5 Phase Synchronization

In a previous lab we saw that the real part of the base band signal is modulated onto a cosine wave at radio frequency, and the imaginary part onto a sine wave. Since the cosine and sine functions are orthogonal, the two signal parts can be separated in the receiver. However, we run into a similar problem as the one discussed in the last lab: The receiver merely sees an oscillating waveform (a sinusoid): $r(t) \propto \sin(2\pi f_c t + \theta)$ during one symbol duration. Is it a sine, a cosine, or something in between? In order to differentiate between the two base functions, the receiver needs a phase reference, i.e. it needs to know the phase offset θ. The task of the phase synchronization unit is to provide this reference.

5.1 Signal Model

The phase synchronization is performed after the interpolator and thus works on the timing-corrected symbols

$$z_\varepsilon[n] = z(nT + \hat{\varepsilon}T). \tag{5.1}$$

We will assume a perfect timing recovery and interpolation, so that $z_\varepsilon[n] = z(nT + \varepsilon T)$.
In the complex baseband, the phase offset θ translates to a multiplication of the received signal with $e^{j\theta}$. The effect of this is a rotation of the signal space, as illustrated in Figure 5.1. The left plot shows a sequence of received QPSK symbols after timing correction without any phase offset, with an SNR of 8 dB. The borders of the decision regions are the x- and y-axis; for example, the symbol demapper decides that the bits $b_0 b_1 = 11$ have been transmitted if the received symbol lies in the upper right quadrant of the complex plane. In Figure 5.1(a) we can easily recognized the transmitted QPSK constellation, and we see that most symbols will be correctly detected, even though the clouds extend to the coordinate axes and therefore a few bit errors will occur. Figure 5.1(b) shows the same sequence with a phase error of $20°$. It is obvious that the bit error rate will be far worse if we attempt to demap these symbols without correcting the phase offset. The phase offset is not constant but varies slowly over time, e.g. due to non-ideal analog components. Since the variation of the phase is a random process, this effect is called phase noise. The received symbol sequence can now be written as

$$z_\varepsilon[n] = a[n]e^{j\theta[n]} + w'[n]. \tag{5.2}$$

A simple but realistic phase noise model is the random walk: The first phase offset $\theta[0]$ is uniformly distributed in $[0; 2\pi)$. Then, over time, the phase offset evolves according to

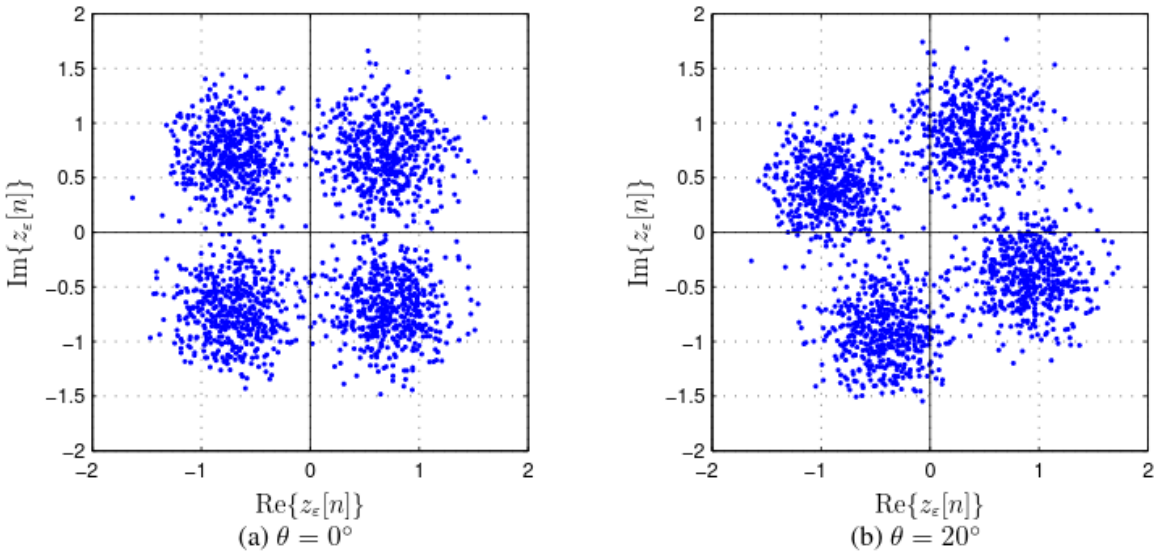


Figure 5.1: Received symbols without (a) and with (b) phase offset

$$\theta[n] = \theta[n-1] + \Delta\theta[n] \quad (\mod 2\pi) \tag{5.3}$$

where the difference between two successive phase offsets is a Gaussian random variable

$$\Delta\theta[n] \sim \mathcal{N}(0, \sigma^2_{\Delta\theta}). \tag{5.4}$$

Figure 5.2 illustrates a typical phase noise process for a standard deviation of $\sigma_{\Delta\theta} = 0.004 \backsimeq 0.23°$. We see that the phase drift is rather slow, but still non negligible if the transmission is longer than about thousand symbols. Therefore, the phase offset must be tracked during the transmission.

5.2 Synchronization via Rotation

The correction of the phase offset is straightforward. Since the phase offset causes a rotation of the signal space,$z_\varepsilon[n] = a[n]e^{j\theta[n]} + w'[n]$. , we simply rotate the signal space into the opposite direction:

$$\begin{aligned} z_{\varepsilon;\theta}[n] &= z_\varepsilon[n]e^{-j\hat{\theta}[n]} \\ &= a[n]e^{j(\theta[n]-\hat{\theta}[n])} + w''[n] \end{aligned} \tag{5.5}$$

with $w''[n] = w'[n]e^{-j\hat{\theta}[n]}$. The phase synchronization unit is illustrated in Figure 5.3.

5.3 The Estimation Algorithm

The estimation algorithm, which is known in the literature as the "Viterbi and Viterbi" algorithm, uses the fact that we have a QPSK modulation. We therefore know that

$$a[n] \in \{e^{j(\frac{\pi}{2}l+\frac{\pi}{4})} \mid l \in \{0,1,2,3\}\} \tag{5.6}$$

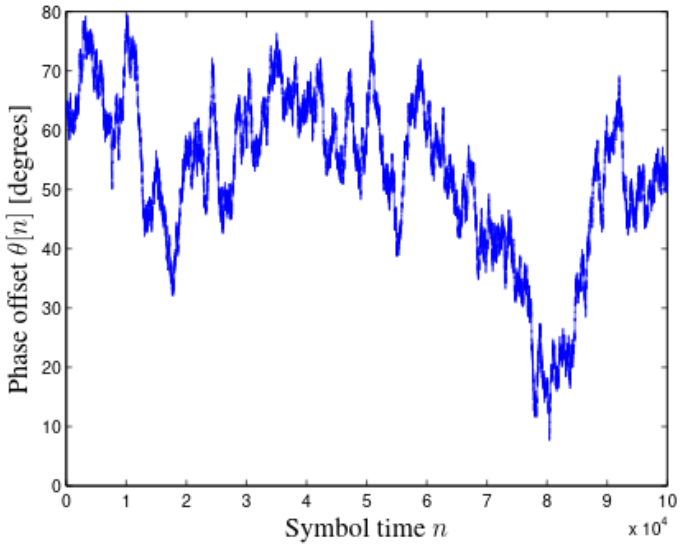


Figure 5.2: Typical phase noise process for $\sigma_{\Delta\theta} = 0.004$

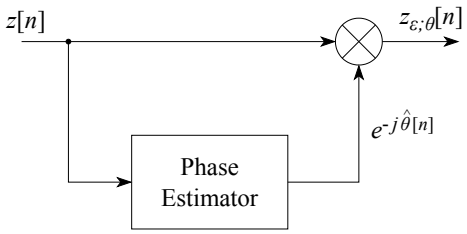


Figure 5.3: Phase synchronization unit

for all transmitted symbols $a[n]$, and thus

$$a[n]^4 = e^{4j(\frac{\pi}{2}l+\frac{\pi}{4})} = e^{j(2\pi l+\pi)} \equiv -1. \tag{5.7}$$

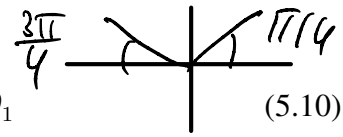
The operation $(\cdot)^4$ removes the information from the data symbols. Exploiting this fact, we can derive the following estimator, which operates on a symbol-by-symbol basis (i.e. does not take any correlation between the phases into account):

$$\hat{\theta}[n] = \frac{1}{4}\arg\{-z_\varepsilon[n]^4\}. \tag{5.8}$$

This estimator is approximately unbiased if the SNR is high:

$$\begin{aligned} E\{\hat{\theta}[n]\} &= E\{\tfrac{1}{4}\arg\{-z_\varepsilon[n]^4\}\} \\ &= E\{\tfrac{1}{4}\arg\{-(a[n]e^{j\theta[n]} + w'[n])^4\}\} \\ &\overset{(a)}{\approx} \tfrac{1}{4}\arg\{E\{-(a[n]e^{j\theta[n]} + w'[n])^4\}\} \\ &\overset{(b)}{=} \tfrac{1}{4}\arg\{e^{j4\theta[n]}\} \\ &\overset{(c)}{=} \theta[n]. \end{aligned} \tag{5.9}$$

The approximation (a) is valid for a high SNR, because then the variation of the argument is small, and the nonlinear argument operation can be linearized. In this case, the expectation and the argument operation can be exchanged. Equation (b) holds because the noise has zero-mean and is uncorrelated with the data sequence.
The step (c) does not allow us to uniquely identify any $\theta \in [0; 2\pi)$. To see this, consider, for example, $\theta_1 = \frac{\pi}{4}$ and $\theta_2 = \frac{3\pi}{4}$ (we drop the symbol index for simplicity). Then, the estimator will give us:



$$\hat{\theta_1} = \tfrac{1}{4}\arg\{e^{j4\theta_1}\} = \tfrac{1}{4}\arg\{e^{j\pi}\} = \frac{\pi}{4} = \theta_1 \tag{5.10}$$

$$\hat{\theta_2} = \tfrac{1}{4}\arg\{e^{j4\theta_2}\} = \tfrac{1}{4}\arg\{e^{j3\pi}\} = \frac{\pi}{4} \overset{!}{\neq} \theta_2 \tag{5.11}$$

We can only identify θ correctly if we know which quadrant it is in. One way to have access to this information is to consider the phase estimate for the previous symbol. Assuming that the phase changes slowly, the two consecutive phases will most likely be close to each other (BUT: not necessarily in the same quadrant). However, in order to use this method we need a initial phase information about which quadrant we can start from.

5.4 Initial Phase Estimation

An initial phase estimate can be obtained by exploiting the preamble that is used for frame synchronization in the following way. Assume that there is no noise, so that $z[n] = a[n]e^{j\theta[n]}$, and let $p[n]$ denote the preamble sequence of length N_p . The frame synchronization function computes the following correlation:

$$c[n] = \sum_{i=0}^{N_p-1} \bar{p}^*[i]z[n-i], \tag{5.12}$$

where $p[i] = p[N_p - 1 - i]$ is the reverted preamble sequence. $c[n]$ is maximized when the sequences $z[n-i]$ and $p[i], i = 0, ..., N_p - 1$, are equal. Let the n which maximizes the correlation be denoted by n_{max} . Assuming that $\theta[n_{max} - i] \approx \theta$ over the preamble, we have:

$$c[n_{max}] = \sum_{i=0}^{N_p-1} \bar{p}^*[i]p[n_{max}-i]e^{j\theta[n_{max}-i]} \tag{5.13}$$

$$\approx \sum_{i=0}^{N_p-1} \bar{p}^*[i]p[n_{max}-i]e^{j\theta} \tag{5.14}$$

$$= e^{j\theta}\sum_{i=0}^{N_p-1} \bar{p}^*[i]p[n_{max}-i]. \tag{5.15}$$

For the phase of $c[n_{max}]$ we have:

$$\arg\{c[n_{max}]\} = \theta. \tag{5.16}$$

Thus, the phase of the peak of the correlator can be used as an initial phase estimate.

5.5 Your Tasks

A5T1 Create phase noise by implementing the random walk described in (5.3).

A5T2 Modify the phase synchronization unit so that it provides an initial phase estimate according to (5.16).

A5T3 Implement the phase synchronization unit and integrate it into your system. Compare your phase estimates with the true values.

(1) Apply phase noise to signal with awgn noise.

(2) Apply viterbi-viterbi algorithm on the payload data.

(3) Correct the payload with the estimated phase error.

(4) Consider a way to filter the phase according to the nature of random walk, write a new theta estimation trial in parallel with (2) and (3) to allow a comparison of the estimate.

Note: MATLAB *angle* function returns angles in the interval $[-\pi, \pi]$. After step (3), the image will still experience some erroneous phase jump unless a very high SNR is used (e.g. 100dB).