,, \\ w_q(t) &= \sin\big(\Delta_\theta\, s(t)\big) \quad \Longrightarrow \quad w_q(nT_B) = w_q[n] = \sin\big(\Delta_\theta\, a_n\big)\,, \end{aligned}$$

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

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

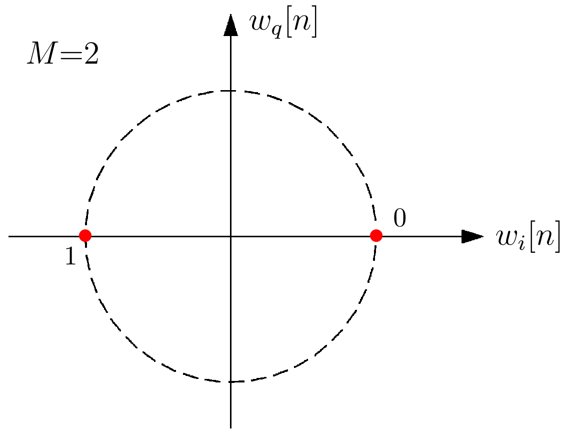
The quantities $w_i[n]$ and $w_q[n]$ can be interpreted as the real and imaginary parts, respectively, of a point on the unit circle at angle $\Delta_\theta\, a_n$ in the complex plane. If a_n can only take on M discrete values, e.g., $a_n \in \{0, 1, \ldots, M-1\}$, then the resulting M points can be 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

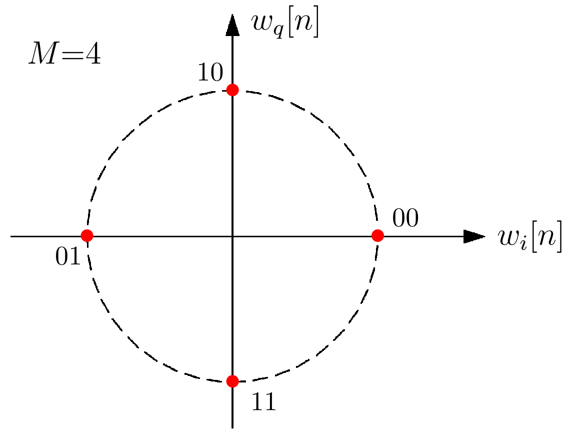
$$a_n \in \{0, 1, \ldots, M-1\}\,, \qquad \Delta_\theta = \frac{2\pi}{M}\,,$$

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 $M = 4$.

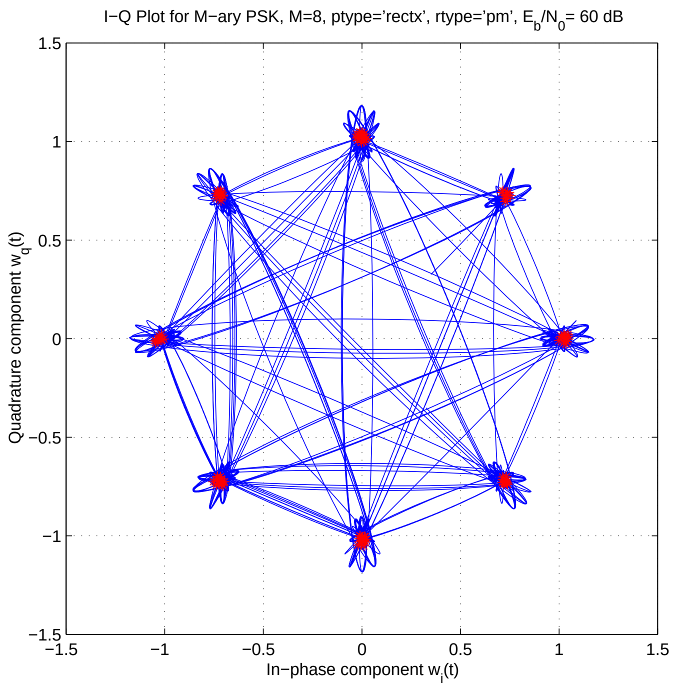




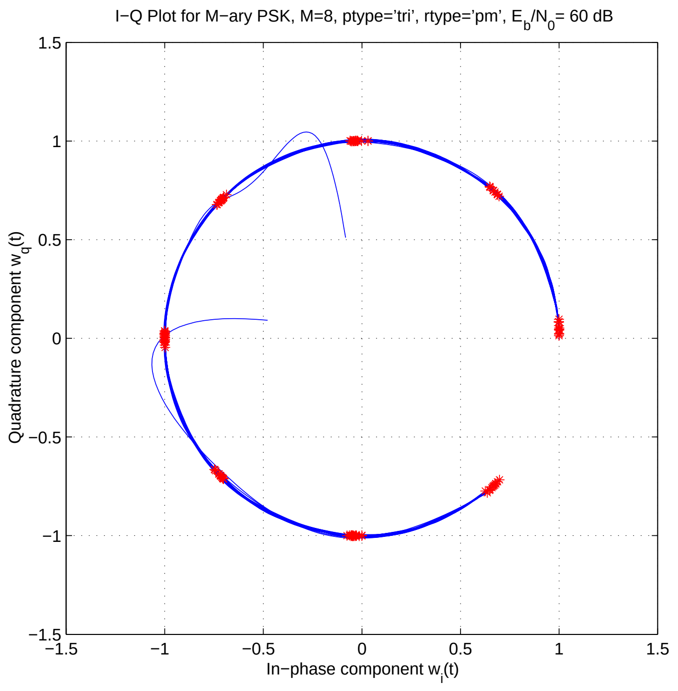
The graph which shows the locations of the M signal points and their geometrical relationships with respect to each other is called a **signal constellation**. For **M-ary phase shift keying (PSK)**, the signal constellation consists of M points spread equally around a (unit) circle in a 2-dimensional 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 $M = 2^1 \ldots 2^4$ 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	$\{0, \ldots, 6, 7, 8, 9, \ldots, 15\}$	$\{0000, \ldots, 0110, 1110, 0001, 1001, \ldots, 1111\}$

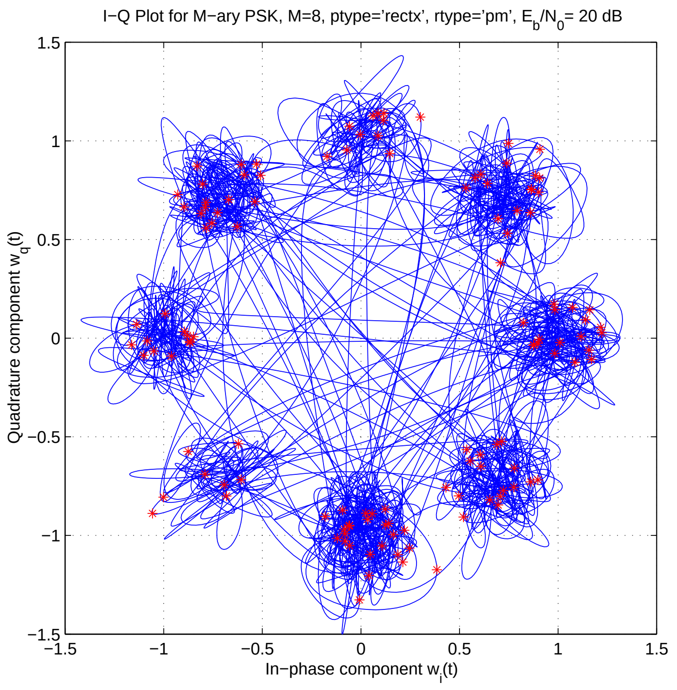
The signal constellations given so far only show the sampled in-phase and quadrature components $w_i[n]$ and $w_q[n]$ at the outputs of the lowpass filters in the PM receiver. A more general **I-Q plot**, which in addition also shows the transitions between the signal points, is obtained by also plotting $w_q(t)$ 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.



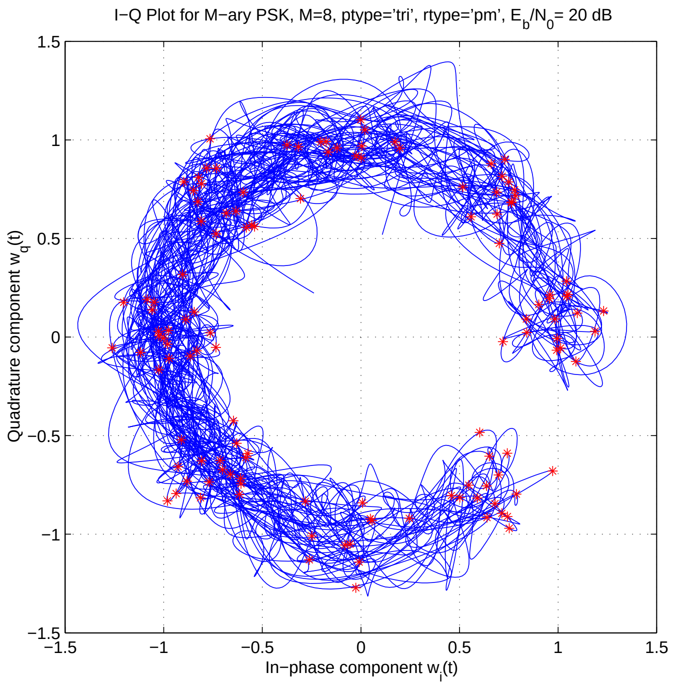
If $p(t)$ is rectangular as in the above plot, then the transitions between signal points are essentially straight lines. If $p(t)$ is triangular as shown in the next plot, then the phase changes occur more gradual and the transitions between signal points all occur on the circle on which the signal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



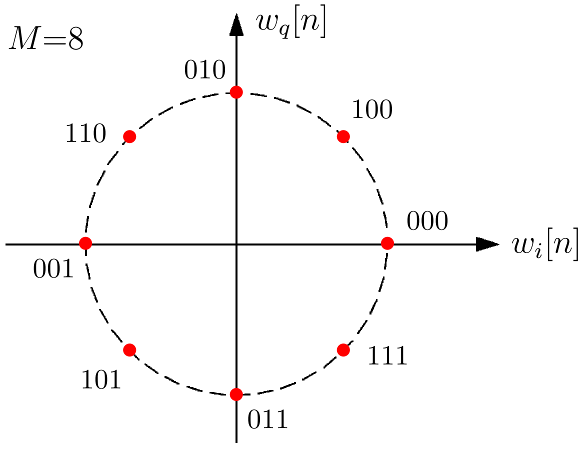
In a real communication system the received signal is usually noisy. In this case, the signal points (and the transitions between them) are scattered around their nominal locations. An example for $M = 8$, rectangular $p(t)$, and a signal-to-noise ratio (SNR) of $E_b/\mathcal{N}_0 = 20$ dB is shown in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

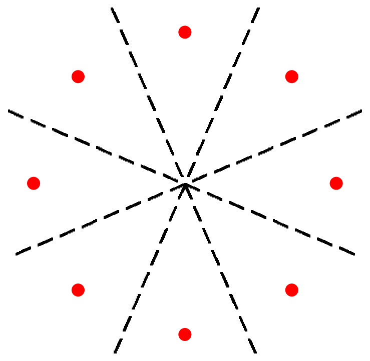


If $p(t)$ is triangular instead of rectangular, this changes as shown next.



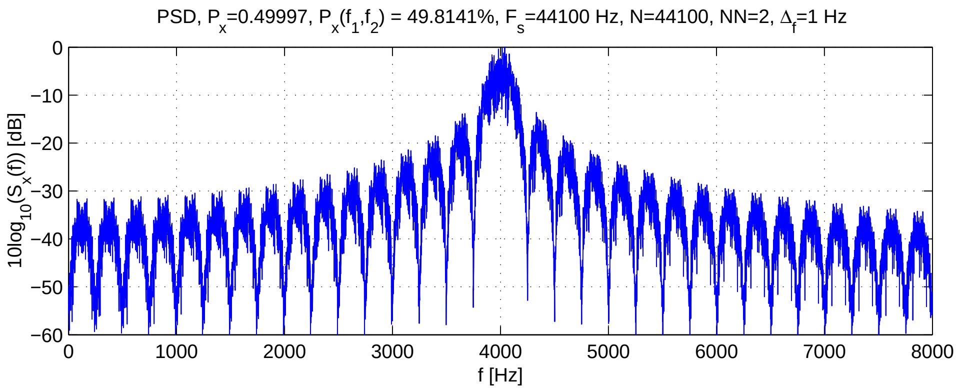
Assuming that the in-phase and quadrature noise components have equal power, and all M signal points are equally likely, the decision rule at the receiver is to assume that the most likely transmitted signal is the one that is closest in (Euclidean) distance to the received signal point. This leads to 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.



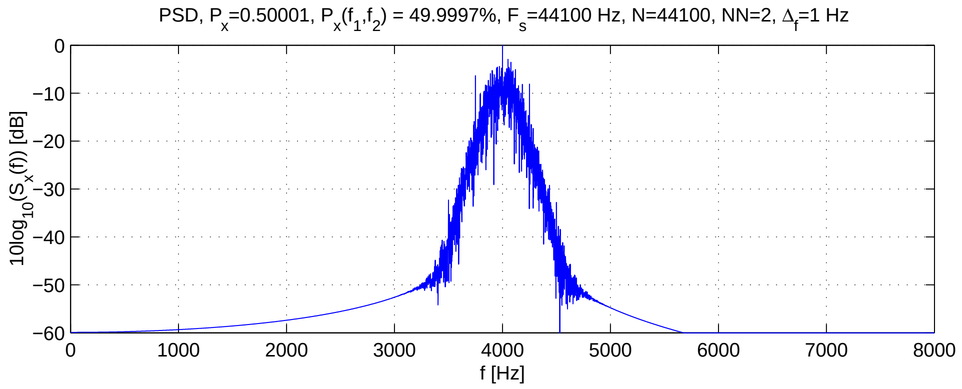


As the SNR decreases, some received signal points will eventually cross the decision region boundaries, which in turn leads to symbol and bit errors at the receiver output.

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, and 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

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

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 (matched to $p(t)$) 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.

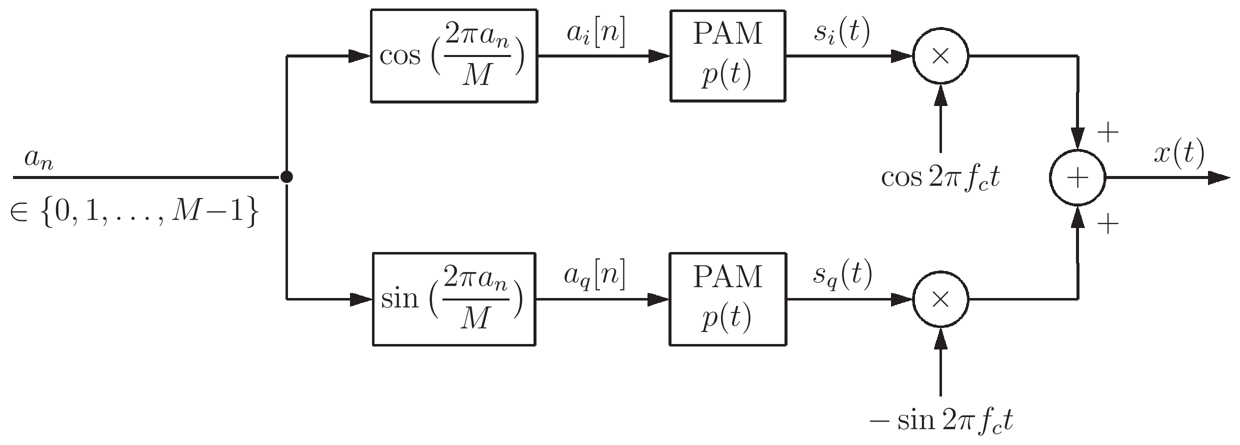
Another approach that is very often used in practice is **QAM-based PSK** which splits the DT sequence a_n up into an in-phase component $a_i[n]$ and a quadrature component $a_q[n]$ by setting

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

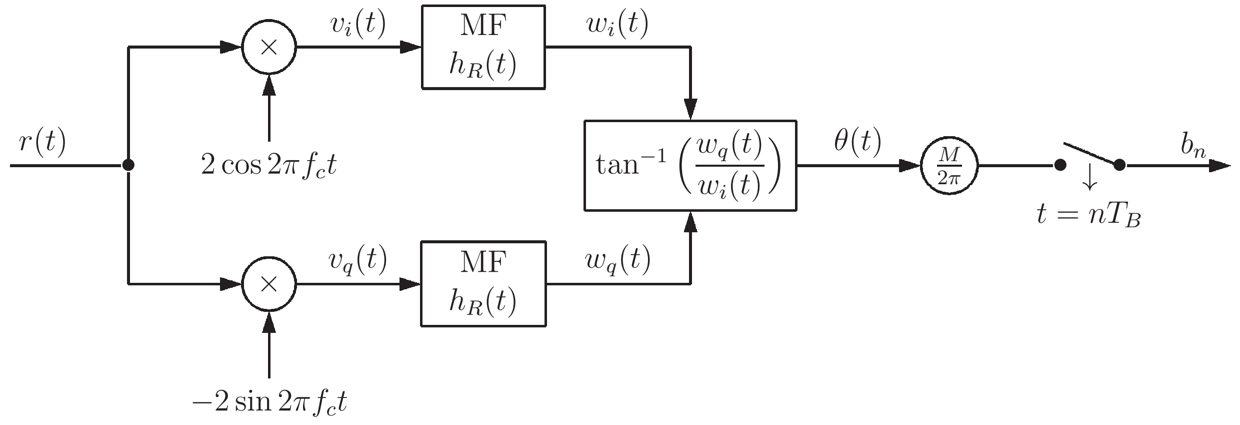
Each of these two signals is then converted separately into a PAM signal, using the same pulse $p(t)$ in most cases. The two PAM signals are then fed into a quadrature amplitude modulator (QAM) to form a PSK signal $x(t)$ of the form

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

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



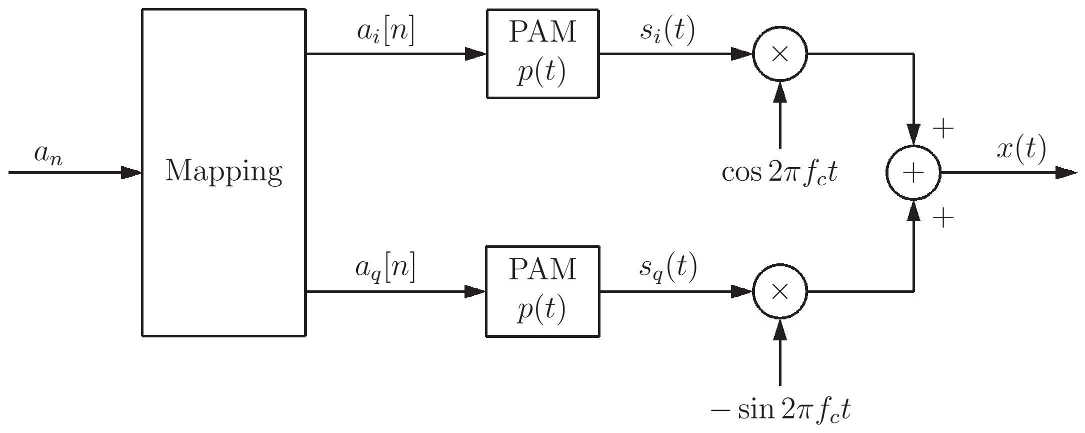
The advantage of using separate in-phase and quadrature components is that now a QAM receiver like the one shown next can be used, where the LPF's are replaced by matched filters (MF), matched to $p(t)$ (assuming an ideal transmission channel).



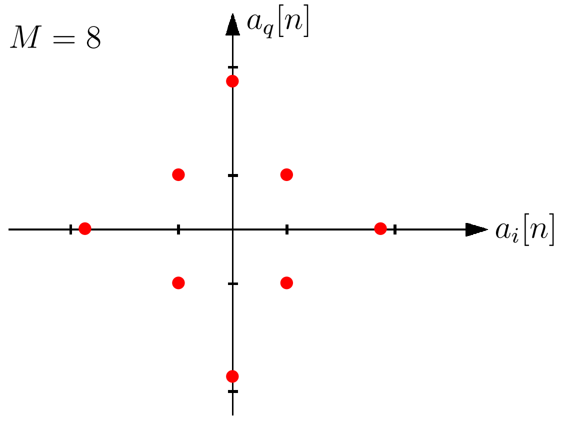
To make scatter plots and *I*-*Q* plots from this receiver, just simply plot $w_q(nT_B)$ versus $w_i(nT_B)$ or $w_q(t)$ versus $w_i(t)$, in the same way as this was done before with the PM receiver.

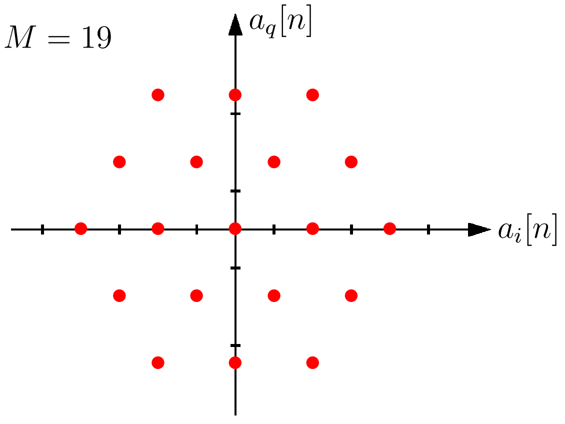
1.4 Hybrid Amplitude/Phase Shift Keying

In practice the use of M-PSK is usually limited to $M \leq 8$ distinct phases. For larger M a combination of ASK and PSK, termed hybrid M-ary amplitude/phase shift keying (APSK) in this lab description, is generally used. One convenient way to implement hybrid APSK is to split up the digital data into an in-phase and a quadrature data sequence, use 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) modulator. Thus, for APSK both the amplitude and the phase of the carrier are 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 to waveforms using PAM with a pulse $p(t)$ and combined into a QAM signal $x(t)$ at carrier frequency f_c (and carrier phase θ_c) using a regular QAM modulator.

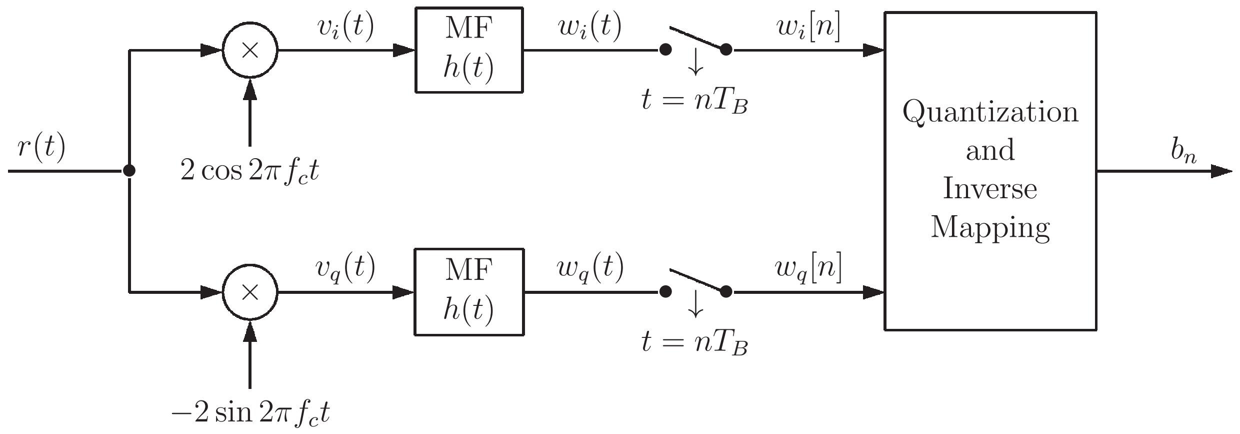


The signal constellation is obtained by plotting $a_q[n]$ versus $a_i[n]$. The geometry of the signal constellation is determined by the mapping from a_n to $a_i[n]$ and $a_q[n]$. In principle, any 2-dimensional constellation, such as the two examples for $M = 8$ and $M = 19$ shown below, can be used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

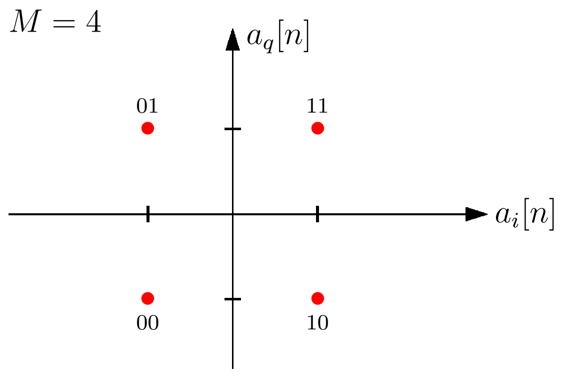


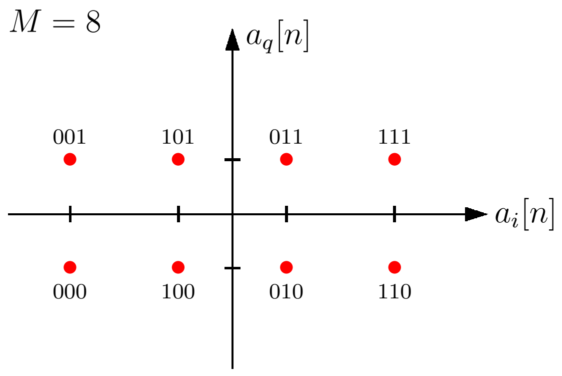


However, since in general a receiver as shown in the following blockdiagram is used, rectangular or **cartesian signal constellations (CSC)** are preferred because they lead to easier implementations of the receiver.

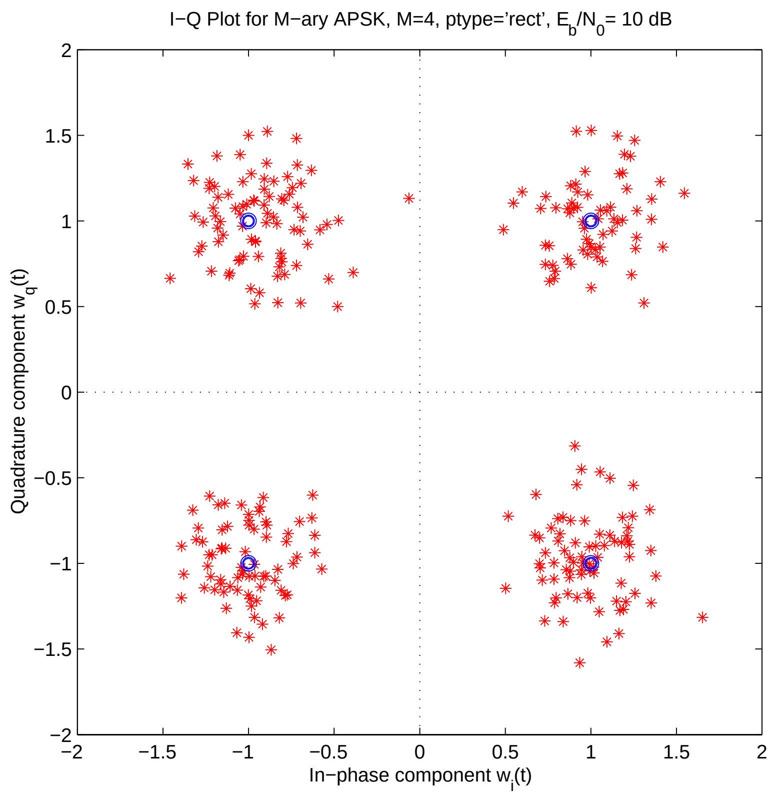


CSC constellations for $M = 4$ and $M = 8$ are shown in the next two figures. For $M = 4$ it is quite natural to use one bit per symbol for $a_i[n]$ and the other one for $a_q[n]$. For $M = 8$ it is less clear how three bits should be split up between the in-phase and quadrature data sequences. A somewhat 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.



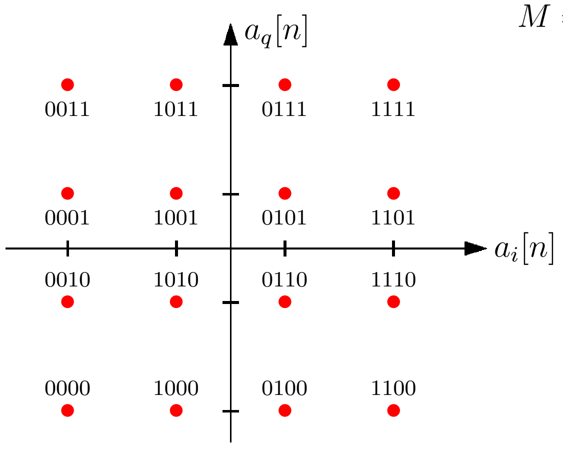


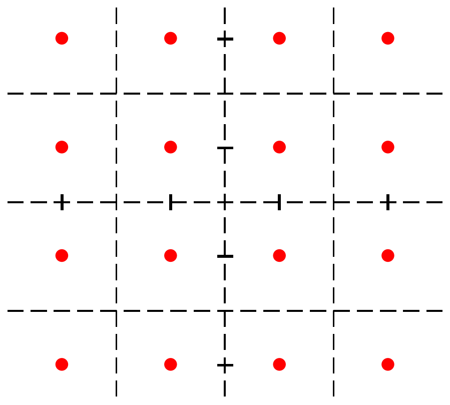
Apart from the usual problem of synchronizing the receiver with the transmitter, the main problem of the receiver is to associate the received noisy sample pairs $(w_i[n], w_q[n])$ with the most likely transmitted signal point $(a_i[n], a_q[n])$. A maximum liklelihood (ML) receiver measures the Euclidean 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.

$M = 16$





1.5 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 $d_n \in \{0, 1, \ldots, M-1\}$

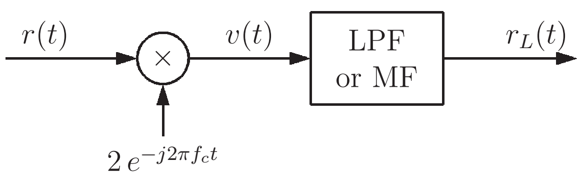
$$d_i[n] = \cos\Big(\frac{2\pi d_n}{M}\Big) \,, \qquad \text{and} \qquad d_q[n] = \sin\Big(\frac{2\pi d_n}{M}\Big) \,,$$

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

$$v(t) = 2\,r(t)\,e^{-j2\pi f_c t} = \left[s_L(t)\,e^{j(2\pi f_c t+\theta_e(t))} + s_L^*(t)\,e^{-j(2\pi f_c t+\theta_e(t))}\right] e^{-j2\pi f_c t}$$
$$= s_L(t)\,e^{j\theta_e(t)} + s_L^*(t)\,e^{-j(4\pi f_c t+\theta_e(t))}\,,$$

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,

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

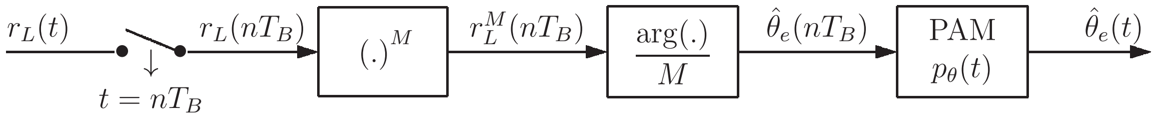
If there is no ISI and $p(nT_B)=\delta_n$, then the samples at $t=nT_B$ are in both cases

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

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

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

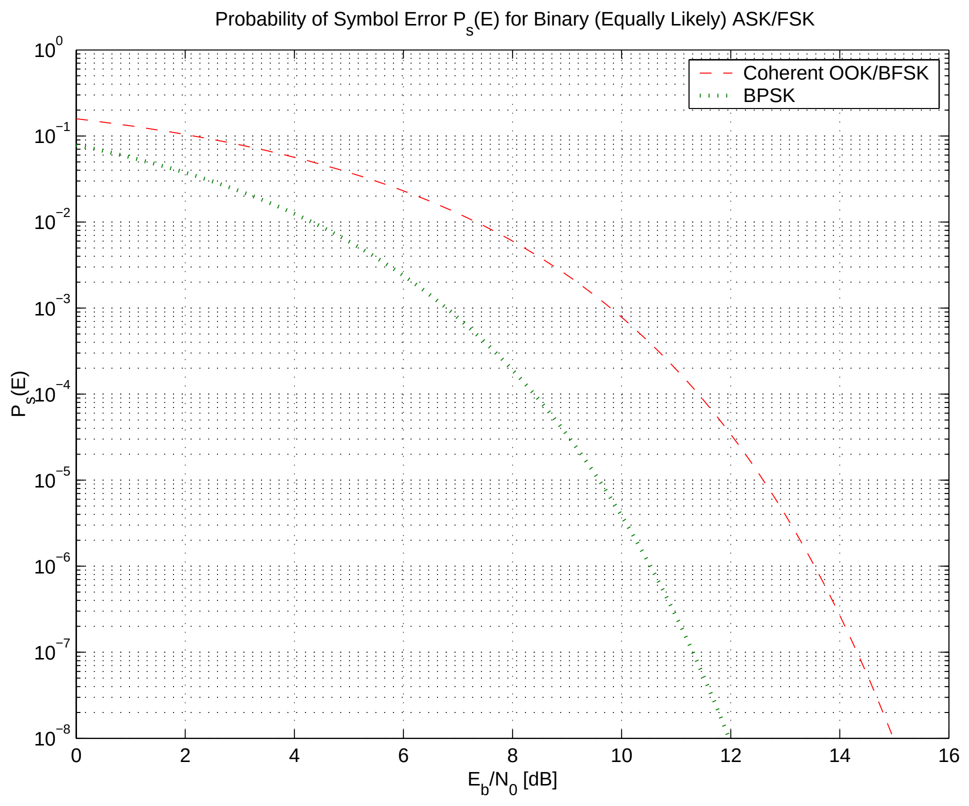
Taking the magnitude squared yields

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

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

1.6 Signal Space, Probability of Error

The graph below shows the probability $P_s(\mathcal{E})$ of error for OOK, BPSK and BFSK when coherent receivers are used.

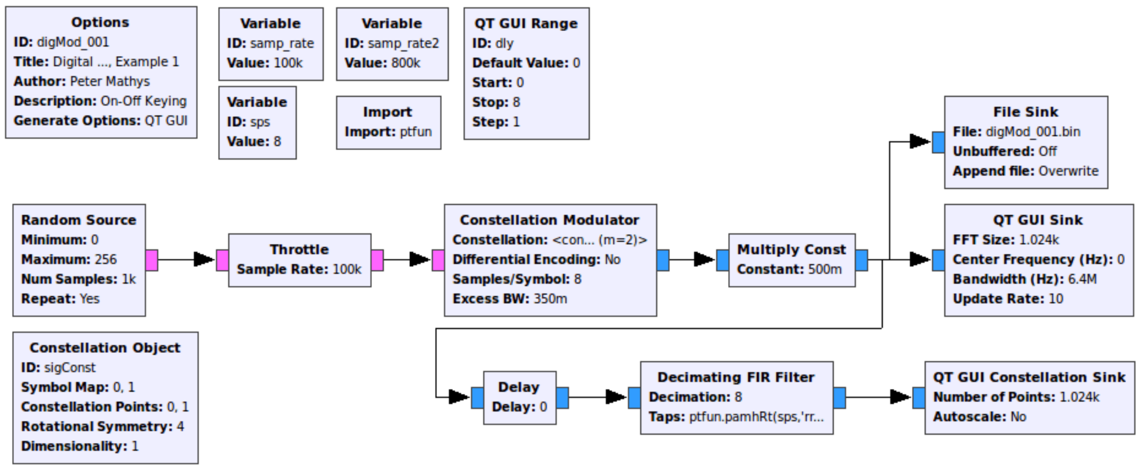


In general a non-coherent receiver will have a larger $P_s(\mathcal{E})$ for a given $E_b/\mathcal{N}_0$. The reason for this is that a coherent OOK/ASK receiver at some carrier frequency f_c (or f_{ci} for the i-th FSK frequency) only needs to look at the in-phase component of the demodulation, whereas a non-coherent receiver needs to look at both the in-phase and the quadrature components. Thus, a coherent receiver only picks up noise from the in-phase channel, whereas a non-coherent receiver picks up noise from both the in-phase and the quadrature channels.

1.7 Digital Modulation in GNU Radio

The GNU Radio Companion (version 3.7.9.1) comes with a number of specific digital modulator blocks, e.g., for CPFSK (continuous phase FSK), for DPSK (differential PSK), GFSK (Gaussian FSK), and GMSK (Gaussian minimum shift keying). There is also a general Constellation Modulator whose properties can be controlled using a Constellation Object block. This modulator offers more possibilities to experiment with different parameters and constellations and we can use it to generate OOK, BPSK, and QAM signals.

The figure below shows a baseband modulator for OOK.


Options
ID: digMod_001
Title: Digital ..., Example 1
Author: Peter Mathys
Description: On-Off Keying
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 100k

Variable
ID: samp_rate2
Value: 800k

QT GUI Range
ID: dly
Default Value: 0
Start: 0
Stop: 8
Step: 1

Variable
ID: sps
Value: 8

Import
Import: ptfun

File Sink
File: digMod_001.bin
Unbuffered: Off
Append file: Overwrite

Random Source
Minimum: 0
Maximum: 256
Num Samples: 1k
Repeat: Yes

Throttle
Sample Rate: 100k

Constellation Modulator
Constellation: <con... (m=2)>
Differential Encoding: No
Samples/Symbol: 8
Excess BW: 350m

Multiply Const
Constant: 500m

QT GUI Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 6.4M
Update Rate: 10

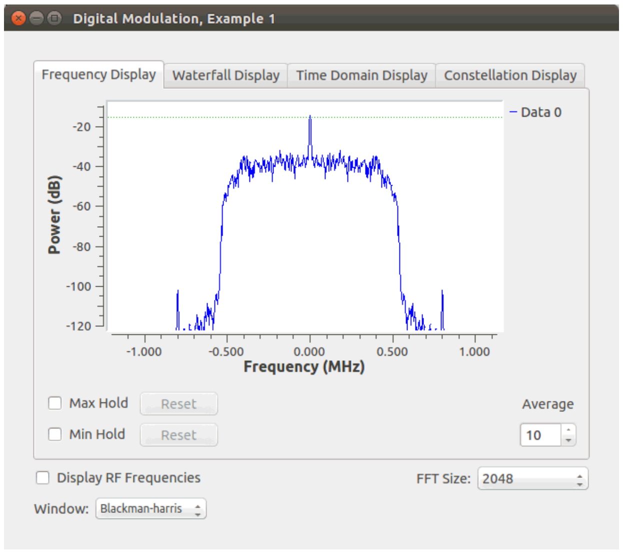
Constellation Object
ID: sigConst
Symbol Map: 0, 1
Constellation Points: 0, 1
Rotational Symmetry: 4
Dimensionality: 1

Delay
Delay: 0

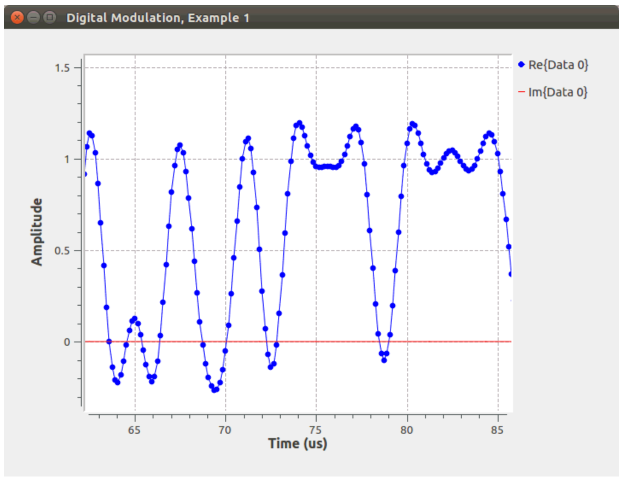
Decimating FIR Filter
Decimation: 8
Taps: ptfun.pamhRt(sps,'rr...

QT GUI Constellation Sink
Number of Points: 1.024k
Autoscale: No

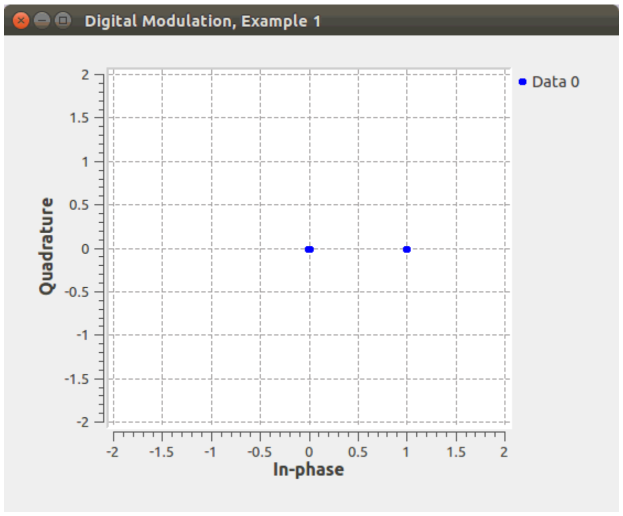
The Random Source block generates bytes with random values in the range from 0 to 255 which are fed to the input of the Constellation Modulator via a Throttle block. With the settings chosen in the above block diagram 1000 random bytes are generated and then repeated. The basic sample rate is set to 100k samples per second and the Throttle block limits the dataflow to (approximately) 100,000 bytes per second. The Constellation Modulator Properties are set to not use differential encoding, to use 8 samples per symbol via the `sps` variable, and to use an excess bandwidth of 0.35 (i.e., $\alpha = 0.35$ for the root raised cosine pulse shape). The data input to the modulator is in the form of (8-bit) bytes which are transmitted serially, MSB first. Thus, the bit rate is 800 kbits/sec (`samp_rate2` in the flowgraph) and the overall sampling rate at the output of the modulator is `samp_rate2*sps` which amounts to 6.4 Msps/sec. The signal points and their binary representation are set by the Constellation Object. Here we use symbols 0 and 1 which are mapped to constellation points 0 ("off") and 1 ("on"), respectively. Because the Constellation Modulator assumes a distance of 2 between the two constellation points, the (generally complex-valued) output is multiplied by 0.5 to obtain a signal amplitude of 1. The output of the modulator is displayed using a QT GUI Sink and it is also recorded in a file for further processing, e.g., in Python. The PSD of the generated OOK signal has the following shape.



The dc component that results from OOK is clearly visible. Note also the shape and the bandwidth of the spectrum. The Constellation Modulator uses a rrcf (root raised cosine in frequency) pulse shape with an excess bandwidth of $\alpha = 0.35$ in the example shown. Since the bitrate is 800 kHz, the overall bandwidth of the signal is $BW = 1.35 \times 8 \times 10^5/2 = 540$ kHz. In the time domain the signal is purely real (because of the way the constellation was specified) and the bit interval is 1.25 μs, as shown in the next graph.



To obtain the baseband constellation configuration, the real part of the baseband signal is plotted along the horizontal axis and the imaginary part is plotted along the vertical axis at the optimum sampling time which needs to be adjusted by the Delay in the Delay block. Since 8 samples per symbol are used in the flowgraph, a Decimating FIR Filter with a Decimation factor of 8 was used before displaying the signal in a QT GUI Constellation Sink. The Taps of the FIR filter use the `pamhRt` function from the `ptfun.py` module with pulse type `'rrcf'` and $\alpha = 0.35$. The result is shown next.



As expected (when Delay is properly adjusted), the result is two points, one a the origin (when the OOK signal is off) and one at (1,0) (when the OOK signal is on).

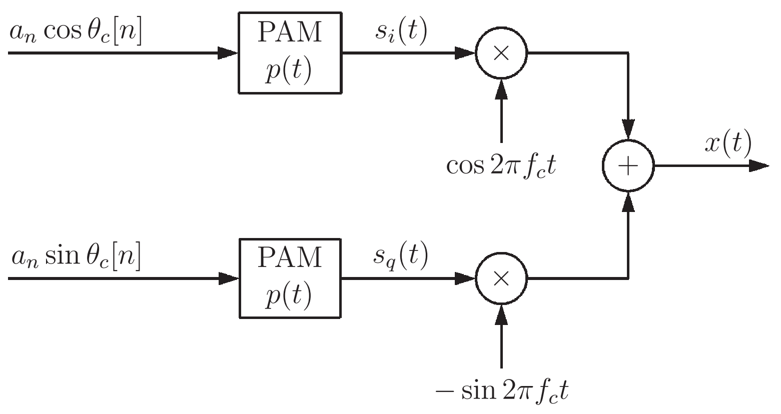
2 Lab Experiments

E1. Amplitude Shift Keying. (a) Start a new Python module called `keyfun.py` and use your **pam12** function from the `pamfun` module as a building block to complete the ASK transmitter function `askxmtr` whose header is shown below.

```
# File: keyfun.py
# Functions for amplitude/frequency/phase shift keying
# ASK, FSK, PSK and hybrid APSK
from pylab import *
import pamfun

def askxmtr(anthcn,FB,Fs,ptype,pparms,xtype,fcparms):
    """
    Amplitude Shift Keying (ASK) Transmitter for
    Choherent ('coh') and Non-coherent ('noncoh') ASK Signals
    >>>>> tt,xt,st = askxmtr(anthcn,FB,Fs,ptype,pparms,xtype,fcparms) <<<<<
    where  tt:      time axis for x(t), starts at t=-TB/2
           xt:      transmitted ASK signal, sampling rate Fs
                    x(t) = s(t)*cos(2*pi*fc*t+(pi/180)*thetac)
           tt:      time axis for x(t), starts at t=-TB/2
           st:      baseband PAM signal s(t) for 'coh'
           st = sit + 1j*sqt  for 'noncoh'
           sit:     PAM signal of an*cos(pi/180*thetacn)
           sqt:     PAM signal of an*sin(pi/180*thetacn)
           xtype:   Transmitter type from list {'coh','noncoh'}
           anthcn = [an]            for {'coh'}
           anthcn = [[an],[thetacn]]    for {'noncoh'}
           an:      N-symbol DT input sequence a_n, 0<=n<N
           thetacn: N-symbol DT sequence theta_c[n] in degrees,
                    used instead of thetac for {'noncoh'} ASK
           FB:      baud rate of a_n (and theta_c[n]), TB=1/FB
           Fs:      sampling rate of x(t), s(t)
           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'}
           fcparms = [fc, thetac]  for {'coh'}
           fcparms = [fc]          for {'noncoh'}
           fc:      carrier frequency in Hz
           thetac:  carrier phase in deg (0: cos, -90: sin)
    """
```

To generate noncoherent ASK signals with a random carrier sequence $\theta_c[n]$, use in-phase and quadrature PAM signals $s_i(t)$ and $s_q(t)$ and QAM modulation as shown in the following blockdiagram.



Test your transmitter by generating a short (about 10 symbol times) random coherent OOK signal and a short random noncoherent OOK signal and displaying them in the time domain. Use $F_s = 44100$ Hz, $F_B = 100$ baud, $f_c = 300$ Hz, and a rectangular pulse $p(t)$ of width $T_B = 1/F_B$. Generate a uniformly distributed random phase for noncoherent OOK using the `rand` function.

(b) In the `keyfun.py` module, complete the following ASK receiver function, called `askrcvr`. The goal is to be able to use it to receive coherent and noncoherent ASK signals, and to make scatter plots.

```
def askrcvr(tt,rt,rtype,fcparms,FBparms,ptype,pparms):
    """
    Amplitude Shift Keying (ASK) Receiver for
    Coherent ('coh') and Non-coherent ('noncoh') ASK Signals
    >>>>> bn,bt,wt,ixn = askrcvr(tt,rt,rtype,fcparms,FBparms,ptype,pparms) <<<<<
    where  bn:       received DT sequence b[n]
           bt:       received 'CT' PAM signal b(t)
           wt = wit + 1j*wqt
           wit:      in-phase component of b(t)
           wqt:      quadrature component of b(t)
           ixn:      sampling time indexes for b(t)->b[n], w(t)->w[n]
           tt:       time axis for r(t)
           rt:       received (noisy) ASK signal r(t)
           rtype:    receiver type from list ['coh','noncoh']
           fcparms = [fc, thetac]  for {'coh'}
           fcparms = [fc]          for {'noncoh'}
           fc:       carrier frequency in Hz
           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 b(t)->b[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 the `pamrcvr10` function in module `pamfun` as part of the ASK receiver. Test `askrcvr` together with `askxmtr` using random (unipolar) binary data and the parameters given in part (a).

(c) Let $F_s = 44100$ Hz, $F_B = 100$ baud, $f_c = 2100$ Hz, and let $p(t)$ be a rectangular pulse of width T_B. Use random binary data to produce (i) a coherent OOK signal, (ii) a noncoherent OOK signal, and (iii) a BPSK signal, each of duration 2 sec. Plot and compare the PSDs for all three cases. Then use the `wt` and `wn=wt[ixn]` outputs of the `askrcvr` function to make scatter plots for the three signals and compare them. Are there any interesting spectral lines if you look at the PSDs of the squared ASK signals? How do things change if you use a triangular pulse $p(t)$ (of total width $2T_B$ from $-T_B$ to $+T_B$) instead of the rectangular pulse?

(d) The wav files `asksig901.wav`, `asksig902.wav` and `asksig903.wav` are binary ASK signals which contain 8-bit, LSB-first, ASCII signals. Analyze the three signals and extract the text messages. **Hint:** The signals in the wav files have been scaled so that their amplitude (including noise) is less than or equal to one. Use eye diagrams and/or scatter plots to determine the proper threshold below which the receiver decides that a 0 was received and above which it decides that a 1 was received.

E2. Frequency Shift Keying. (a) For the Python module `keyfun.py`, write a function called `fskxmtr` which implements an M-ary FSK transmitter for either coherent, noncoherent, or continuous phase FSK. The header of this function is shown below.

```
def fskxmtr(M,dnthcn,FB,Fs,ptype,pparms,xtype,fcparms):
    """
    M-ary Frequency Shift Keying (FSK) Transmitter for
    Choherent ('coh'), Non-coherent ('noncoh'), and
    Continuous Phase ('cpfsk') FSK Signals
    >>>>> tt,xt = fskxmtr(M,dnthcn,FB,Fs,ptype,pparms,xtype,fcparms) <<<<<
    where  tt:      time axis for x(t), starts at t=-TB/2
           xt:      transmitted FSK signal, sampling rate Fs
           M:       number of distinct symbol values in d[n]
           xtype:   Transmitter type from set {'coh','noncoh','cpfsk'}
           dnthcn = [dn]           for ['coh','cpfsk']
           dnthcn = [dn, thetacn]  for ['noncoh']
           dn:      M-ary (0,1,..,M-1) N-symbol DT input sequence d_n
           thetacn: N-symbol DT sequence theta_c[n] in degrees,
                    used instead of thetac0..thetacM-1 for {'noncoh'} FSK
           FB:      baud rate of d_n (and theta_c[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 = [[fc0,fc1,...,fcM-1],[thetac0,thetac1,...,thetacM-1]]
                     for {'coh'}
           fcparms = [fc0,fc1,...,fcM-1]  for {'noncoh'}
           fcparms = [fc, deltaf]         for {'cpfsk'}
           fc0,fc1,...,fcM-1:   FSK (carrier) frequencies
                                for {'coh','noncoh'}
           thetac0,thetac1,...,thetacM-1: FSK (carrier) phases in deg
                                (0: cos, -90: sin) for {'coh'}
           fc:      carrier frequency for {'cpfsk'}
           deltaf:  frequency spacing for {'cpfsk'}
                    for dn=0 -> fc, dn=1 -> fc+deltaf,
                    dn=2 -> fc+2*deltaf, etc
    """
```

Test `fskxmtr` by recreating the three (time domain) sample graphs for $F_B = 100$ baud, $f_{c0} = 300$ Hz and $f_{c1} = 400$ Hz, which were given in the introduction for binary coherent FSK, noncoherent FSK, and CPFSK. Use $d_n = \{0, 1, 1, 1, 0, 0, 1, 0\}$ and (for the second graph) $\theta_c[n] = \{270°, 225°, 4°, 135°, 250°, 90°, 40°, 240°\}$.

(b) In the `keyfun.py` module, implement the FSK receiver function `fskrcvr` whose header is given below.

```
def fskrcvr(M,tt,rt,rtype,fcparms,FBparms,ptype,pparms):
    """
    M-ary Frequency Shift Keying (FSK) Receiver for
    Coherent ('coh'), Non-coherent ('noncoh'), and
    Phase Detector ('phdet') FSK Reception
    >>>>> bn,wt,ixn = fskrcvr(M,tt,rt,rtype,fcparms,FBparms,ptype,pparms) <<<<<
    where  bn:       received DT sequence b[n]
           wt = [w0it+1j*w0qt,w1it+1j*w1qt,...,wM-1it+1j*wM-1qt]
           wmit:     m-th in-phase matched filter output
           wmqt:     m-th quadrature matched filter output
           ixn:      sampling time indexes for b(t)->b[n], w(t)->w[n]
           M:        number of distinct FSK frequencies
           tt:       time axis for r(t)
           rt:       received (noisy) FSK signal r(t)
           rtype:    receiver type from list {'coh','noncoh','phdet'}
           fcparms = [[fc0,fc1,...,fcM-1],[thetac0,thetac1,...,thetacM-1]]
                     for {'coh'}
           fcparms = [fc0,fc1,...,fcM-1]  for {'noncoh'}
           fcparms = [fc, deltaf]         for {'phdet'}
           fc0,fc1,...,fcM-1:   FSK (carrier) frequencies
                                for {'coh','noncoh'}
           thetac0,thetac1,...,thetacM-1: FSK (carrier) phases in deg
                                (0: cos, -90: sin) for {'coh'}
           fc:      carrier frequency for {'cpfsk'}
           deltaf:  frequency spacing for {'cpfsk'}
                    for dn=0 -> fc, dn=1 -> fc+deltaf,
                    dn=2 -> fc+2*deltaf, etc
           FBparms = [FB, dly]
           FB:      baud rate of PAM signal, TB=1/FB
           dly:     sampling delay for wm(t)->wm[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'}
    """
```

Depending on the choice of the `rtype`, the receiver should perform demodulation using either M coherent MFs, M MFEDs, or using a phase detctor followed by differentiation and a MF. Test all receiver modes with the signals that you generated in (a) when you were testing `fskxmtr`.

(c) Generate a coherent binary FSK signal from random data with equally likely 0's and 1's, using a rectangular $p(t)$, $F_B = 100$ baud, $f_{c0} = 300$ Hz, $\theta_{c0} = 0°$, $f_{c1} = 400$ Hz, and $\theta_{c1} = 0°$. Use a coherent demodulator to produce a scatter plot of $w_0[n]$ versus $w_1[n]$. Change the phase θ_{c1} from 0° to 180° and check whether the signals transmitted at f_{c0} and at f_{c1} remain orthogonal in the signal space spanned by $w_0[n]$ and $w_1[n]$. Is it possible to reduce the frequency spacing $\Delta_f = f_{c1} - f_{c0}$ to a value less than F_B while maintaining orthogonality? Try changing f_{c1} to 350 Hz and vary the phase θ_{c1} again from 0° to 180°.

(d) Use uniformly distributed random M-ary data of length about 2 sec to generate PSD plots of $M = 2$ and $M = 4$ coherent, noncoherent, and continous-phase FSK (CPFSK) signals with rectangular $p(t)$. Determine the -40 dB bandwidth in all cases. Use $F_B = 100$ baud, $f_{c0} = 2100$ Hz, and $f_{cm} = f_{c0} + mF_B$. For coherent FSK set $\theta_{cm} = 0$ for all m. For CPFSK choose the phases θ_{cm} such that there are no phase jumps.

(e) The wav files `fsksig901.wav` and `fsksig902.wav` contain binary FSK signals made from 8-bit, LSB-first, ASCII encoded characters. Analyze the two signals and extract the text messages.

(f) The binary GNU Radio file `digMod_905.bin` contains several (complex-valued) binary communication signals. Use the GNU Radio Companion to find the signals recorded in the file and determine their properties such as the type of modulation and the carrier frequencies. Each of the signals contains an ASCII coded (MSB first) message. Try to demodulate the signals and extract the messages.

E3. Phase and Amplitude/Phase Shift Keying. –To be completed–