Get Start with gr-cdma

Yu Wang
umwangyu@umich.edu

August 31, 2016

Contents

Chapter 1

Introduction

1.1 What is gr-cdma

Project gr-cdma is the physical layer of CDMA communication. This project is developed based on GNURadio[1]. gr-cdma is created and maintained by Prof. Anastasopoulos. The whole source codes are on github `https://github.com/anastas/gr-cdma`. This project is to build a communication block which employs DS-CDMA technique to build the link. The user does not have to know the details of the schemes, but could transmit reliably through this physical layer protocol.

1.2 What is GNURadio

GNURadio[1] is a open-source *unix based platform which enables developers and researchers design and test their self-built communication or signal processing system. The whole platform is written by c++. Developer could use c++ or python to customize their own block. gr-cdma is a customized block as well.

1.3 What is the Current State of gr-cdma

Till 17-Jun-2016, the main part of this project is finished. The next step is to optimize the working strategy for several block to make the system more robust in field test. This document is an introduction of what this project is, how to get start with GNURadio and gr-cdma and what are the issues to be solved.

Chapter 2

Working Strategy of gr-cdma

In this chapter, we will discuss the working strategy of gr-cdma, the principle of each main block. Mathematic models and analyses are involved to describe the situation.

2.1 The diagram of The Tx & Rx

Here are the digrams for Tx and Rx.

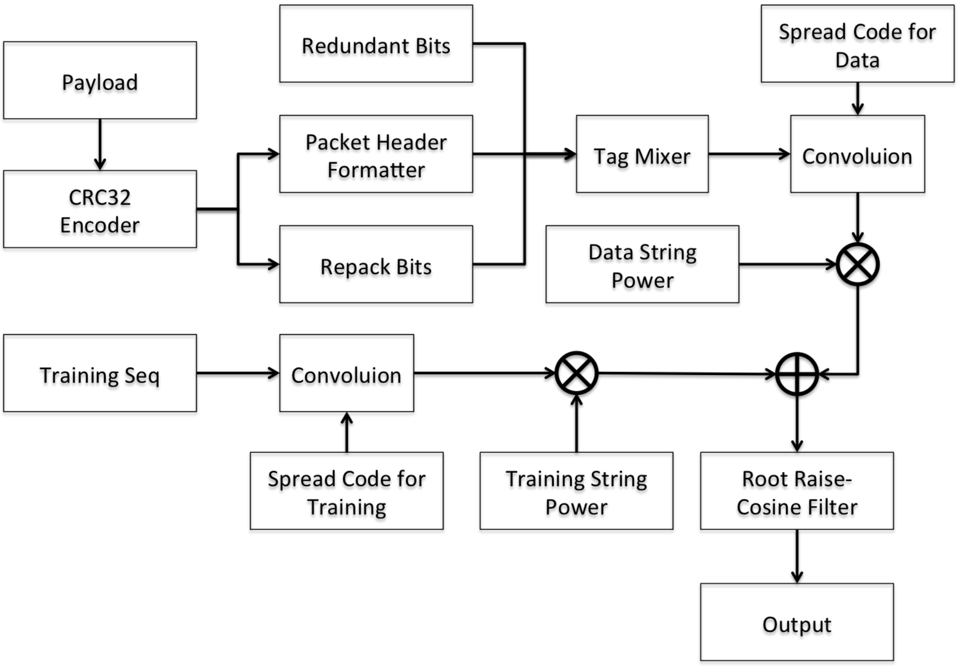


Figure 2.1: Diagram for Tx

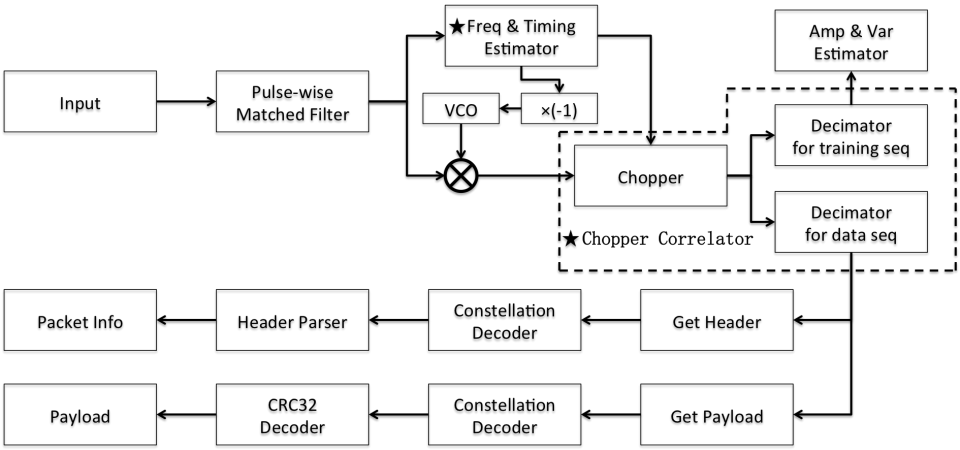


Figure 2.2: Diagram for Rx

From fig 2.1 and fig 2.2, you may find that the system is a little bit more complicated than you thought. And also, the receiver is bigger than the transmitter. In fact, it is for the reason that the receiver will deal with no only the decoding part, but also the synchronization problem. which is actually the most difficult part for this system.

2.2 Transmitter

The duty for transmitter is to package the payload or the input data, build the packet structure, generate the training sequence, modify the power for each section and form the waveform.

2.2.1 Spreading Code

Similar to other CDMA system, one of the essential part of the system is about the spreading code. The quality of the spreading code will affect greatly about the performance of synchronization parts. Intuitively, we would like to use codes like m-sequences and gold sequence, which has high auto-correlation and flat cross-correlation value.

The most famous spreading code is the m-sequence.

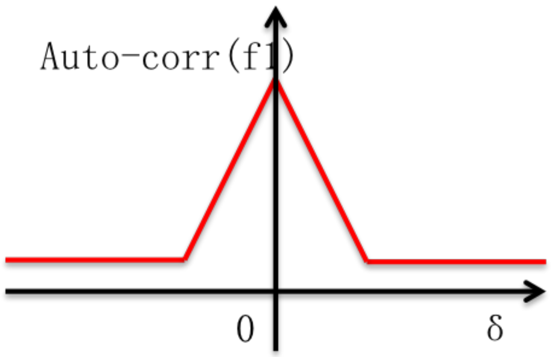
2.2.1.1 M-Sequence

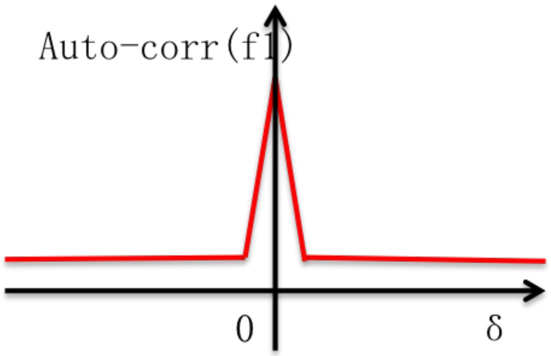
M-sequence is a kind of sequnece with very good property. for m-sequnece $p(k)$,

$$p(k) * p(k + \delta_k) = \begin{cases} m & \delta_k = 0 \\ -1 & \delta_k \neq 0 \end{cases} \tag{2.1}$$

m is the length of sequence, δ_k ranges from $-m+1$ to $m-1$.

Here is the table for some m-sequence, table 2.1





2.2.1.2 Gold Sequence

But also as shown in the table above, the number of m-sequence for certain length is not that much. So, some other sequences like “Gold sequences” are generated. “Gold sequence” is actually the shift-add of two m-sequence. When the shift space for tow m-sequence is different, more relatively good sequences are generated.

2.2.1.3 Spreading Seuquence used in gr-cdma

But for gr-cdma, we adopt sequence of length 8. From the diagram, we know that a functioning system needs two spreading sequences for “Training” and “Data”. So we only need two relatively good sequences. Because the length of codes are not long enough, so brutal search is feasible. For current system, the two sequences are as shown in table 2.2

2.2.2 Packet Structure

Packeting is also one main part for Transmitter. In physical layer, transmitting purely meaningless 0s and 1s reliably and efficiently is good. But for a system, we may care the index of each sequence, which is good for placing different pieces in order, modulation scheme, adaptive modulations may be used, and so on. The training and data packet structure for gr-cdma is like figure 2.4 and figure 2.3:

Table 2.1: M-sequence feedback connections generated using LFSR generator

Number of shift registers, L	Code length $N_c = 2^l - 1$	Feedback Taps for M-sequence	No. of M-sequence codes
2	3	[2,1]	1
3	7	[3,2]	2
4	15	[4,1]	2
5	31	[5,3], [5,4,3,2], [5,4,2,1]	6
6	63	[6,1], [6,5,2,1], [6,5,3,2]	6
7	127	[7,1], [7,3],[7,3,2,1], [7,4,3,2], [7,6,4,2], [7,6,3,1], [7,6,5,2], [7,6,5,4,2,1], [7,5,4,3,2,1]	18
8	255	[8,4,3,2], [8,6,5,3], [8,6,5,2], [8,5,3,1], [8,6,5,1], [8,7,6,1], [8,7,6,5,2,1], [8,6,4,3,2,1]	16
9	511	[9,4], [9,6,4,3], [9,8,5,4], [9,8,4,1], [9,5,3,2], [9,8,6,5], [9,8,7,2], [9,6,5,4,2,1], [9,7,5,3,1], [9,8,7,6,5,3]	48
10	1023	[10,3], [10,8,3,2], [10,4,3,1], [10,8,5,1], [10,8,5,4], [10,9,4,1], [10,8,4,3], [10,5,3,2], [10,5,2,1], [10,9,4,2], [10,6,5,3,2,1], [10,9,8,6,3,2], [10,9,8,7,6,5,4,3], [10,8,7,6,5,4,3,1]	176

Table 2.2: Spread Codes for gr-cdma

Type of Sequence	Sequence
Training Sequence	[+1, +1, +1, +1, -1, +1, +1, -1]
Data Sequence	[-1, +1, -1, +1, -1, -1, -1, -1]

In current version, because the number of binary bits for each Segment of the frame is set through "python/cdma_parameter.py", the length for these part is

2.2.3 Power Control

As shown in the figure 2.1, training sequence and data sequence are two separate branches. In reality, the power a transmitter could provide is certain, so before adding them up, the power arrangement should be considered. In "python/cdma_parameter.py", variable "training_percent" is to control the percentage of power used for Training sequence.

2.2.4 Multi-Sampling and Pulse Shaping

Multi-Sampling is good for synchronization. In modern CDMA communication systems, like GPS, each chip will be sampled twice. Besides, we could not transmit digital signals directly, blocks like Root Raise-Cosine Filter will be

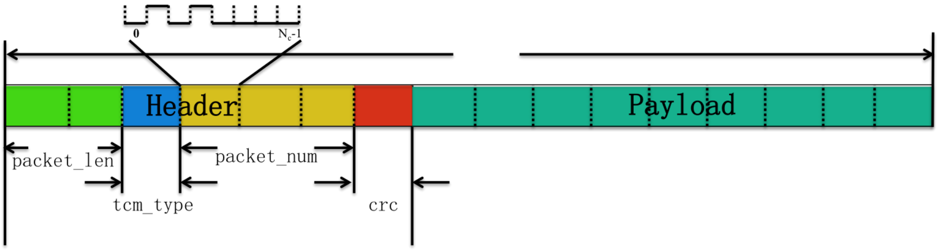


Figure 2.3: Data Frame

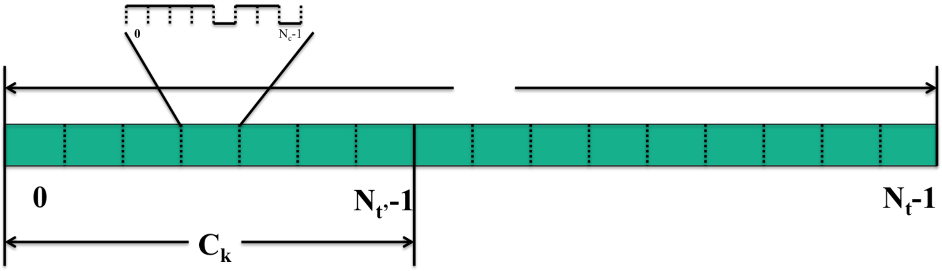


Figure 2.4: Training Frame

added at the end of the block. The parameter set for RRCF is

2.3 Receiver

Comparatively, receiver is more complicated than transmitter. Because in the real time, receiver does not have any knowledge of received signal's time and frequency shift. As shown in the diagram Figure 2.2, the most important block for receiver is not the decoding part, but the frequency and timing estimator. The theory for frequency and timing estimator will be shown in detail in Chapter 3

2.3.1 Frequency and Timing Estimator

Frequency and Timing Estimator generally gives the estimation of timing information and frequency shift. We do not design a frequency estimator and timing estimator separately, but together.

2.3.1.1 Timing Information

In communication system, information bit sequence needs to be packaged with certain structure form. Same as gr-cdma, we also create a certain frame structure. The structure is shown in Figure 3.3 and Figure 3.4. When we need to decode them, or say extract the payload from the frame, we need to know "Where should we start cutting". We want to remain exactly the information bit sequence, rather than a broken sequence with other "meaningless" bits. So, we need to get the start point of each frame.

Table 2.3: Length for Each Part of Frame

Frame Type	Segment Type	Segment Length
Training	/	260
Data	packet length	12
Data	tcm type	4
Data	packet index	16
Data	header crc	8

Table 2.4: Parameters for Root Raise-Cosine Filter

Parameter	Value
Samples Per Chip	2
The rolloff factor	0.35
Filter length	21

The method we use is pretty straight forward. We use another frame called Training Frame. The output signal is the sum of Training and Data. Because we carefully select the spread code for training and data, they will not mass up when added up together. Each training frame are exactly the same. So, we could use a matched filter to find the end point of the frame. Because when the matched filter outputs a peak, it means there exists a potential whole length training frame in the received signal sequence. Thanks to the design that training frame and data frame are perfectly aligned, the ending point for training frame is the same as the ending point for data frame. So, we solve the problem of timing.

2.3.1.2 Frequency Information

To be precisely, the frequency information we talk here is the frequency shift provided by the channel. In real environment, reasons like clock drift and Doppuler effect can change the received signal's frequency. If the transmitter and receiver's oscillator are not synchronized, the performance will not be that good. So, the receiver will try to find out the frequency difference between the received signal and local oscillator, and narrow the gap as small as possible.

Frequency estimator share nearly the same idea as timing estimator. It is true that there are many other methods like doing FFT and find out the frequency gap. But as mentioned, we need to do a joint estimation of both timing and frequency. We designed a matched filter bank to deal with this problem.

2.3.1.3 Structure for Joint Estimator

The diagram for Frequency and Timing estimator is shown in Figure 2.5.

In gnuradio, there is a block called “xlating filter”. It works exactly the same as multiplying a frequency and then do matched filtering.

Because there is a large number of matched filter in this block, we need to make this block working as efficiently as it could. We choose to use the FFT version of filter rather than the traditional one.

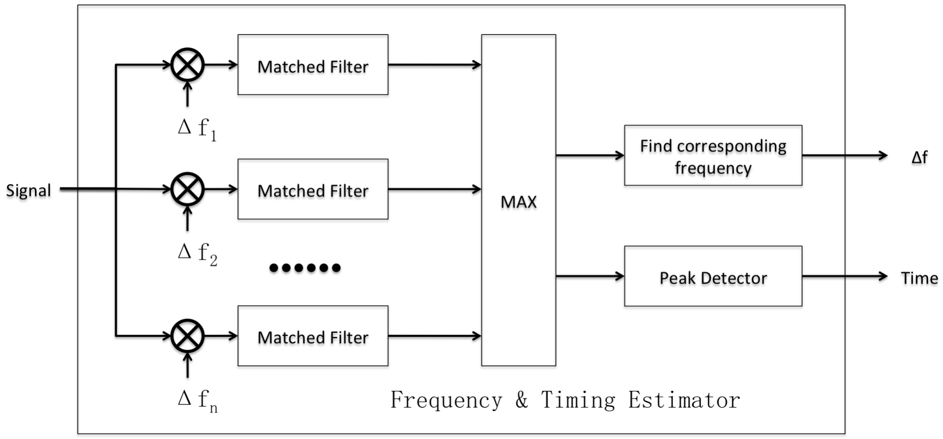


Figure 2.5: Diagram for Frequency & Timing Estimator

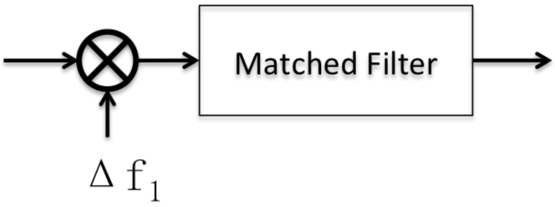


Figure 2.6: xlating-filter in gnuradio companion

2.3.2 De-spreading

De-spreading is to demodulate the CDMA signal. The method is straight forward. Use a matched filter which equipped with the patten matching spreading code for training / data, then see whether the peak is positive or negative. Then, sample only at positions where we see peak.

The block in gnuradio that can do this is called "decimation".

2.3.3 Packet Parser

Packet Parser only check the header information. In data frame, there is a field called "CRC". It the receiver could generate exactly the same CRC as the bits in the CRC field, this frame will be regarded as valid. It is true that, there are cases where the packet is broken, but the CRC happens to be "Correct". We have nothing to do with cases of such.

2.3.4 Error Calculator

This block is to give a rough estimation of packet error rate. The working strategy is as below.

There are two counters. One counter counts the number of packets pass the Packet Parser, which are regard as valid packet. The value for this counter is called C_{vp}. Another counter counts the total number of any packets that received in one iteration. This counter's value is called C_{all}. This block has one parameter called “Window Size”. Every time, when $C_{vp} = WindowSize$, we will output the ratio C_{vp}/C_{all}. But, we should bury in mind that, the Packet Parser will output only the valid packet. So, how do we count C_{all}? To solve this problem, we choose a little complicated way. We use the index of each packet. If the packets’ indices are consecutive, like $1, 2, 3...$, then we will add 1, 1 and 1 to both C_{vp} and C_{all}. But when we miss a packet, we may see indices like $1, 2, 5$, we know that there should have missed $5-2-1 = 2$ packets. But pay attention, the index for packet could not be infinity, there is a limit. If the largest index is 255, when we see $254, 255, 0$, we should not treat it as nonconsecutive. Also, we need to treat problems when we see $254, 255, 2$. Out solution to treat situations where we face the end of one iteration of index is to add the number of iteration to the index which has already start from 0. When we see $255, 0$, we will add 256 to 0. So we will deal with sequence $255, 256$, they are consecutive obviously. But the next step would return to deal with $0, 1$, rather than $256, 257$.

Chapter 3

Mathematic Bases

3.1 Introduction

CDMA system is famous for its large user capacity and superior ability of keeping security. In practical CDMA system, the diagram of the whole system is like

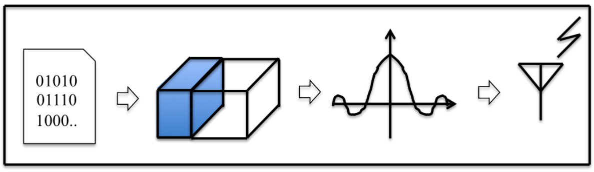


Figure 3.1: TX model

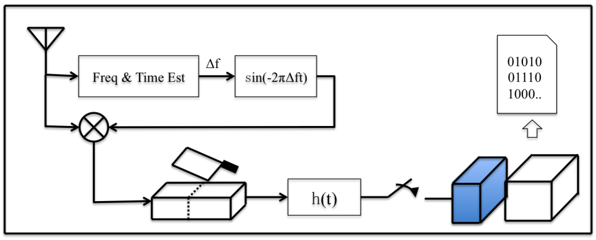


Figure 3.2: RX model

From the diagrams, we can see that the complexity of RX model is much more complicated then TX model. The reason is that, the channel will add several uncertainty to the signal, while the transmitter doesn't have to deal with that. So, the main block in the RX model is the "freq_&time_est", which will give out the frequency and timing information of the received signal. This article is mainly focus on the performance of this block.

3.2 Frequency and TIming Trade-off in Estimation

Trade-off is common in the world. There is one famous theory called uncertainty principle[2]. Also, there is also a trade off for estimating frequency and timing.

In this scenario, we are trying to estimate the frequency offset of the received signal, and estimate the start point of the frame. We will prove that, the estimation cannot be perfect.

3.2.1 Mathematical Model

The signal we use is $s(t)$:

$$s(t) = \begin{cases} \alpha & 0 \leq t \leq T_s \\ 0 & else \end{cases} \tag{3.1}$$

This is a very simple rectangle wave. For the receiver, we only consider about delay t_0 and frequency offset Δf, the received signal $r(t)$ would be

$$r(t) = s(t - t_0)e^{j2\pi\Delta f t} \tag{3.2}$$

But for the receiver, it doesn't know anything about the receiver, so the only tag receiver could choose for the matched filter is $h(t)$

$$h(t) = s(T_s - t) \tag{3.3}$$

Then, the output of the matched filter is $R(t)$, the result of convolution between $r(t)$ and $h(t)$.

$$\begin{aligned} R(t) = r(t) * h(t) &= \int_{-\infty}^{+\infty} r(\tau)h(t-\tau)d\tau & (3.4) \\ &= \int_{t_0}^{t} s(\tau - t_0)e^{j2\pi\Delta f\tau}s(\tau - (t - T_s))d\tau & (3.5) \\ &= \int_{t_0}^{t} \alpha^2 e^{j2\pi\Delta f\tau}d\tau & (3.6) \\ &= \alpha^2 \frac{e^{j2\pi\Delta f t}}{j2\pi\Delta f}\bigg|_{t_0}^{t} & (3.7) \\ &= \alpha^2 \frac{e^{j2\pi\Delta f t} - e^{j2\pi\Delta f t_0}}{j2\pi\Delta f} & (3.8) \end{aligned}$$

Now, we will discuss three specific situations:

3.2.1.1 $\Delta f = 0$ and $t_0 = 0$

We will find that, the numerator and denominator are all 0, so we will turn to L'Hospital.

$$\begin{aligned} &\lim_{\Delta f \to 0} \alpha^2 \frac{e^{j2\pi\Delta f t} - e^{j2\pi\Delta f t_0}}{j2\pi\Delta f} & (3.9) \\ =&\lim_{\Delta f \to 0} \alpha^2 \frac{j2\pi t e^{j2\pi\Delta f t} - j2\pi t_0 e^{j2\pi\Delta f t_0}}{j2\pi} & (3.10) \\ =&\alpha^2(t - t_0) & (3.11) \end{aligned}$$

When $t_0 = 0$, $R(t) = \alpha^2 t$. And we know that, because of the domain of t, we know that, when $t = T_s$, $R(t)$ will get its maximum value $R(Ts) = \alpha^2 T_s$.

3.2.1.2 $\Delta f = 0$, $t_0 = ?$

Similar to the procedure in part 3.2.1.1, also sample at $t = T_s$

$$R(t) = \alpha^2(T_s - t_0) \Rightarrow \hat{t_0} = T_s - \frac{R(t)}{\alpha^2} \tag{3.12}$$

3.2.1.3 $t_0 = 0$, but $\Delta f = ?$

Sample at $t = T_s$

$$R(t) = \alpha^2 \frac{e^{j2\pi\Delta f T_s} - 1}{j2\pi\Delta f} = \alpha^2 T_s e^{j\pi\Delta f T_s} sa(\Delta 2\pi f Ts) \tag{3.13}$$

So, we know that, when we get the correct Δf, we will get the largest value.

But we still haven't touch the situation where $\Delta f = ?$ and $t_0 = ?$. Our method is brute search. We enumerate all the possible pair of parameters $(t_0', \Delta f')$, and try them. With the help of above three theory, we know that, the correct/similar to true value will give out the largest $R(t)$.

3.3 Input Signal of Peak Detector

For the "freq_timing_est" block, frequency and timing information are supposed to be estimated. As discussed in the "freq_timing_tradeoff", we can use brutal search to get the correct value. But, before we make our decision by choosing the maximum peak, we still need to have a rigorous mathematic model for the peak.

This section will discuss the mathematic model for "True" peak and other points.

3.3.1 Mathematical Model

Because the signal we received are discrete values, so we only consider about discrete signals, rather than continuous ones.

First, the structure of training and data frame are like: The structure of training is like Fig. 3.3 and Fig. 3.4

3.3.1.1 Description of Variables

We first express each variables by mathematical languages.

According to the description in table 3.1, and also in order to simplify the following notation, the definition of P_y^x is defined as

$$\begin{aligned} P_k^x &= \begin{cases} P_k^x & k = 0, \cdots, N_c - 1 \\ 0 & else \end{cases} \\ x &= “t''or“d'' \end{aligned} \tag{3.14}$$

For instance, the figure for P_k^t is like Fig. 3.5

Notation	Description
s_k	received sample at time k
t_k	training sample at time k
d_k	data sample at time k
n_k	noise at time k
P_k^t	the k^{th} chip of spread code for one frame of training
P_k^d	the k^{th} chip of spread code for one frame of data
D_l^t	the l^{th} symbol for training
D_l^d	the l^{th} symbol for data
N_f	number of chips per frame
N_c	number of chips per symbol
N_s	number of symbols per frame
$N_{s'}$	number of symbols per shortened training sequence
σ^2	variance of Gaussian noise
k_0	number of samples of delay
f_0	frequency offset
T_c	duration between two chips

Table 3.1: Notation Table

So, training chip at time k can be described as:

$$t_k = \sum_{k=0}^{N_c} D_0^t P_k^t \tag{3.15}$$

$$= D_0^t P_k^t \tag{3.16}$$

If we consider the frame-level periodicity, t_k would be

$$t_k = D^t_{\lfloor k/N_c \rfloor \% N_s} P^t_{k \% N_c} \tag{3.17}$$

Based on nearly the same idea, d_k can be written as:

$$d_k = D^d_{\lfloor k/N_c \rfloor} P^d_{k \% N_c} \tag{3.18}$$

Noise is simple. We simply choose IID Gaussian noise. So

$$n_k \sim \mathcal{N}(0, \sigma^2) \tag{3.19}$$

3.3.1.2 Description of Output of Matched Filter

Consider the time delay and frequency offset, the received signal would be

$$r_k = (t_{k-k_0} + d_{k-k_0}) e^{j2\pi f_0 k T_c} + n_k \tag{3.20}$$

We first deal with the contribution of noise. Define the Pulse Response of Matched Filter to be $h_{N_f - k}$.

$$h_k = \begin{cases} P_k^t = D^t_{\lfloor k/N_c \rfloor} P^t_{k \% N_c} & k = 0, \cdots, N_f \\ 0 & else \end{cases} \tag{3.21}$$

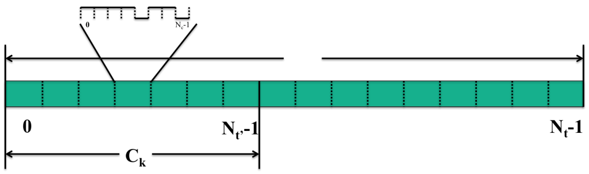


Figure 3.3: Training Frame Structure

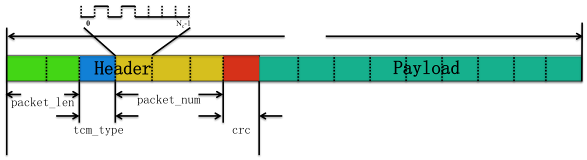


Figure 3.4: Data Frame Structure

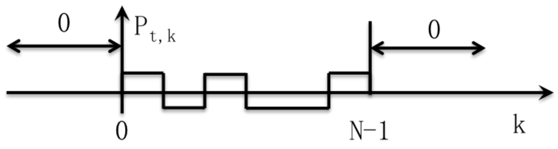


Figure 3.5: Sketch Map of P^t_k

Let $g^n_{k_0} = \sum_{k'=0}^{N_f-1} n_{k'} \cdot h_{k'}$

$$\mathbb{E}\{g_{k_0}^n\} = \mathbb{E}\{\sum_{k'=0}^{N_f-1} n_{k'} \cdot h_{k'}\} \tag{3.22}$$

$$= \sum_{k'=0}^{N_f-1} \mathbb{E}\{n_{k'} \cdot h_{k'}\} \tag{3.23}$$

$$= \sum_{k'=0}^{N_f-1} \mathbb{E}\{n_{k'}\} h_{k'} \tag{3.24}$$

$$= \sum_{k'=0}^{N_f-1} 0 \cdot h_{k'} \tag{3.25}$$

$$= 0 \tag{3.26}$$

$$\mathbb{E}\{(g_{k_0}^n)^2\} = \mathbb{E}\{\sum_{k'=0}^{N_f-1} (n_{k'} \cdot h_{k'}\} \tag{3.27}$$

$$= \mathbb{E}\{\sum_{k'=0}^{N_f-1} n_{k'}^2 \cdot \underbrace{(h_{k'})^2}_{\equiv 1}\} \tag{3.28}$$

$$= \sum_{k'=0}^{N_f-1} \mathbb{E}\{n_{k'}^2\} \tag{3.29}$$

$$= N_f \sigma^2 \tag{3.30}$$

So $g_{k_0}^n \sim \mathcal{N}(0, N_f\sigma^2)$

The next step is to analyze the output of Matched Filter. Define the patten for Matched Filter for training frame to be P_{-k}^{t*}. The output of the Matched Filter at sampled point $k = N_f$ is:

$$g_{k_0} = r_k * h_{N_f-k}\Big|_{k=N_f-1} \tag{3.31}$$

$$= (t_{k-k_0} + d_{k-k_0} + n_k) * h_{N_f-1-k}\Big|_{k=N_f-1} \tag{3.32}$$

$$= \underbrace{\sum_{k'=0}^{N_f-1} t_{k'-k_0} \cdot h_{k'}}_{g_{k_0}^t} + \underbrace{\sum_{k'=0}^{N_f-1} d_{k'-k_0} \cdot h_{k'}}_{g_{k_0}^d} + \underbrace{\sum_{k'=0}^{N_f-1} n_{k'} \cdot h_{k'}}_{g_{k_0}^n} \tag{3.33}$$

$$= g_{k_0}^t + g_{k_0}^d + g_{k_0}^n \tag{3.34}$$

Now, we will discuss all the possible situations for timing. Let's formate the time offset k_0 in terms of N_s and N_c.

$$k_0 = \alpha \cdot N_c + \beta \quad \alpha = \mathbb{Z} \; and \; \beta = 0, \cdots, N_c - 1 \tag{3.35}$$

3.3.1.2.1 Perfectly Aligned Perfectly Aligned means the sample at time k is also the first sample of patten for Matched Filter, as shown in fig. 3.6. For

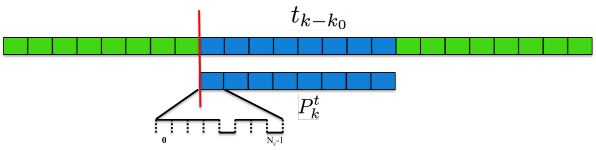


Figure 3.6: Perfectly Aligned

this situation, k_0 satisfies

$$\begin{cases}\alpha = z \cdot N_s \ z \in \mathbb{Z} \\ \beta = 0\end{cases} \quad \Rightarrow k_0 = z \cdot N_s \cdot N_c \tag{3.36}$$

Then we can analyze $g^t_{k_0}$ and $g^d_{k_0}$ separately.
For $g^t_{k_0}$:

$$g^t_{k_0} = \sum_{k'=0}^{N_f-1} t_{k'-k_0} \cdot h_{k'} \tag{3.37}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{\lfloor (k'-k_0)/N_c \rfloor \% N_s} P^t_{k' \% N_c} \cdot h_{k'} \tag{3.38}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{\lfloor (k'-z \cdot N_s \cdot N_c)/N_c \rfloor \% N_s} P^t_{k' \% N_c} \cdot h_{k'} \tag{3.39}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{\lfloor k'/N_c \rfloor \% N_s} P^t_{k' \% N_c} \cdot h_{k'} \tag{3.40}$$

$$\tag{3.41}$$

$$\because h_{k'} = D^t_{\lfloor k'/N_c \rfloor} P^t_{k' \% N_c} \tag{3.42}$$

$$\therefore g^t_{k_0} = \sum_{k'=0}^{N_f-1} D^t_{\lfloor k'/N_c \rfloor \% N_s} P^t_{k' \% N_c} \cdot D^t_{\lfloor k'/N_c \rfloor} P^t_{k \% N_c} \tag{3.43}$$

$$= \sum_{k'=0}^{N_f-1} \underbrace{(D^t_{\lfloor k'/N_c \rfloor \% N_s})^2}_{\equiv 1} \underbrace{(P^t_{k' \% N_c})^2}_{\equiv 1} \tag{3.44}$$

$$= N_f \tag{3.45}$$

To analyzie g^s_k, we will first define a function to describe the convolution

$$C^n_s(f_1, f_2, k_0) = \sum_{k=s}^{s+n} f_1(k) \cdot f_2(k - k_0) \tag{3.46}$$

The diagram of the above formula is like Fig.3.7 Then $g^d_{k_0}$:

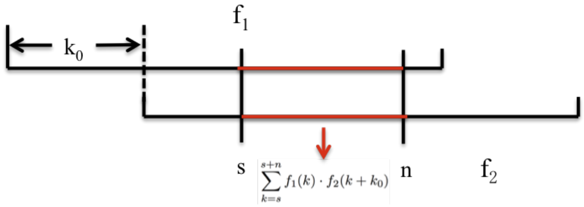


Figure 3.7: New Defined Convolution

$$g^d_{k_0} = \sum_{k'=0}^{N_f-1} d_{k'-k_0} \cdot h_{k'} \tag{3.47}$$

$$= \sum_{k'=0}^{N_f-1} D^d_{\lfloor (k'-k_0)/N_c \rfloor} P^d_{(k'-k_0)\%N_c} \cdot h_{k'} \tag{3.48}$$

$$= \sum_{k'=0}^{N_f-1} D^d_{\lfloor (k'-z\cdot N_s \cdot N_c)/N_c \rfloor} P^d_{(k'-z\cdot N_s\cdot N_c)\%N_c} \cdot h_{k'} \tag{3.49}$$

$$= \sum_{k'=0}^{N_f-1} D^d_{\lfloor k'/N_c \rfloor - z\cdot N_s} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c \rfloor} P^t_{k'\%N_c} \tag{3.50}$$

For data symbols, we could only assume that

$$D^d_k = \begin{cases} +1 & p = 0.5 \\ -1 & p = 0.5 \end{cases} \tag{3.51}$$

Because the D^d_k here is a random variable, we could only exam the mean and variance for it, so do the $g^d_{k_0}$

$$\mathbb{E}\{D^d_k\} = 1 \times 0.5 + (-1) \times 0.5 = 0 \tag{3.52}$$

$$\mathbb{E}\{D^d_k)^2\} = 1 \times 0.5 + (-1)^2 \times 0.5 = 1 \tag{3.53}$$

and they are IID.

For $g^d_{k_0}$,

$$\mathbb{E}\{g^d_{k_0}\} = \mathbb{E}\{\sum_{k'=0}^{N_f-1} D^d_{\lfloor k'/N_c \rfloor - z\cdot N_s} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c \rfloor} P^t_{k'\%N_c}\} \tag{3.54}$$

$$= \sum_{k'=0}^{N_f-1} \mathbb{E}\{D^d_{\lfloor k'/N_c \rfloor - z\cdot N_s} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c \rfloor} P^t_{k'\%N_c}\} \tag{3.55}$$

$$= \sum_{k'=0}^{N_f-1} P^d_{k'\%N_c} P^t_{k'\%N_c} \mathbb{E}\{D^d_{\lfloor k'/N_c \rfloor - z\cdot N_s} D^t_{\lfloor k'/N_c \rfloor}\} \tag{3.56}$$

$$= C_0^{N_c-1}(P^d_k, P^t_k, 0) \sum_{k''=\lfloor k'/N_c \rfloor}^{\lfloor k'/N_c \rfloor + N_s - 1} D^t_{k''} \mathbb{E}\{D^d_{k''-z}\} \tag{3.57}$$

$$= 0 \tag{3.58}$$

$$\mathbb{E}\{(g^d_{k_0})^2\} = \mathbb{E}\{\sum_{k'=0}^{N_f-1} D^d_{\lfloor k'/N_c\rfloor - z\cdot N_s} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c\rfloor \% N_s} P^t_{k\%N_c}\}^2 \tag{3.59}$$

$$= \mathbb{E}\{\sum_{k''=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} D^t_{k''} D^d_{k''-z} \cdot \sum_{m=0}^{N_c} P^d_{m\%N_c} P^t_{m\%N_c}\}^2 \tag{3.60}$$

$$= \big(\sum_{m=0}^{N_c} P^d_m P^t_m\big)^2 \mathbb{E}\{\sum_{k''=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} D^t_{k''} D^d_{k''-z}\}^2 \tag{3.61}$$

$$= (C_0^{N_c-1}(P^d_k, P^t_k, 0))^2 \mathbb{E}\{\sum_{k''=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} D^t_{k''} D^d_{k''}\}^2 \tag{3.62}$$

$$\begin{aligned} &= (C_0^{N_c-1}(P^d_k, P^t_k, 0))^2 \Bigg(\mathbb{E}\{\sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} (D^t_m D^d_m)^2\} \\ &+ \mathbb{E}\{\sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} \sum_{n\neq m} D^t_m D^d_m D^t_n D^d_n\}\Bigg) \end{aligned} \tag{3.63}$$

$$\begin{aligned} &= (C_0^{N_c-1}(P^d_k, P^t_k, 0))^2 \Bigg(\sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} \mathbb{E}\{D^d_m\}^2 \\ &+ \sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} \sum_{n\neq m} D^t_m D^t_n \mathbb{E}\{D^d_m\}\mathbb{E}\{D^d_n\}\Bigg) \end{aligned} \tag{3.64}$$

$$= (C_0^{N_c-1}(P^d_k, P^t_k, 0))^2 \Bigg(\sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} \cdot 1\Bigg) \tag{3.65}$$

$$= (C_0^{N_c-1}(P^d_k, P^t_k, 0))^2 \cdot N_s \tag{3.66}$$

3.3.1.2.2 Chip-level Aligned Chip-level Aligned means the start chip of received symbol is aligned with patten's symbol's first chip. But, the begining symbol of a frame for received data may not be aligned with patten's first symbol. Here is the diagram

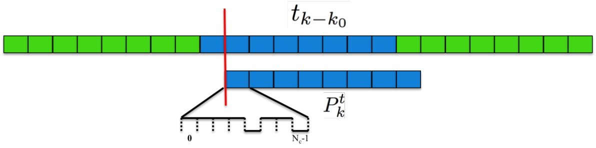


Figure 3.8: Chip-level Aligned

At this time

$$\begin{cases} \alpha = z & z \in \mathbb{Z} \\ \beta = 0 \end{cases} \Rightarrow k_0 = z \cdot N_c \tag{3.67}$$

For $g^t_{k_0}$:

$$g^t_{k_0} = \sum_{k'=0}^{N_f-1} t_{k'-k_0} \cdot h_{k'} \tag{3.68}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{\lfloor (k'-k_0)/N_c \rfloor \% N_s} P^t_{k\%N_c} \cdot h_{k'} \tag{3.69}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{\lfloor (k'-z\cdot N_c)/N_c \rfloor \% N_s} P^t_{k\%N_c} \cdot D^t_{\lfloor k'/N_c \rfloor} P^t_{k\%N_c} \tag{3.70}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{(\lfloor k'/N_c \rfloor - z)\% N_s} D^t_{\lfloor k'/N_c \rfloor} \underbrace{P^t_{k\%N_c} P^t_{k\%N_c}}_{\equiv 1} \tag{3.71}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{(\lfloor k'/N_c \rfloor - z)\% N_s} D^t_{\lfloor k'/N_c \rfloor} \tag{3.72}$$

$$= C_z^{N_s-1}(D^t_k, D^t_k, z) + C_{N_s-z}^{N_s-1}(D^t_k, D^t_k, N_s - z) \tag{3.73}$$

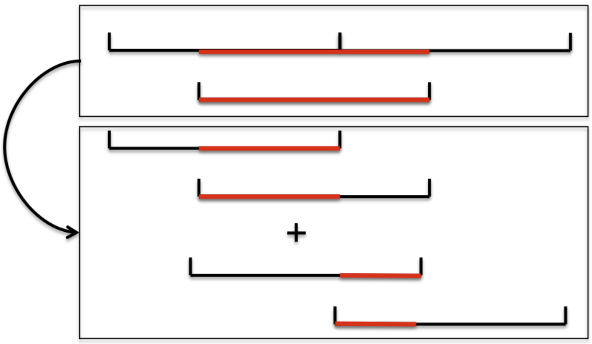


Figure 3.9: Diagram for Misaligned Convolution

For $g^d_{k_0}$:

$$\mathbb{E}\{g^d_{k_0}\} = \mathbb{E}\{\sum_{k'=0}^{N_f-1} D^d_{\lfloor k'/N_c \rfloor - z} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c \rfloor \% N_s} P^t_{k\%N_c}\} \tag{3.74}$$

$$= \sum_{k'=0}^{N_f-1} \mathbb{E}\{D^d_{\lfloor k'/N_c \rfloor - z} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c \rfloor \% N_s} P^t_{k\%N_c}\} \tag{3.75}$$

$$= \sum_{m=0}^{N_c} (P^d_m P^t_m) \sum_{m=\lfloor k'/N_c \rfloor}^{\lfloor k'/N_c \rfloor + N_s - 1} D^t_m \mathbb{E}\{D^d_{m-z}\} \tag{3.76}$$

$$= C_0^{N_c}(P^d_k, P^t_k, 0) \cdot \sum_{m=\lfloor k'/N_c \rfloor}^{\lfloor k'/N_c \rfloor + N_s - 1} D^t_m \cdot 0 \tag{3.77}$$

$$= 0 \tag{3.78}$$

$$\mathbb{E}\{(g^d_{k_0})^2\} = \mathbb{E}\{\sum_{k'=0}^{N_f-1} D^d_{\lfloor k'/N_c\rfloor - z} P^d_{k'\%N_c} \cdot D^t_{\lfloor k'/N_c\rfloor\%N_s} P^t_{k\%N_c}\}^2 \tag{3.79}$$

$$= (C_0^{N_c}(P^d_k, P^t_k, 0))^2 \mathbb{E}\{\sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} D^t_{m-z} D^d_m\}^2 \tag{3.80}$$

$$= (C_0^{N_c}(P^d_k, P^t_k, 0))^2 \mathbb{E}\{\sum_{m=\lfloor k'/N_c\rfloor}^{\lfloor k'/N_c\rfloor + N_s - 1} D^t_m D^d_m\}^2 \tag{3.81}$$

$$= (C_0^{N_c-1}(P^d_k, P^t_k, 0))^2 \cdot N_s \tag{3.82}$$

The mean and variance of $g^d_{k_0}$ is the same as the situation of perfectly aligned.

3.3.1.2.3 Totally Misaligned Totally Misaligned means the first chip of received data is not correspond to the first chip of patten. Like the Fig. 3.10

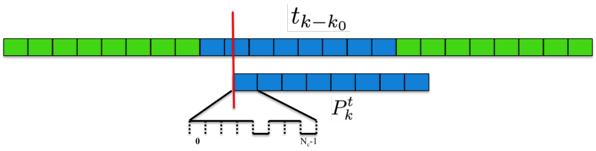


Figure 3.10: Totally Misaligned

$$k_0 = z, \ \ z \in \mathbb{Z} \tag{3.83}$$

For $g^t_{k_0}$:

$$g^t_{k_0} = \sum_{k'=0}^{N_f-1} t_{k'-k_0} \cdot h_{k'} \tag{3.84}$$

$$= \sum_{k'=0}^{N_f-1} D^t_{\lfloor (k'-z)/N_c\rfloor\%N_s} P^t_{(k'-z)\%N_c} \cdot D^t_{\lfloor k'/N_c\rfloor} P^t_{k\%N_c} \tag{3.85}$$

We will first deal with a smaller problem. If we only focus on patten of length N_c Define the result of the above situation to be $f^t(s_1, s_2, t_1, z)$

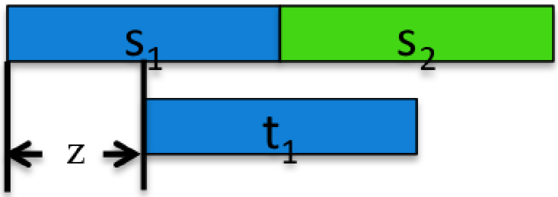


Figure 3.11: Chip-level Misaligned

$$f^t(s_1, s_2, t_1, z) = (s_1 \cdot t_1) C_z^{N_c-1}(P^t_k, P^t_k, z) + (s_2 \cdot t_1) C_{N_c-z}^{N_c-1}(P^t_k, P^t_k, N_c - z) \tag{3.86}$$

Now, back to $g^t_{k_0}$

$$g^t_{k_0} = \sum_{m=\lfloor(-k_0)/N_c\rfloor,n=0}^{\lfloor m=(-k_0)/N_c\rfloor+N_s,n=N_s} f^t_{m\%N_s,m+1\%N_s,n,z\%N_c} \tag{3.87}$$

For $g^d_{k_0}$

$$\mathbb{E}\{g^d_{k_0}\} = \sum_{k'=0}^{N_f-1} D^d_{\lfloor(k'-z)/N_c\rfloor\%N_s} P^d_{(k'-z)\%N_c} \cdot D^t_{\lfloor k'/N_c\rfloor} P^t_{k\%N_c} \tag{3.88}$$

We can also define

$$f^d_1(s_1, s_2, t_1, z) = (s_1 \cdot t_1) C^{N_c-1}_z (P^d_k, P^t_k, z) + (s_2 \cdot t_1) C^{N_c-1}_{N_c-z}(P^d_k, P^t_k, N_c - z) \tag{3.89}$$

Then $g^d_{k_0}$ would be

$$g^d_{k_0} = \sum_{m=\lfloor(-k_0)/N_c\rfloor,n=0}^{\lfloor m=(-k_0)/N_c\rfloor+N_s,n=N_s} f^d(m, m+1, n, z\%N_c) \tag{3.90}$$

But for the expectation, we may need to change a little bit. Because we want to split the patten symbol rather than the received symbol.

$$g^d_{k_0} = \sum_{m=\lfloor(-k_0)/N_c\rfloor,n=0}^{\lfloor m=(-k_0)/N_c\rfloor+N_s,n=N_s} f^d(n, n+1, m, z\%N_c) \tag{3.91}$$

Then

$$\mathbb{E}\{g^d_{k_0}\} = \mathbb{E}\{\sum_{m=\lfloor(-k_0)/N_c\rfloor,n=0}^{\lfloor m=(-k_0)/N_c\rfloor+N_s,n=N_s} f^d(n, n+1, m, z\%N_c)\} \tag{3.92}$$

$$= \sum_{m=\lfloor(-k_0)/N_c\rfloor,n=0}^{\lfloor m=(-k_0)/N_c\rfloor+N_s,n=N_s} \mathbb{E}\{f^d(n, n+1, m, z\%N_c)\} \tag{3.93}$$

$$= 0 \tag{3.94}$$

$$\mathbb{E}\{(g^d_{k_0})^2\} = \left(C^{N_c-1}_{z\%N_c}(P^d_k, P^t_k, z\%N_c) + C^{z-1}_0(P^d_k, P^t_k, -z\%N_c)\right)^2 \cdot N_s \tag{3.95}$$

Because we can view it as a shifted training patten.

3.3.1.3 Ideal Situation

The formulas above are for practical situations. For idea situations, we would like to let spread codes have properities like below:

1. Different spread codes are orthogonal;

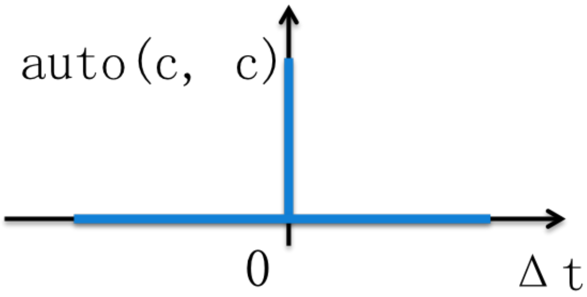


Figure 3.12: Auto-correlation for certain spread codes

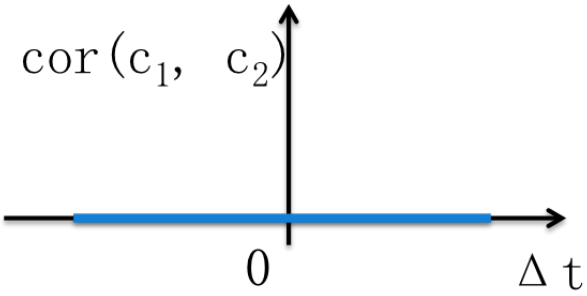


Figure 3.13: Cross-correlation for different spread codes

2. Autocorrelation is a δ function.

So the auto-correlation and cross-correlation arelike:
The mathematical description for this property would be:

for different spread codes $C_i(t)$and$C_j(t), i \neq j$means they are different.

$$c_i(t) * c_i(-t + \Delta t) = \begin{cases} N_c & \Delta t = k \cdot N_c, k = \mathbb{Z} \\ 0 & else \end{cases} \tag{3.96}$$

$$c_i(t) * c_j(-t + \Delta t) = 0, \forall k \in \mathbb{Z} \tag{3.97}$$

Compared the ideal situation with the analysis did in subsection 3.3.1.2, because the spread code for training and data are orthogonal, or say different, so

$$g^d_{k_0} \equiv 0 \tag{3.98}$$

Because the auto-correlation function has perfect property, eq 3.3.1.2.2 and eq 3.3.1.2.3 should be as close to 0 as possible.

Bibliography

[1] GNURadio. Gnuradio, 2016.

[2] Uncertainty principle. Uncertainty principlen, 2016.