Abstract.

Polyphase codes derived from linear-frequency-modulation (LFM), such as P3 code and P4 code, are a useful class of pulse compression waveforms. The general Polyphase codes such as Frank code, P1 code , P2 code, P3 code and Bi-phased Barker code is discussed along with their peak side-lobe level (PSL) and Integrated Sidelobe level(ISL). This thesis also discusses and compares the pulse compression performance including peak side-lobe level (PSL) and signal-to-noise ratio (SNR) loss of P4 code under different side-lobe suppression methods. Different windowing techniques such as Rectangular window , Hann Window, Hamming window, Kaiser window for different ß values are discussed and analysed using P4 code. Two new methods including mismatch filter design based on second-order cone programming and optimal design method combined arbitrary phase codes with mismatch filter optimization are introduced. It was shown that, the mismatch filter based on second order cone programming result in low PSL and very low SNR loss, the higher the mismatch filter order is, the lower the PSL we can get. The design method combined arbitrary phase codes with mismatch filters optimization result in the lowest PSL, the lowest SNR loss and the highest range resolution in these methods.

Keywords

Radar Pulse Compression, PSL.ISL, SNR.

1.INTRODUCTION

RADAR is an acronym of Radio Detection And Ranging. It is an object-detection system which uses electromagnetic waves specifically radio waves to determine the range, altitude, direction or speed of both moving and fixed objects such as aircraft ,ships, spacecraft ,guided missiles ,motor vehicles, weather formations, and terrain. The radar dish, or antenna, transmits pulses of radio waves or microwaves which bounce off any object in their path. The object returns a tiny part of the wave's energy to a dish or antenna which is usually located at the same site as the transmitter. The modern uses of radar are highly diverse, including air traffic control, radar astronomy, and aircraft anti-collision systems, antimissile.

The rapid advances in digital technology made many theoretical capabilities practical with digital signal processing and digital data processing. Radar signal processing is defined as the manipulation of the received signal, represented in digital format, to extract the desired information whilst rejecting unwanted signals. Pulse compression allowed the use of long waveforms to obtain high energy simultaneously achieves the resolution of a short pulse by internal modulation of the long pulse. The resolution is the ability of radar to distinguish targets that are closely spaced together in either range or bearing. The internal modulation may be binary phase coding, polyphase coding, frequency modulation, and frequency stepping. There are many advantages of using pulse compression techniques in the radar field. They include reduction of peak power, relevant reduction of high voltages in radar transmitter, protection against detection by radar detectors, significant improvement of range resolution, relevant reduction in clutter troubles and protection against jamming coming from spread spectrum action .

In pulse compression technique, the transmitted signal is frequency or phase modulated and the received signal is processed using a specific filter called "matched filter". In this form of pulse compression, a long pulse of duration T is divided into N sub pulses each of width τ. The phase of each sub-pulse is chosen to be either 0 or π radians. A matched filter is a linear network that maximizes the output peak-signal to noise ratio of a radar receiver which in turn maximizes the detectability of a target. In 1950-60, the practical realization of radars using pulse compression has taken place. At the starting, the realization of matched filters was difficult using traverse filters because of lack of delay line with enough bandwidth. Later matched filters have been realized by using dispersive networks made with lumped-constant filters. In recent years, instead of matched filters, many sophisticated filters are in use.

The binary choice of 0 or π phase for each sub-pulse may be made at random. However, some random selections may be better suited than others for radar application. One criterion for the selection of a good “random” phase-coded waveform is that its autocorrelation function should have equal time side-lobes.Barker codes have called perfect codes because the highest sidelobe is only one code element amplitude high.However,the largest pulse compression ratio that can be obtained with barker code is only 13.

The codes that use any harmonically related phases on certain fundamental phase increments are called polyphase codes. Frank proposed a polyphase code called as Frank code which is more Doppler tolerant and has lower sidelobes than binary codes. Krestschmer and Lewis have presented the variants of Frank code. P1 code which is derived from step frequency, Bolter matrix derived P2 code and linear frequency derived P3 and P4 codes. The significant advantage of P1 and P2 codes over the Frank code and the P4 code over P3 is that they are tolerant to receiver band limitations.

1.2. Motivation

The pulse compression in radar has major applications in the recent years. For better pulse compression, peak signal to sidelobe ratio should be as high as possible so that the unwanted clutter gets suppressed and should be very tolerant under Doppler shift conditions. Many pulse compression techniques have come into existence including neural networks. Most of the waveforms used for pulse compression generate random, noise-like sidelobe patterns, which make them hardly practical for sidelobe cancellation, the uniform sidelobe patterns of the Woo filter are promising for such a scheme.Conventional sidelobe reduction techniques have suffered from performance degradation and SNR gains so asymmetrical weighting receivers can be use in such cases because they keep the sidelobe levels uniformly flat over all time delays while the range resolution loss is prevented . The study of polyphase codes and their sidelobe reduction techniques are carried out since the polyphase codes have low sidelobes and are better Doppler tolerant and better tolerant to pre- compression bandlimiting.

1.3. Thesis Organization

Module-1 Introduction

Module-2 Polyphase Codes

This chapter deals with the introduction of Bi-phase codes and their limitations and different polyphase codes such as Frank, P1, P2, P3, P4 and complementary codes. The study of these codes, properties and their advantages over bi-phase codes is described

Module-3 Conventional Sidelobe Reduction techniques for polyphase codes

This chapter deals with the different Conventional sidelobe reduction techniques such as Amplitude weighting using different windows, and an optimal technique for uniform range sidelobe and reduction of ISL. The study of these techniques and their properties are carried out.

Module-4 Sidelobe suppression Using second Order Cone Programming

This chapter describes the Optimisation filter concept of sidelobe reduction, the advanced versions of Second order Cone Programming Optimzation filters and a proposed pulse compression technique is described.

Module-5 Conclusion and scope for future work

The concluding remarks for all the chapters are presented in this chapter. It also contains some future research topics which need attention and further investigation**.**

Appendix: Matlab code to simulate the results

Module 2

Polyphase Codes

2.1. Introduction

Radar is an electromagnetic system for detection and location of objects such as aircraft, ships, spacecraft, vehicles, people and natural environment [2.1]. It operates by radiating energy into space and detecting the echo signal reflected from object or target. The reflected energy that is returned to the radar not only indicates the presence of the target, but by comparing the received echo signal with the signal that was transmitted, its location can be determined along with other target-related information.

The basic principle of radar is simple. A transmitter generates an electro-magnetic signal (such as a short pulse of sine wave) that is radiated into space by an antenna. A portion of the transmitted signal is intercepted by a reflecting object (target) and is re-radiated in all directions. It is the energy re-radiated in back direction that is of prime interest to the radar. The receiving antenna collects the returned energy and delivers it to a receiver, where it is processed to detect the presence of the target and to extract its location and relative velocity. The distance to the target is determined by measuring the time taken for the radar signal to travel to the target and back. The range is

$$R=CT_R/2 \ ; \tag{2.1}$$

Where T_R is the time taken by the pulse to travel to target and return, c is the speed of propagation of electromagnetic energy. Radar provides the good range resolution as well as long detection of the target.

The most common radar signal or waveform is a series of short duration, somewhat rectangular-shaped pulses modulating a sine wave carrier. Short pulses are better for range resolution, but contradict with energy, long range detection, carrier frequency and SNR. Long pulses are better for signal reception, but contradict with range resolution and minimum range. At the transmitter, the signal has relatively small amplitude for ease to generate and is large in time to ensure enough energy in the signal. At the receiver, the signal has very high amplitude to be detected and is small in time .

A very long pulse is needed for some long-range radar to achieve sufficient energy to detect small targets at long range. But long pulse has poor resolution in the range dimension. Frequency or phase modulation can be used to increase the spectral width of a long pulse to obtain the resolution of a short pulse. This is called “pulse compression”.

2.2. Pulse Compression

The term radar signal processing incorporates the choice of transmitting waveforms for various radars, detection theory, performance evaluation, and the circuitry between the antenna and the displays or data processing computers. The relationship of signal processing to radar design is analogous to modulation theory in communication systems. Both fields continually emphasize communicating a maximum of information in a special bandwidth and minimizing the effects of interference.

Although the transmitted peak power was already in megawatts, the peak power continued to increase more and more due to the need of longer range detection. Besides the technical limitation associated with it, this power increase poses a financial burden. Not only that, target resolution and accuracy became unacceptable. Siebert and others pointed out the detection range for given radar and target was dependent only on the ratio of the received signal energy to noise power spectral density and was independent of the waveform. The efforts at most radar laboratories then switched from attempts to construct higher power transmitters to attempts to use pulses that were of longer duration than the range resolution and accuracy requirements would allow.

Increasing the duration of the transmitted waveform results in increase of the average transmitted power and shortening the pulse width results in greater range resolution. Pulse compression is a method that combines the best of both techniques by transmitting a long coded pulse and processing the received echo to get a shorter pulse.

The transmitted pulse is modulated by using frequency modulation or phase coding in order to get large time-bandwidth product. Phase modulation is the widely used technique in radar systems. In this technique, a form of phase modulation is superimposed to the long pulse increasing its bandwidth. This modulation allows discriminating between two pulses even if they are partially overlapped. Then upon receiving an echo, the received signal is compressed through a filter and the output signal will look like the one. It consists of a peak component and some sidelobes.

2.3. Matched filter

A matched filter is a linear network that maximises the output peak-signal to noise (power) ratio of a radar receiver which in turn maximizes the detectability of a target. It is obtained by correlating a known signal, or a template, with an unknown signal to detect the presence of the template in the unknown signal. This is equivalent to convolving the unknown signal with a conjugated time-reversed version of the template. It is the optimal linear filter for maximizing the signal to noise ratio (SNR) in the presence of additive stochastic noise. It has a frequency response function which is proportional to the complex conjugate of the signal spectrum.

$$H(f) = G_a S^*(f) \exp(-j2\pi f t_m) \qquad (2.2)$$

Where G_a is a constant, t_m is the time at which the output of the matched filter is a maximum (generally equal to the duration of the signal), and S*(f) is the complex conjugate of the spectrum of the (received) input signal s(t), found from the Fourier transform of the received signal s(t) such that

$$S(f) = \int_{-\infty}^{\infty} S(t)exp(-j2\pi ft)dt \qquad (2.3)$$

A matched filter for a transmitting a rectangular shaped pulse is usually characterized by a bandwidth B approximately the reciprocal of the pulse with τ or $B\tau \approx 1$. The output of a matched filter receiver is the cross-correlation between the received waveform and a replica of the transmitted waveform [2.5]

2.4. Phase coded pulse compression

In this form of pulse compression, a long pulse of duration T is divided into N sub- pulses each of width τ as shown in Figure 2.2. An increase in bandwidth is achieved by changing the phase of each sub-pulse. The phase of each sub-pulse is chosen to be either 0 or π radians or they can be harmonically related. The output of the matched filter will be a spike of width τ with an amplitude N times greater than that of long pulse. The pulse compression ratio is $N = T/\tau \approx BT$, where $B \approx 1/\tau$ = bandwidth. The output waveform extends a distance T to either side of the peak response, or central spike. The portions of the output waveform other than the spike are called time side-lobes. Phase coding can be either binary phase coding or polyphase coding.

2.4.1. Binary phase codes

The binary choice of 0 or π phase for each sub-pulse may be made at random. However, some random selections may be better suited than others for radar application. One criterion for the selection of a good “random” phase-coded waveform is that its autocorrelation function should have equal time sidelobes [2.1]. The binary phase-coded sequence of 0, π values that result in equal side-lobes after passes through the matched filter is called a Barker code. An example is shown in Figure 2(a). This is a Barker code of length 13. The (+) indicates 0 phase and (−) indicates π radians phase. The auto-correlation function, or output of the matched filter, is shown in Figure 2(b). There are six equal time side-lobes to either side of the peak, each of label 22.3 dB below the peak. The longest Barker code length is 13. The barker codes are listed in Table 2.1. When a larger pulse-compression ratio is desired, some form of pseudo random code is usually used. To achieve high range resolution with-out an incredibly high peak power, one needs pulse compression.

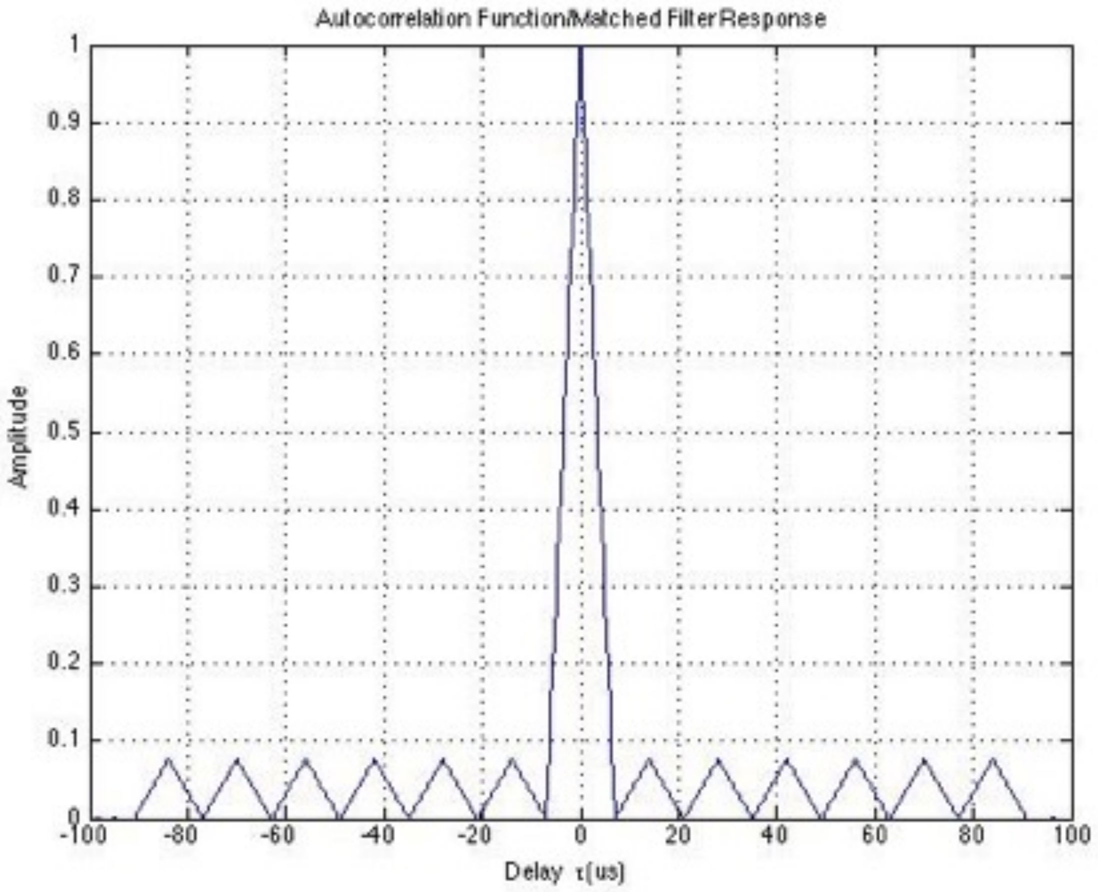


Fig 2.1 Barker Code Autocorrelation Output

Barker codes have been called the perfect codes because the highest sidelobe is only one code element amplitude high. However, the largest pulse compression ratio that can be obtained with the barker codes only . The sidelobe levels obtained with the polyphase codes are not limited to any finite pulse compression ratio and exhibit better Doppler tolerance for broad range-Doppler coverage than do the biphase codes, and they exhibit relatively good sidelobe characteristics.

2.5 Polyphase codes

The codes that use any harmonically related phases based on a certain fundamental phase increment are called Polyphase codes and these codes are derived conceptually coherently detecting a frequency modulation pulse compression waveform with either a local oscillator at the band edge of the waveform (single side band detection) or at band center (double sideband detection) and by sampling the resultant inphase I and Q data at the Nyquist rate. The Nyquist rate in this case is once per cycle per second of the bandwidth of the waveform [2.8].

Frank proposed a polyphase code with good non-periodic correlation properties and named the code as Frank code [2.9]. Kretscher and Lewis proposed different variants of Frank polyphase codes called p-codes which are more tolerant than Frank codes to receiver bandlimiting prior to pulse compression [2.10, 2.11]. Lewis has proven that the sidelobes of polyphase codes can be substantially reduced after reception by following the autocorrelation with two sample sliding window subtractor for Frank and P1 codes and for P3 and P4 codes.

Polyphase compression codes have been derived from step approximation to linear frequency modulation waveforms (Frank, P1, P2) and linear frequency modulation waveforms (P3, P4).

These codes are derived by dividing the waveform into subcodes of equal duration, and using phase value for each subcode that best matches the overall phase trajectory of the underlying waveform. In this section the polyphase codes namely Frank, P1, P2, P3, P4 codes and their properties are described.

2.5.1. Frank Code

The Frank code is derived from a step approximation to a linear frequency modulation waveform using N frequency steps and N samples per frequency [2.9]. Hence the length of Frank code is N^2. The Frank coded waveform consists of a constant amplitude signal whose carrier frequency is modulated by the phases of the Frank code.

The phases of the Frank code is obtained by multiplying the elements of the matrix A by phase (2π/N) and by transmitting the phases of row1 followed by row 2 and so on.

$$A = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 2 & \dots & (N-1) \\ 0 & 2 & 4 & \dots & 2(N-1) \\ . & 3 & 6 & \dots & 3(N-1) \\ . & . & . & \dots & . \\ 0 & (N-1) & 2(N-1) & \dots & (N-1)^2 \end{pmatrix} \qquad 2.4$$

The phase of the *i*th code element in the *j*th row of code group is computed as

$$\Phi_{i,j} = \left(\frac{2\pi}{N}\right)(i-1)(j-1) \qquad 2.5$$

Where *i* and *j* ranges from 1 to N. For example, the Frank code with N = 4, by taking phase value modulo 2 π is given by the sequence,

$$\phi_{4\times4} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{\pi}{2} & \pi & \frac{3\pi}{2} \\ 0 & \pi & 0 & \pi \\ 0 & \frac{3\pi}{2} & \pi & \frac{\pi}{2} \end{pmatrix} \qquad 2.6$$

The autocorrelation function under zero Doppler and the phase values of Frank code with length 100 are given in Figure 2.2 .

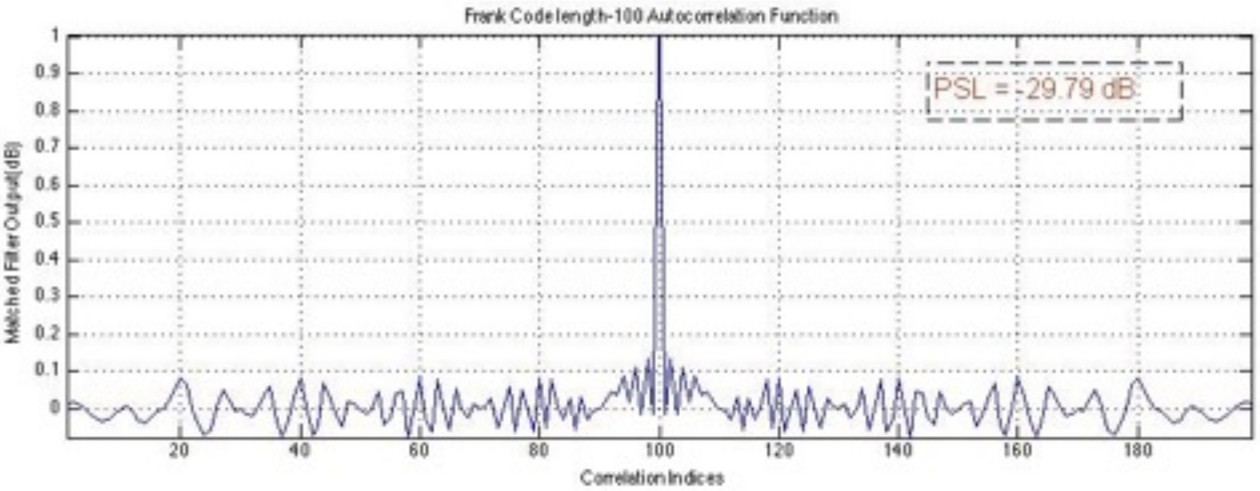


Fig 2.2a

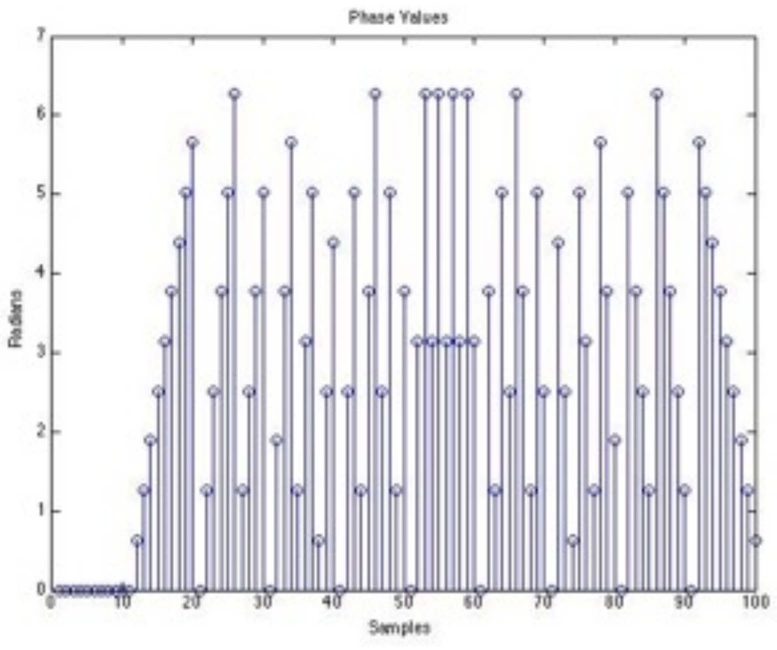


Fig 2.2b

(Figure 2.2) Frank Code for length 100 (a) Autocorrelation under zero Doppler shift (b) phase plot

From the above figure it is evident that the Frank code has the largest phase increments from sample to sample in the centre of the code. Hence, when the code is passed through a bandpass amplifier in a radar receiver, the code is attenuated more in the centre of the waveform. This attenuation tends to increase the sidelobes of the Frank code ACF. Hence it is very intolerant to precompression bandlimiting. But comparing with binary phase codes, the Frank code has a peak sidelobe level (PSL) ratio of -29.79dB which is approximately 10 dB better than the best pseudorandom codes.

In the presence of Doppler shift, the autocorrelation function of Frank codes degrades at much slower rate than that for binary codes, however the peak shifts in position rapidly and a range error occurs due to this shift. The correlation under Doppler frequency f_d is obtained by correlating the transmitted one with received one multiplied by $exp(-j2\pi ft)$, where T is the length of the code. The PSL value under Doppler of 0.05 is calculated as -8.42dB.

2.5.2. P1 Code

The P1, P2, P3, P4 codes are obtained by the modified versions of the Frank code, with the dc frequency term in the middle of the pulse instead of at the beginning. P1 code is derived by placing the synchronous oscillators at the centre frequency of the step chirp IF waveform and sampling the baseband waveform at the Nyquist rate.

The P1 code has N^2 elements and the phase of *i*th element of the *j*th group is represented as

$$\Phi_{i,j} = -\left(\frac{\pi}{N}\right)[N-(2j-1)][(j-1)N+(j-1) \qquad 2.7$$

Where the integers *i* and *j* ranges from 1 to N. For example, the P1 code with N = 4, by taking phase value modulo 2π is given by the sequence,

$$\phi_{4x4} = \begin{pmatrix} 0 & \pi & 0 & \pi \\ \frac{5\pi}{4} & \frac{3\pi}{4} & \frac{\pi}{4} & \frac{7\pi}{4} \\ \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} & \frac{\pi}{2} \\ \frac{7\pi}{4} & \frac{\pi}{4} & \frac{3\pi}{4} & \frac{5\pi}{4} \end{pmatrix}$$

The autocorrelation function and the phase values of P1 code with length 100 are given in Figure 2.3. The PSL value is obtained as -23.99dB. P1 code has the highest phase increments from sample to sample at the two ends of the code. Thus, when waveforms phase coded with these codes are passed through band pass amplifiers in a radar receiver, P1 code is attenuated most heavily at the two ends of the waveform. This reduces the sidelobes of the P1 code autocorrelation function. Hence this exhibits relatively low sidelobes than Frank code. This result shows that P1 code is very pre-compression bandwidth tolerant than Frank code.

Also, P1 code has an autocorrelation function magnitude which is identical to the Frank code for zero Doppler shifts.

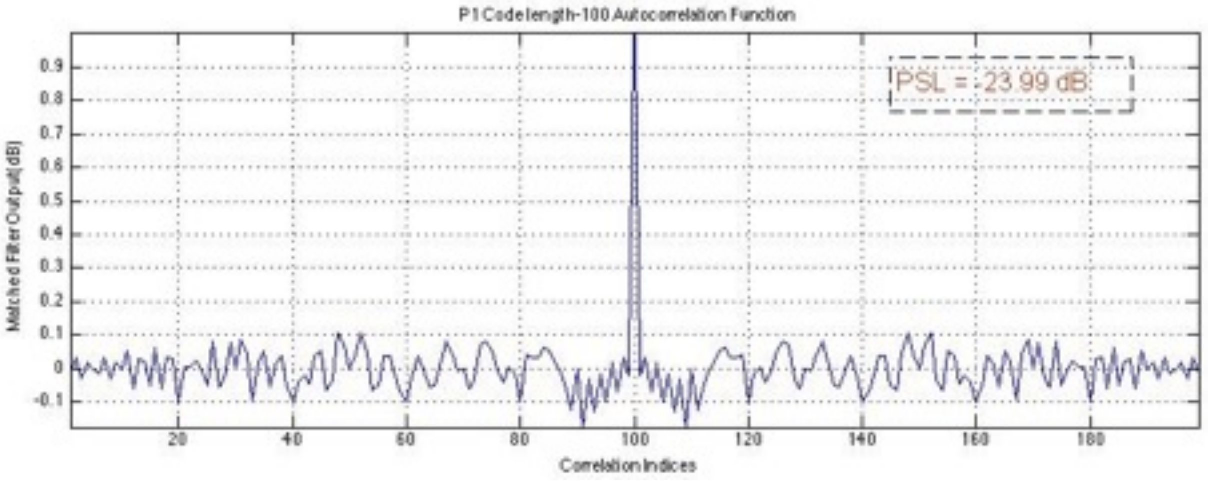


Fig 2.3a

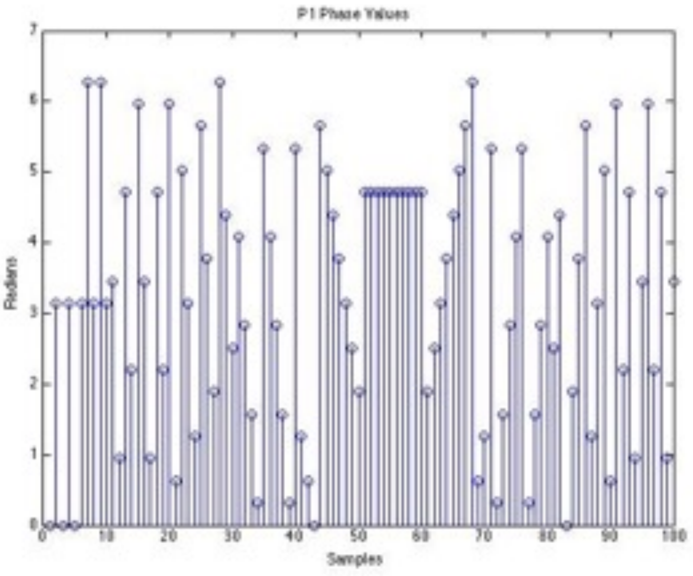


Fig 2.3b

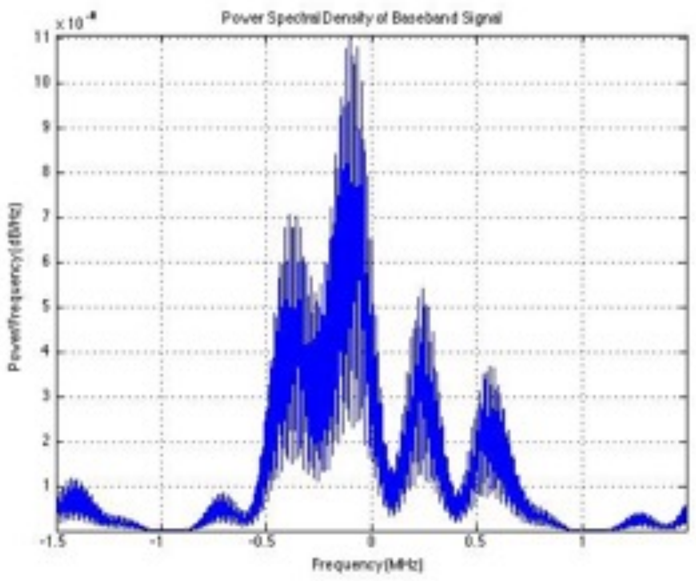


Fig 2.3c

Figure 2.3. P1 Code for length 10 (a) its Autocorrelation (b) its phase and magnitude plots (c) Power Spectrum

2.5.3. P2 Code

The P2 code has the same phase increments within each phase group as the P1 code, except that the starting phases are different . The P2 code has N^2 elements and the phase of *i*th element of the *j*th group is represented as

$$\Phi_{i,j} = -\left(\frac{\pi}{2N}\right)[N-2i+1][N+2j-1] \qquad 2.8$$

Where *i* and *j* are integers ranges from 1 to N.The value of N should be even in order to get low autocorrelation sidelobes. An odd value of N results in high autocorrelation sidelobes.

For example, the P2 code with N = 4, by taking phase value modulo 2 πis given by the sequence,

$$\phi_{4\times4} = \begin{pmatrix} \frac{9\pi}{8} & \frac{3\pi}{8} & \frac{13\pi}{8} & \frac{7\pi}{8} \\ \frac{3\pi}{8} & \frac{\pi}{8} & \frac{13\pi}{8} & \frac{15\pi}{8} \\ \frac{13\pi}{8} & \frac{15\pi}{8} & \frac{\pi}{8} & \frac{3\pi}{8} \\ \frac{7\pi}{8} & \frac{13\pi}{8} & \frac{3\pi}{8} & \frac{9\pi}{8} \end{pmatrix}$$

The autocorrelation function under zero Doppler,and the phase values of P2 code with length 100are given in Figure 2.4.

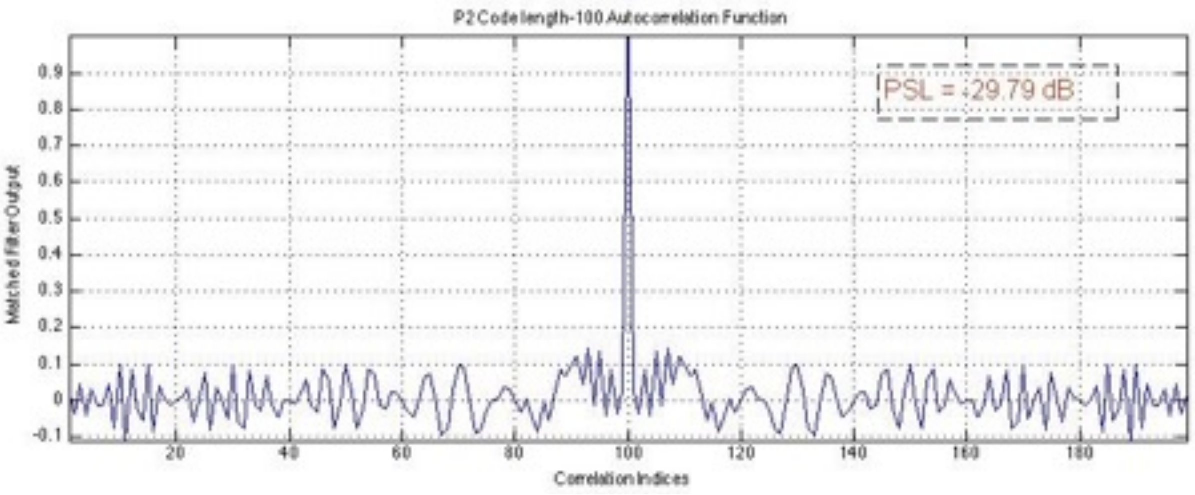


Fig 2.4a

The peak sidelobes of the P2 code are the same as the Frank code for zero Doppler case and the mean square sidelobes of the P2 code are slightly less

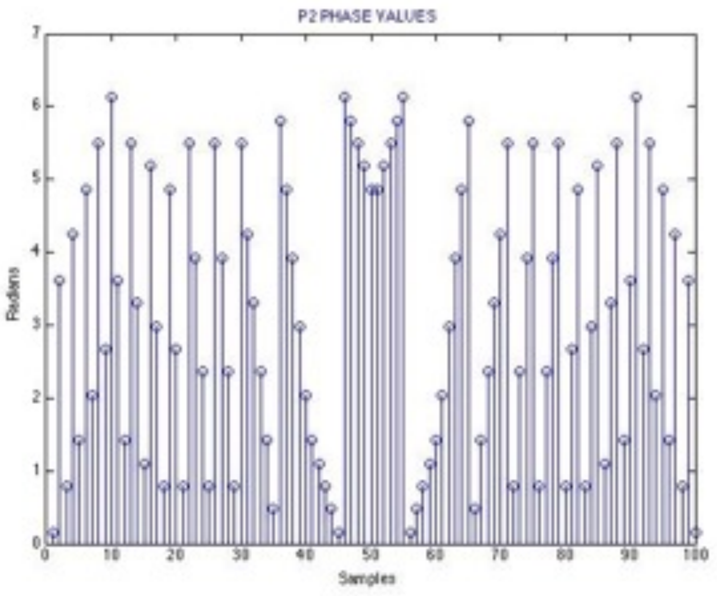


Fig 2.4b

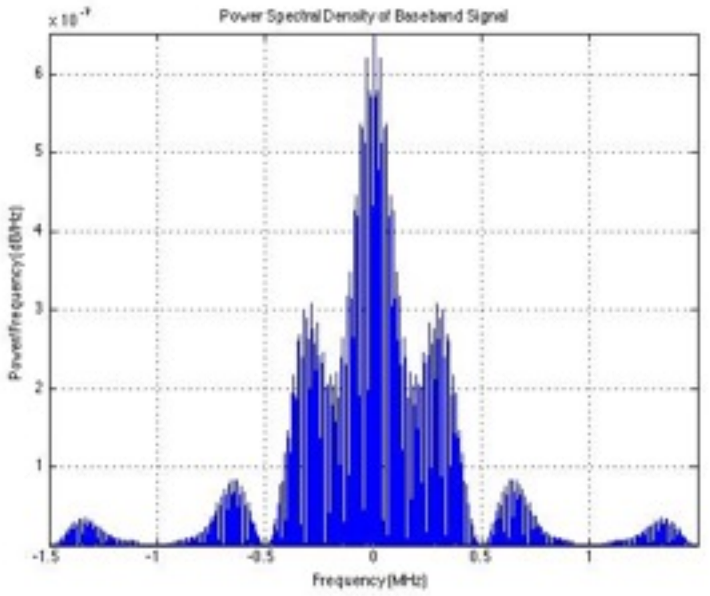


Fig 2.4 c

Fig 2.4(a)Autocorrelation (b) Phase Values of P2 code (c) Power spectrum of P2 code , on length N=100;

The peak sidelobes of the P2 code are the same as the Frank code for zero Doppler case and the mean square sidelobes of the P2 code are slightly less. The value of PSL obtained as -29.79dB which is same as that of Frank code.. The phase changes in P2 code are largest towards the end of the code.

The significant advantage of the P1 and P2 codes over the Frank code is that they are more tolerant of receiver band limiting prior to pulse compression. But P1 and P2 suffers from high PSL value. PSL value is obtained by the ratio of peak sidelobe amplitude to the main lobe amplitude. To obtain low PSL values, we go for P3 and P4 codes.

2.5.4. P3 Code

The P3 code is conceptually derived by converting a linear frequency modulation waveform to baseband using a local oscillator on one end of the frequency sweep and sampling the inphase I and quadrature Q value at the Nyquist rate . Letting the waveform to be coherently detected have a pulse length T and frequency

$$f_o = f + kt \qquad 2.9$$

Where k is a constant, the bandwidth of the signal will be approximately

$$B = kT \qquad 2.10$$

This bandwidth will support a compressed pulse length of approximately

$$t_c = 1/B \qquad 2.11$$

And the waveform will provide a pulse compression ratio of

$$\delta = T/t_c \qquad 2.12$$

Assuming that the first sample of I and Q is taken at the leading edge of waveform, the phases of successive samples taken tc apart is

$$\Phi_i^{p3} = 2\pi \int_0^{(i-1)tc} [(f_c + kT) - f_o]dt \qquad 2.9$$

$$= \pi(i-1)^2 t_c^2$$

Thus the phase sequence of the P3 signal is given by

$$\Phi_i^{p3} = \frac{\pi}{N}(i-1)^2 \qquad 2.10$$

Where i varies from 1 to N and N is the compression ratio. For example, the P3 code with N = 16, by taking phase value modulo 2π is given by the sequence,

$$\Phi_i^{p3} = \left[0 \; \frac{\pi}{16} \; \frac{4\pi}{16} \; \frac{9\pi}{16} \; 0 \; \frac{25\pi}{16} \; \frac{4\pi}{16} \; \frac{17\pi}{16} \; 0 \; \frac{17\pi}{16} \; \frac{4\pi}{16} \; \frac{25\pi}{16} \; \pi \; \frac{9\pi}{16} \; \frac{4\pi}{16} \; \frac{\pi}{16}\right] \qquad 2.11$$

It is thus said to have derived by converting a linear frequency modulation waveform to baseband using a local oscillator on one end of the frequency sweep and sampling the inphase I and quadrature Q value at the Nyquist rate.

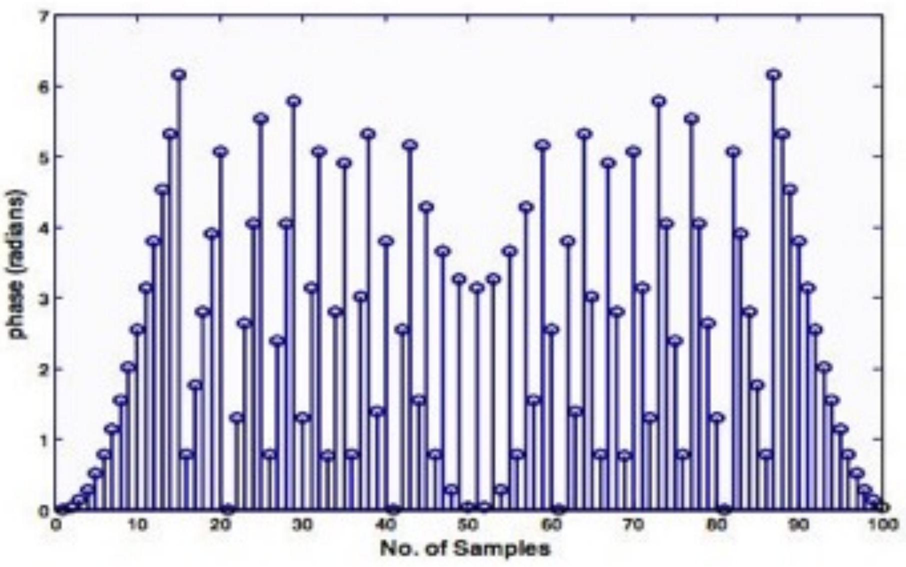


Fig 2.5a

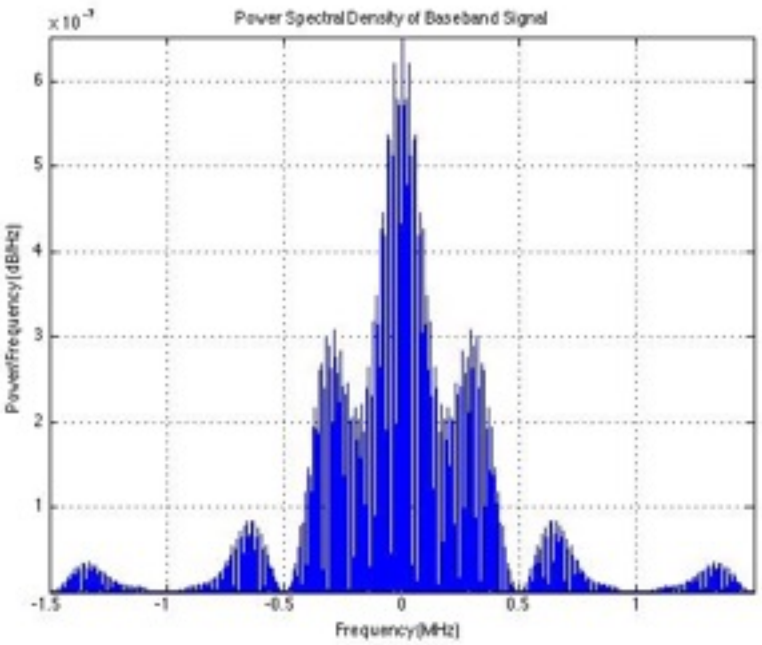


Fig 2.5b

Fig 2.5 (a) P3 phase values (b) Power Spectrum of code length 100

2.5.5. P4 Code

The P4 Code is conceptually derived from the same waveform as the P3 Code[2.10,2.11]. However, in this case, the local oscillator frequency is set equal to *fo=f +kT/2* in the I,Q detectors. With the frequency, the phases of successive samples taken tc apart are tc apart are

$$\phi_i^{P4} = 2\pi \int_0^{(i-1)tc} [(f_0+kt)-(f_0+\frac{kT}{2})]dt \qquad 2.12$$

$$\phi_i^{P4} = 2\pi \int_0^{(i-1)tc} \left(kt-\frac{T}{2}\right) dt$$

$$\phi_i^{P4} = \pi k(i-1)^2tc^2 - \pi kT(i-1)^2tc^2 \qquad 2.13$$

$$= [\pi(i-1)^2/\delta] - \pi(i-1)$$

Thus the phase sequence of the P4 signal is given by:

$$\phi_i = \frac{\pi}{N}(i-1)(i-N-1) \qquad 2.14$$

Where i varies from 1 to N and N is the compression ratio. For example, the P4 code with N = 16, by taking phase value modulo 2 π is given by the sequence,

$$\phi_{16} = \left[0 \;\; \frac{17\pi}{16} \;\; \frac{4\pi}{16} \;\; \frac{25\pi}{16} \;\; \pi \;\; \frac{9\pi}{16} \;\; \frac{4\pi}{16} \;\; \frac{\pi}{16} \;\; 0 \;\; \frac{\pi}{16} \;\; \frac{4\pi}{16} \;\; \frac{9\pi}{16} \;\; \pi \;\; \frac{25\pi}{16} \;\; \frac{4\pi}{16} \;\; \frac{17\pi}{16} \right]$$

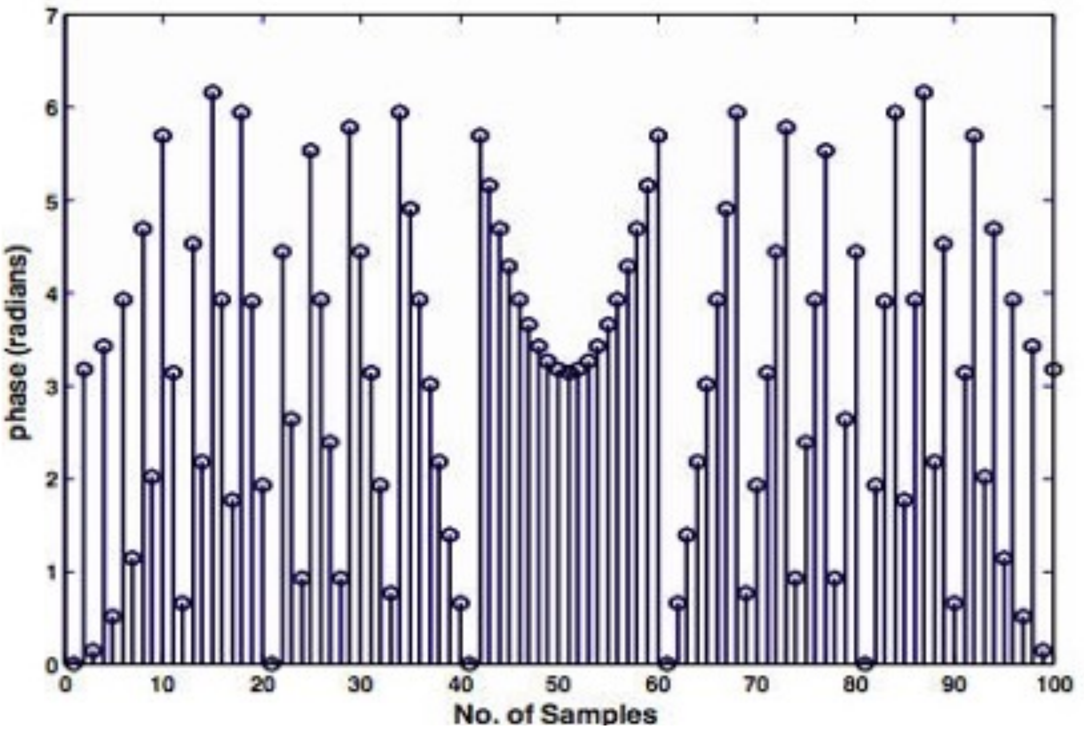


Fig 2.6(a), phase values of P4 code of length 100

The Power spectrum is obtained as shown below for code of length 100. The PSL value is obtained as - 26.32dB under zero Doppler, which is similar to P3 code.

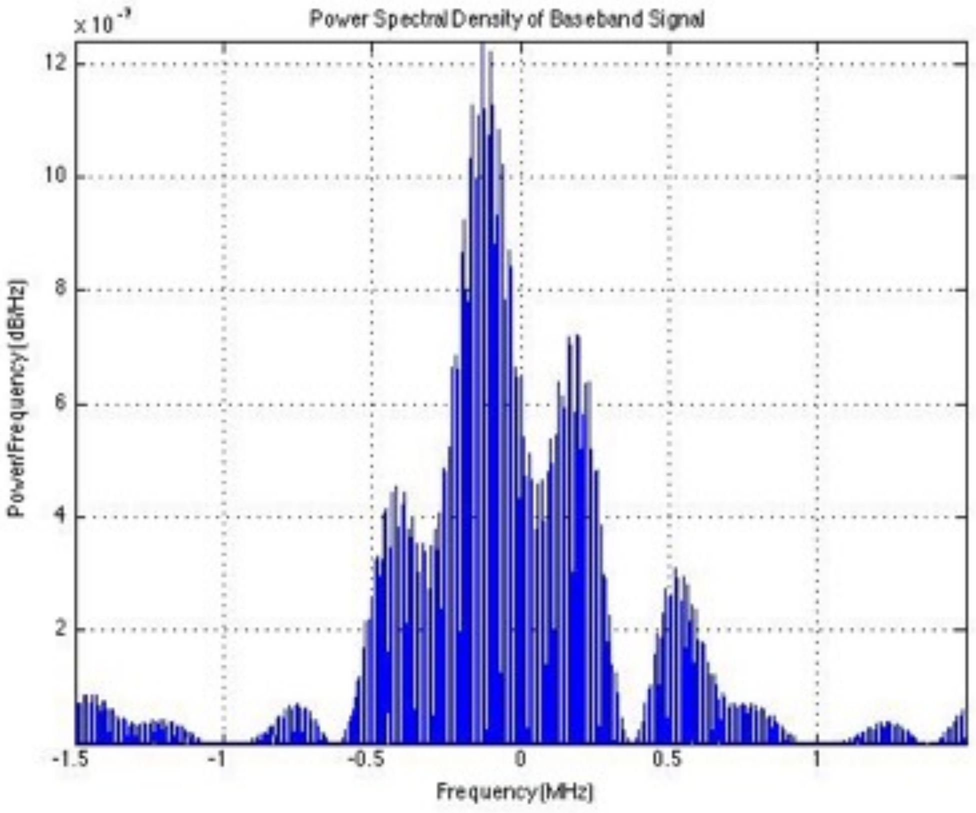


Fig 2.6(b) Power spectrum of P4 code of length 100

Thus the P4 code is more precompression bandwidth limitation tolerant but has same Doppler tolerance than the P3 code. This follows since precompression bandwidth limitations average the code phase increments and would attenuate the P4 code on the ends and the P3 code in the middle. The former increases the peak-to-sidelobe ratio of the compressed pulse while the latter decreases it.

2.6. Summary

In this chapter, Phase coded pulse compression, Binary phase codes, and polyphase codes are described. The performances of polyphase codes namely Frank, P1, P2, P3, P4 codes, their autocorrelation properties, their phase values and their properties under Doppler shift conditions are discussed.

Module 3

Conventional Sidelobe Reduction Techniques for Polyphase Codes

3.1 Range Time Sidelobes

Sidelobes are unwanted because a time sidelobe of a strong echo may hide a weaker target return. Also any clutter present in the sidelobes can leak into range of interest. Most of the energy will be wasted in the sidelobes .Hence detection becomes weaker.[3.1] Range-time sidelobes are the result of convolving the radar return with the non-ideal filter response (i.e. some energy remains outside the desired pulse bandwidth), This results in the "blurring" of returns in range near high reflectivity gradients, like ground clutter.

3.2 Performance measures

The performance measures of Pulse Compression techniques are Peak side level (PSL), integrated sidelobe (ISL), SNR loss and Doppler shift.

3.2.1. Peak Sidelobe Ratio

It is the ratio of the maximum of the sidelobe amplitude to mainlobe amplitude. It measures the highest sideline level next to the mainline level and thus the ratio gives us an idea of how bid the sidelobe is in comparison to the mainlobe[3.1]

$$PSL = 20\log_{10}\left[max\frac{r(i)}{r(0)}\right] i\neq 0 \qquad 3.1$$

3.2.2. Integrated sidelobe ratio

It the ratio of the energy of Autocorrelation function of sidelobes to the total energy of the Autocorrelation function of the mainlobe [3.2].

$$ISL = 10\log_{10}\sum_{i=-L}^{L}\left[\frac{r(i)}{r(0)}\right]^{2} \qquad 3.2$$

3.2.3. SNR Degradation

SNR degradation per dB is the ratio of mainlobe peak amplitude without Doppler shift to mainlobe peak amplitude with Doppler shift.

$$SNR = \left[\frac{\sum_{n=1}^{N} a_{n+k}^{*} a_{n}}{\sum_{n=1}^{N} a_{n} e^{j2\pi n f d\tau} a_{n+k}^{*}}\right] \qquad 3.3$$

3.3 Weighting Techniques for Polyphase Codes

There will be significant reduction in sidelobes and PSL values by implementing time weighting function to the signal code. This sidelobe reduction technique can be analysed in two ways, one is matched weighting with weighting window at the transmitter and the receiver and two is mismatched weighting, where amplitude weighting is performed only at receiver site [3.4]. In this section, simulations are done using mismatched weighting. The tradeoff in reducing the PSL is a spreading of the peak value of the compressed pulse, or mathematically the autocorrelation (ACF) function, resulting in a loss in resolution similar to that of a chirp waveform. The greater the amplitude taper, for example a $\cos n$weighting as n is raised to a higher power, the narrower the bandwidth and hence the wider the compressed pulse. Also, there is a loss in s/n similar to the weighted chirp waveform. Good Doppler tolerance is maintained with these weightings, especially with the Taylor weighting. However, in contrast the sidelobes decrease as the number of P4 code elements increase.

In this section, Kaiser-Bessel time weighting function is analysed due to β parameter and its influence on sidelobe suppression and efficiency in Doppler shift domain, as well. The PSL and integrated sidelobe level (ISL) values are compared for different weighting functions such as Kaiser-Bessel, hamming, hanning, Blackman etc.

3.3.1. Hamming Window

Hamming window belongs to the family of raised cosine windows. The window is optimized to minimize the maximum (nearest) sidelobe, giving it a height of about one-fifth that of the Hann window, a raised cosine with simpler coefficients. The coefficients of a Hamming window are computed from the following equation[3.4].

$W(n) = 0.54 - 0.46\cos(2\pi n/N)\ ,\ 0 \leq n \leq N$ 3.4

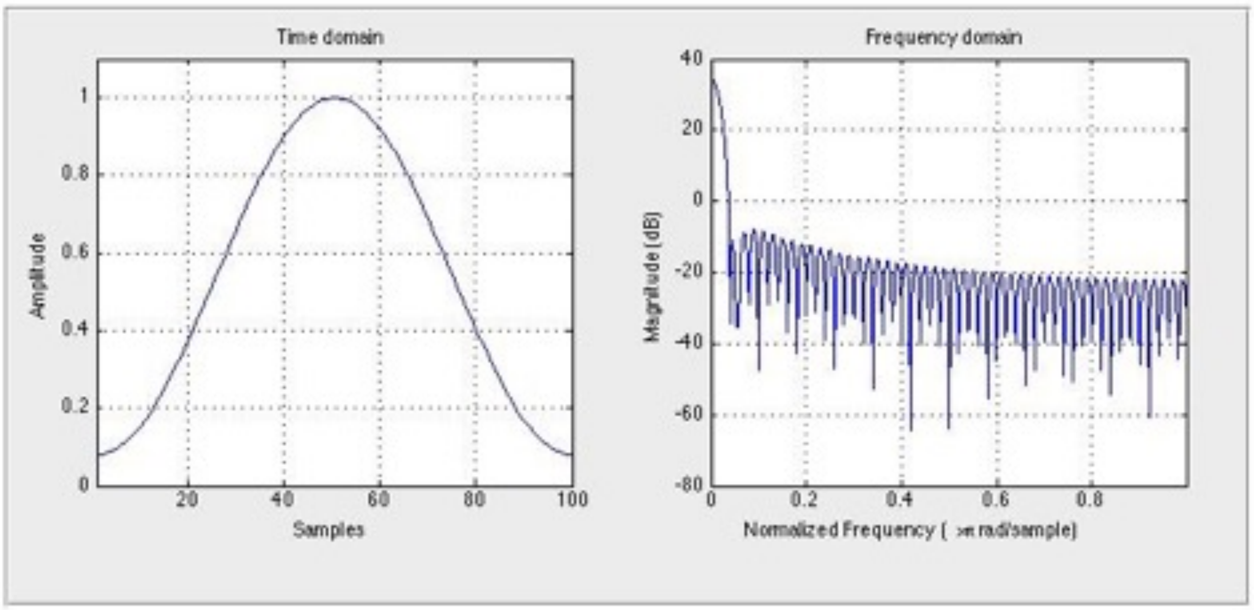


Fig 3.1 100 Point Hamming Window

The 100- point hamming code is shown in Figure 3.1.

3.3.2. Rectangular Window

The rectangular window is sometimes known as a Dirichlet window. It is the simplest window, taking a chunk of the signal without any other modification at all, which leads to discontinuities at the endpoints (unless the signal happens to be an exact fit for the window length, as used in multi tone testing, for instance).[3.5] The first side-lobe is only 13 dB lower than the main lobe, with the rest falling off at about 6 dB per octave.

$$W(n) = 1 \qquad 3.5$$

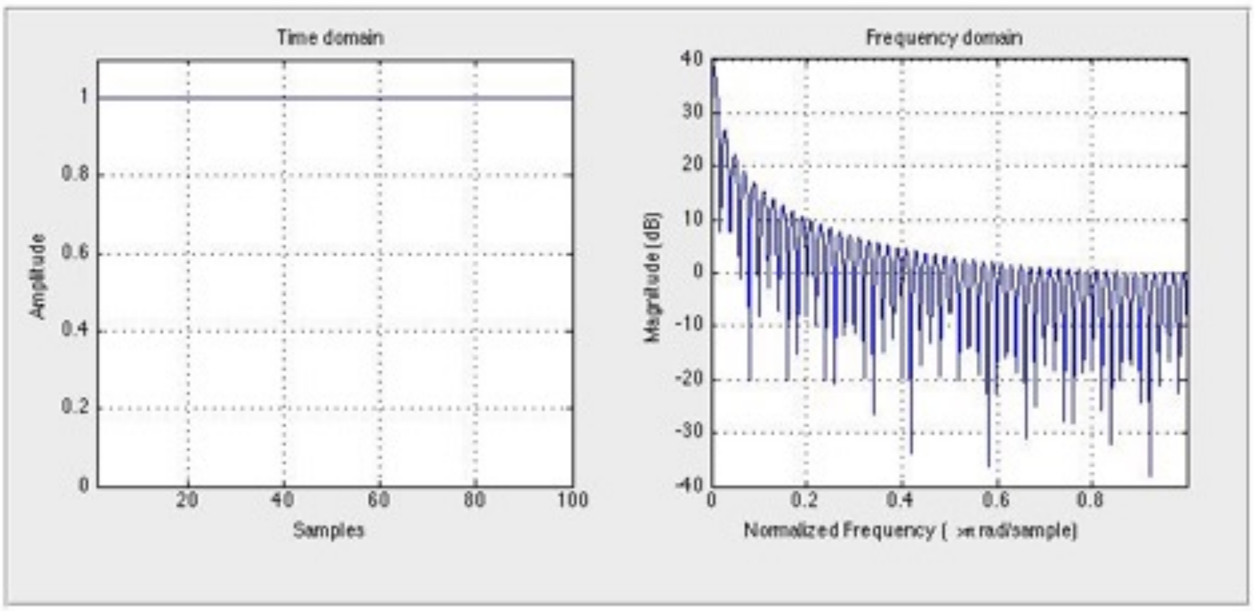


Fig 3.2 100 point Rectangular Window.

3.3.3. Hann Window

The Hann and Hamming windows, both of which are in the family known as "raised cosine" windows, are respectively named after Julius von Hann and Richard Hamming. The term "Hanning window" is sometimes used to refer to the Hann window [3.6, 3.7]. While the Hanning window does a good job of forcing the ends to zero, it also adds distortion to the wave form being analyzed in the form of amplitude modulation; i.e., the variation in amplitude of the signal over the time record. Amplitude Modulation in a wave form results in sidebands in its spectrum, and in the case of the Hanning window, these sidebands, or sidelobes as they are called, effectively reduce the frequency resolution of the analyzer by 50%. The advantage of the Hann window is very low aliasing, and the tradeoff is slightly decreased resolution (widening of the main lobe).

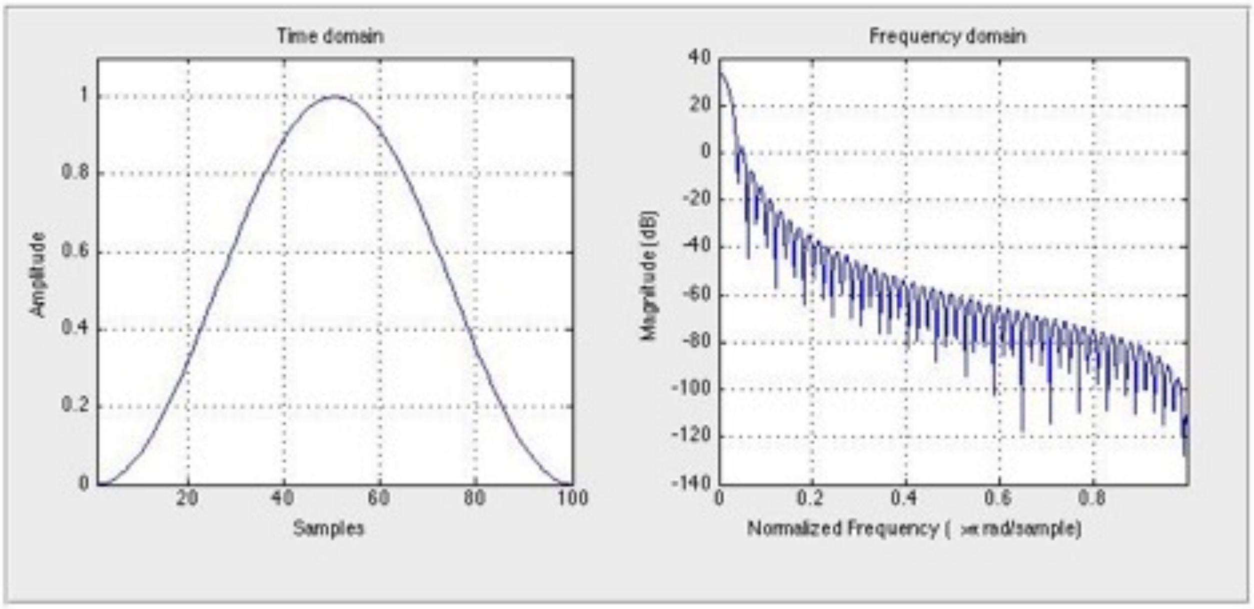


Fig 3.3 100 point Hann Window

3.3.4. Blackman Window

The Blackman window is quite similar to Hann and Hamming window, but it has one additional cosine term to further reduce the ripple ratio. Blackman windows have slightly wider central lobes and less sideband leakage than equivalent length Hamming and Hann windows.

The coefficients of a Hann window are computed from the following equation :

$W(n) = a_0 - a_1 \cos(2\pi n/N-1) + a_2 \cos(2\pi n/N-1)$ 3.6

$a_0 = (1- ß) /2$; $a_1=1/2$; $a_2=ß/2$;

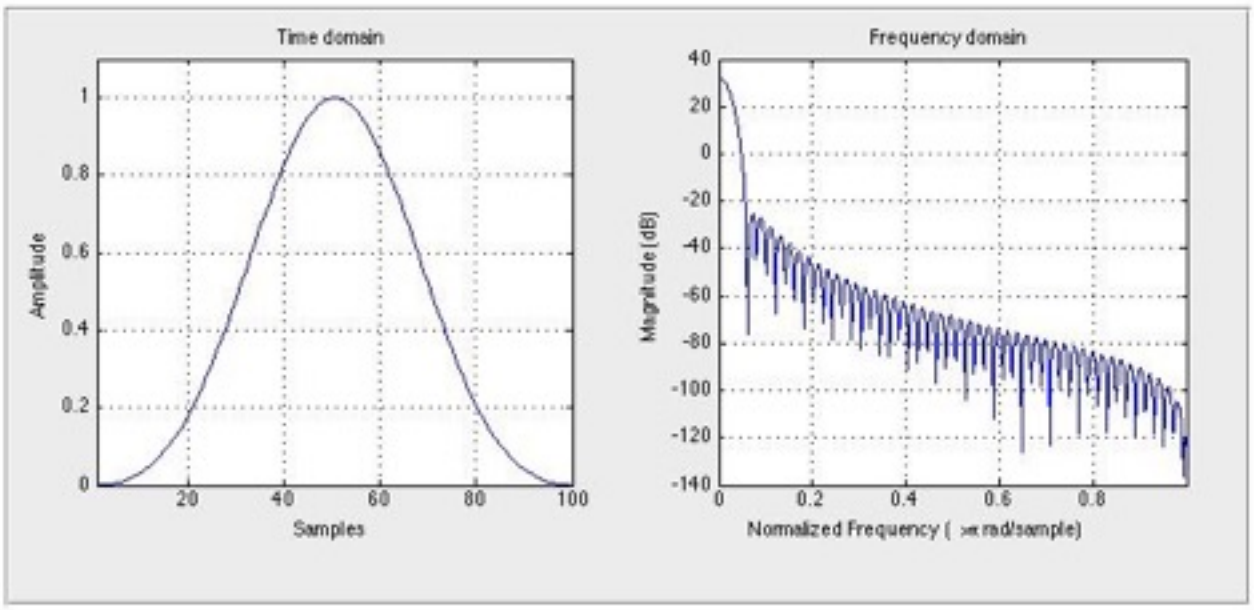


Fig 3.4 100 point Blackman Window

By common convention, the unqualified term *Blackman window* refers to α=0.16.

3.3.5. Kaiser-Bessel Window

For a Kaiser-Bessel window of a particular length N, the parameter β controls the sidelobe height and it affects the sidelobe attenuation of the Fourier transform of the window. This parameter also trades off main lobe width against sidelobe attenuation[3.7]. The Kaiser- Bessel window in sampled version with β is computed as follows :

$$w_n = \begin{cases} \dfrac{\left(\beta\sqrt{1-\left(\dfrac{2n}{N-1}-1\right)^2}\right)}{I_o(\beta)}, & if \quad 1 \leq n \leq N \\ 0, & otherwise \end{cases} \qquad 3.7$$

Where I0 is the zeroth order modified Bessel function of the first kind, β is an arbitrary real number that determines the shape of the window, N is the length of the window. The design formula that is used to calculate β parameter value due required a sidelobe level.

$$\beta = \begin{cases} 0.1102(\alpha-8.7)\ , & \alpha>50 \\ 0.5842(\alpha-21)^{0.4} + 0.07886(\alpha-21)\ , & 21\leq\alpha\leq50 \\ 0, & \alpha>50 \end{cases} \qquad 3.8$$

Where α is sidelobe level in decibels. As β increases, the main lobe width widens and the sidelobe attenuation increases. For β = 0, the Kaiser-Bessel window is a rectangular window. For β= 5.44, the Kaiser-Bessel window is close to the Hamming window. Typically, the value of β is in the range from four to eight and for a given parameter, the sidelobe height is fixed with respect to window length [3.5, 3.6]. For any given window, the signal-to-noise loss (SNR loss) can calculated by the formula:

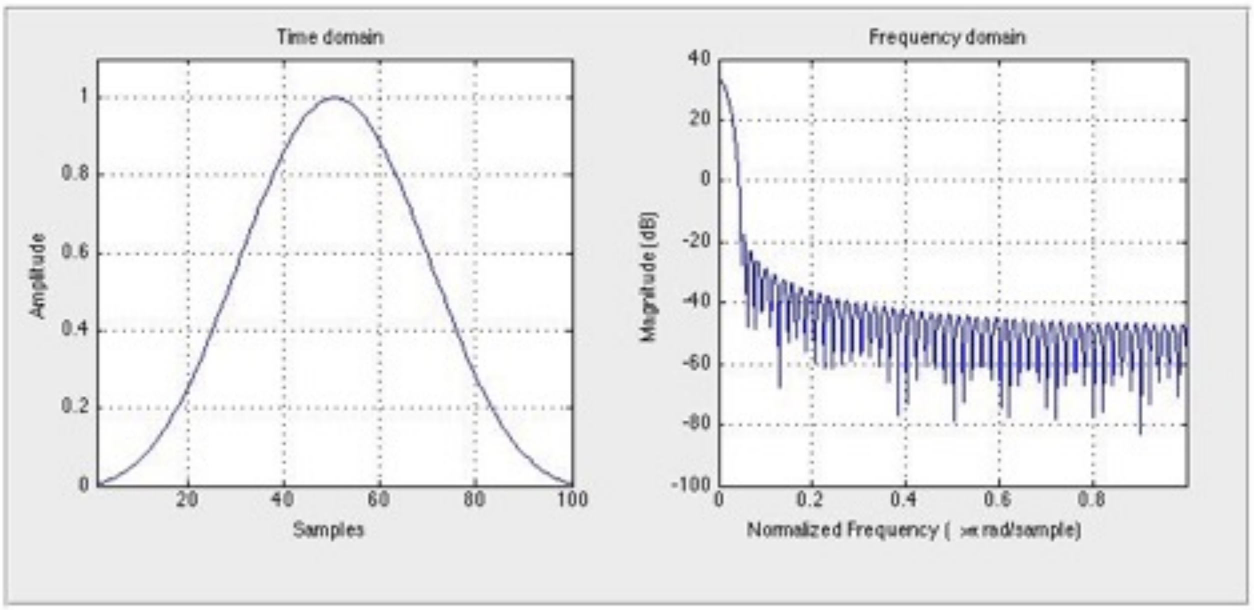


Fig 3.5 100 point Kaiser window of Beta= 4.56;

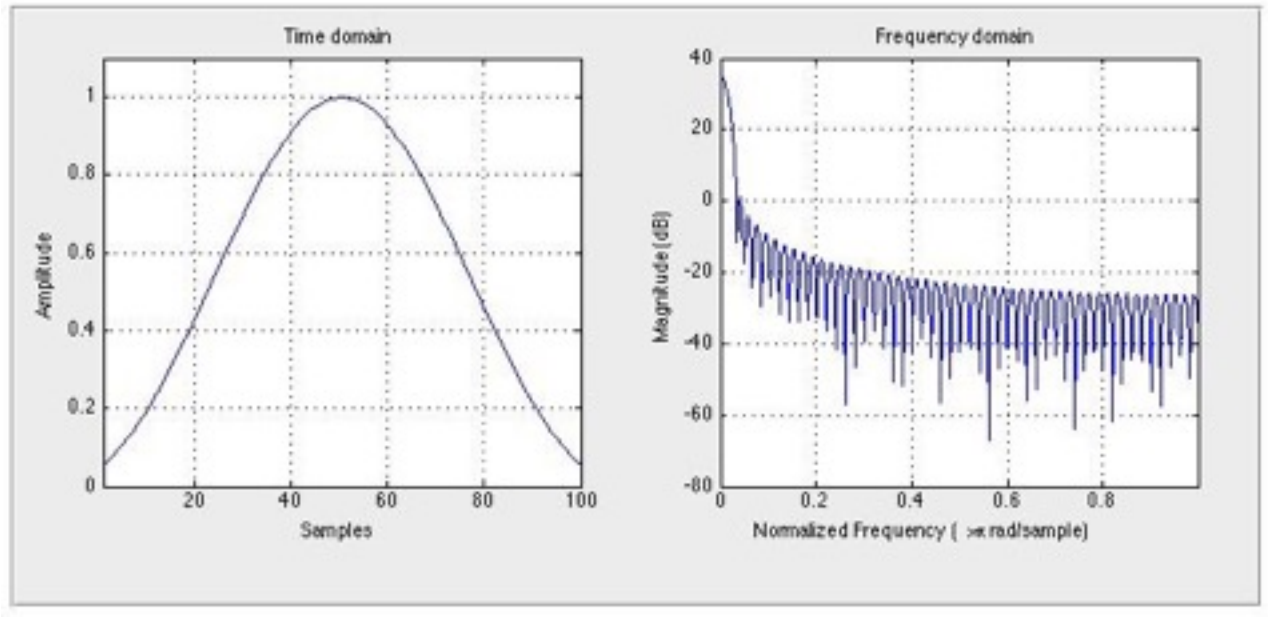


Fig 3.6 , 100 Point Kaiser window of Beta = 7;

For any given window, the signal-to-noise loss (SNR loss) can calculated by the formula:

$$SNR_{loss} = \left[\frac{\left(\sum_{n=1}^{N} w[n] \right)^2}{\sum_{n=1}^{N} w[n]^2} \right] \qquad 3.9$$

3.5.6. Simulation Results and Discussion

All window functions discussed above are applied as sidelobe reductiontechniques for P4 code.

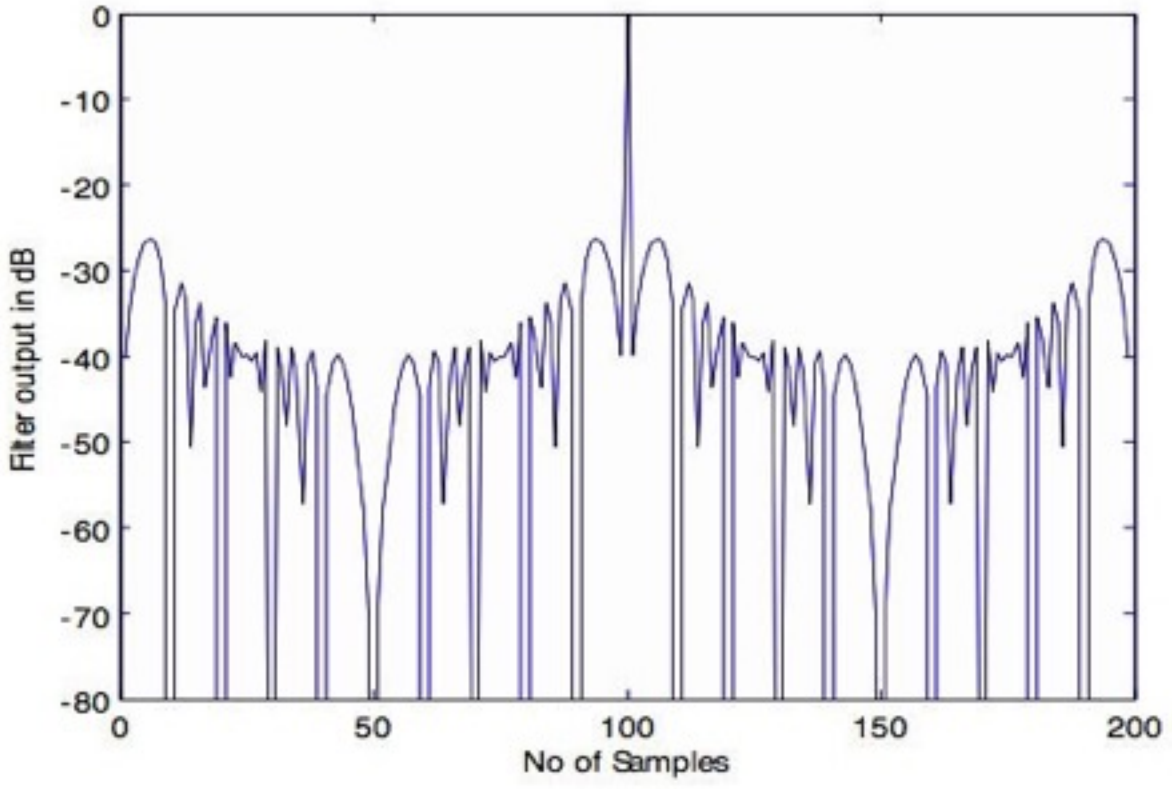


Fig 3.7 P4 ACF of Rectangular window

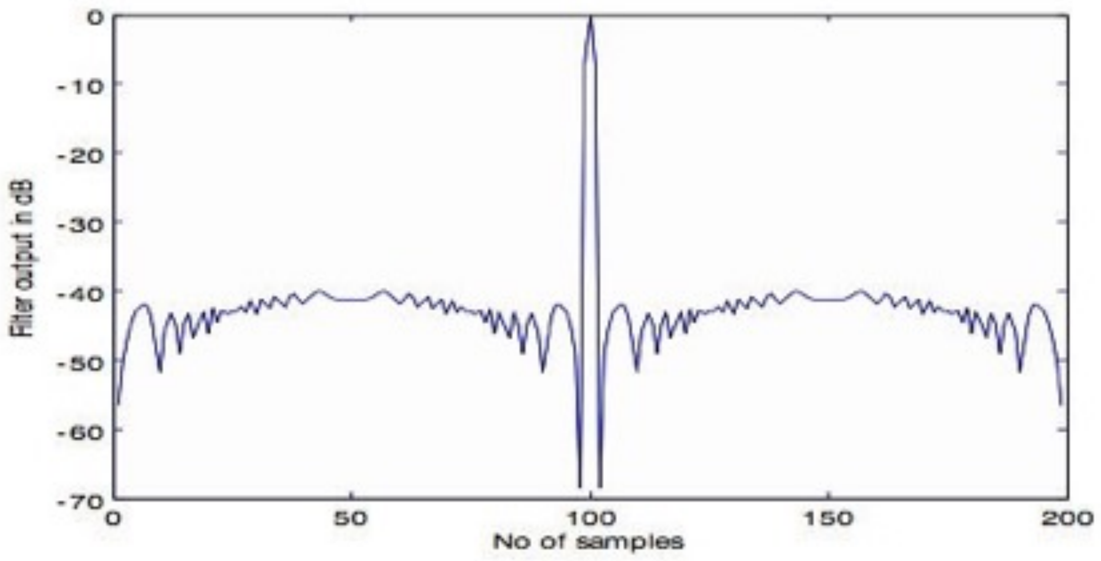


Fig 3.8 P4 ACF of Hann window

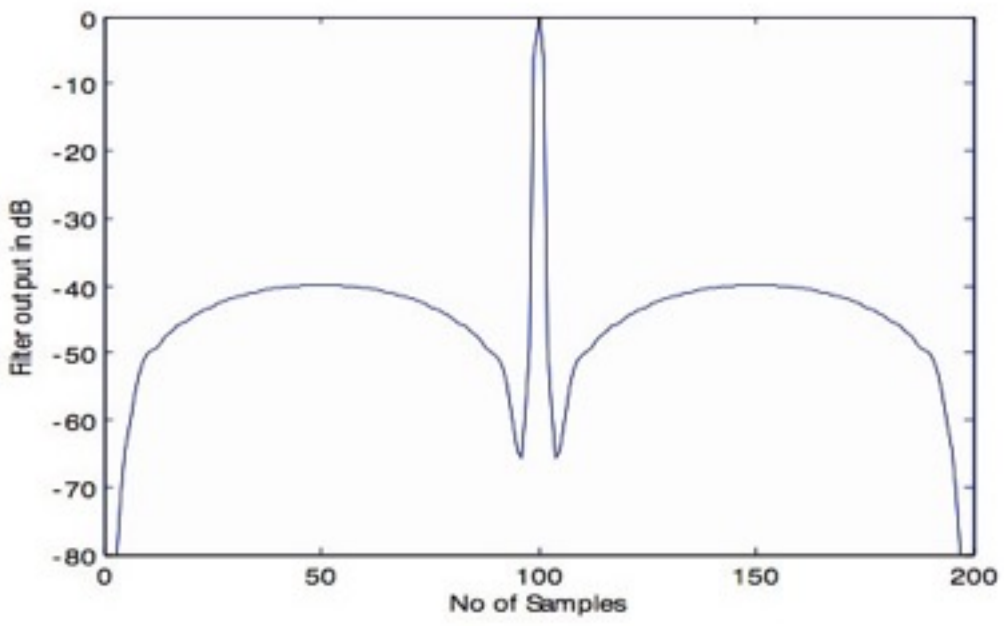


Fig 3.6 P4 ACF of Hamming Window

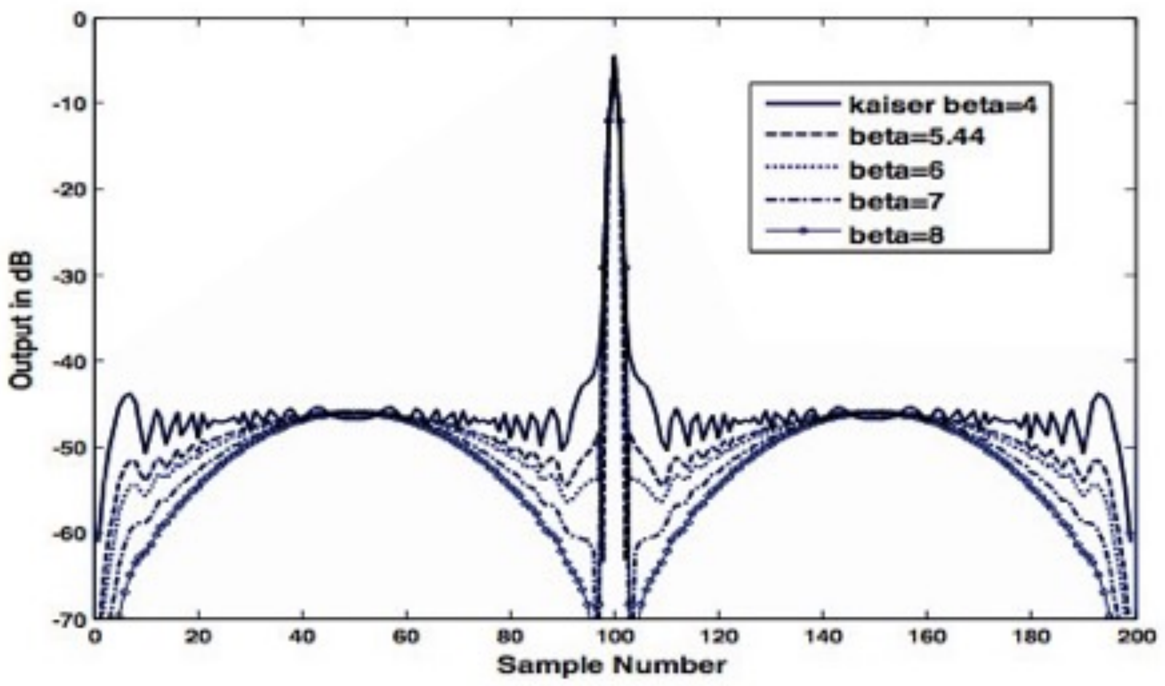


Fig 3.7 P4 ACF of Kaiser Bessel window

Table 1 :Performance of P4 code

WIndow Name	Peak Side Lobe level(dB)	Integrated Sidelobe Level	SNR loss
Hann	-39.69	-19.990	1.2935
Hamming	-37.29	-19.731	1.8046
Blackman	-38.24	-19.825	1.3758
Kaiser(beta=2)	-26.48	-15.445	2.416
Kaiser(beta=3)	-30.18	-17.738	0.57735
Kaiser(beta=4)	-34.31	-19.190	0.98957

3.6. Summary

A detailed structure of sidelobes is discussed and different sidelobe reductions techniques outputs for P4 code are explained in detail and proved that this technique reduces the PSL value. In order to reduce the PSL values further, weighting techniques are employed. The Hamming and Kaiser Bessel windowing functions are studied and their effects to P4 code under Doppler of 0 a are discussed.

Chapter 4

Sidelobe Suppression filter based on Second Order Cone Programming

4.1 Convex Optimization

A convex optimization is one of the form

minimize $f_o(x)$

subject to $f_i(x) \leq bi$, $i= 1,\ldots,m$

where the functions of,…fm: Rn → R are convex, i.e satisfy $f_i(\alpha x+ßy) \leq \alpha f_i(x) + ß\, f_i(y)$

for all x, y $\in$ Rn and all α, $\beta \in$ R with $\alpha+\beta= 1, \alpha\geq 0, \beta\geq 0$. The least-squares problem and linear programming problem are both special cases of the general convex optimisation problem.

4.2 Using convex optimization

Using convex optimization is, at least conceptually, very much like using least squares or linear programming. If we can formulate a problem as a convex optimization problem, then we can solve it efficiently, just as we can solve a least squares problem efficiently. With only a bit of exaggeration, we can say that, if you formulate a practical problem as a convex optimization problem, then you have solved the original problem. There are also some important differences. Recognizing a least-squares problem is straightforward, but recognizing a convex function can be difficult. In addition,there are many more tricks for transforming convex problems than for transforming linear programs. Recognizing convex optimization problems, or those that can be transformed to convex optimization problems, can therefore be challenging. Once the skill of recognizing or formulating convex optimization problems is developed, you will find that surprisingly many problems can be solved via convex optimization.

The challenge, and art, in using convex optimization is in recognizing and formulating the problem. Once this formulation is done, solving the problem is, like least-squares or linear programming, (almost) technology

4.3 Nonlinear optimization

Nonlinear optimization (or nonlinear programming) is the term used to describe an optimization problem when the objective or constraint functions are not linear,but not known to be convex. Sadly, there are no effective methods for solving the general nonlinear programming problem. Even simple looking problems with as few as ten variables can be extremely challenging, while problems with a few hundreds of variables can be intractable. Methods for the general nonlinear programming problem therefore take several different approaches, each of which involves some compromise**.**

4.4 Local optimization

In local optimization, the compromise is to give up seeking the optimal x, which minimizes the objective over all feasible points. Instead we seek a point that is only locally optimal, which means that it minimizes the objective function among feasible points that are near it, but is not guaranteed to have a lower objective value than all other feasible points. A large fraction of the research on general nonlinear programming has focused on methods for local optimization, which as a consequence are well developed.

Local optimization methods can be fast, can handle large-scale problems, and are widely applicable, since they only require differentiability of the objective and constraint functions. As a result, local optimization methods are widely used in applications where there is value in finding a good point, if not the very best. In an engineering design application, for example, local optimization can be used to improve the performance of a design originally obtained by manual, or other, design methods.

There are several disadvantages of local optimization methods, beyond (possibly) not finding the true, globally optimal solution. The methods require an initial guess for the optimization variable. This initial guess or starting point is critical, and can greatly affect the objective value of the local solution obtained. Little information is provided about how far from (globally) optimal the local solution is. Local optimization methods are often sensitive to algorithm parameter values, which may need to be adjusted for a particular problem, or family of problems.

Using a local optimization method is trickier than solving a least-squares prob- lem, linear program, or convex optimization problem. It involves experimenting with the choice of algorithm, adjusting algorithm parameters, and finding a good enough initial guess (when one instance is to be solved) or a method for producing a good enough initial guess (when a family of problems is to be solved). Roughly speaking, local optimization methods are more art than technology. Local opti- mization is a well developed art, and often very effective, but it is nevertheless an art. In contrast, there is little art involved in solving a least-squares problem or a linear program (except, of course, those on the boundary of what is currently possible).

An interesting comparison can be made between local optimization methods for nonlinear programming, and convex optimization. Since differentiability of the objective and constraint functions is the only requirement for most local optimization methods, formulating a practical problem as a nonlinear optimization problem is relatively straightforward. The art in local optimization is in solving the problem (in the weakened sense of finding a locally optimal point), once it is formulated. In convex optimization these are reversed: The art and challenge is in problem formulation; once a problem is formulated as a convex optimization problem, it is relatively straightforward to solve it.

4.4 Global optimization

In global optimization, the true global solution of the optimization problem (1.1) is found; the compromise is efficiency. The worst-case complexity of global opti- mization methods grows exponentially with the problem sizes n and m; the hope is that in practice, for the particular problem instances encountered, the method is far faster. While this favorable situation does occur, it is not typical. Even small problems, with a few tens of variables, can take a very long time (e.g., hours or days) to solve.

Global optimization is used for problems with a small number of variables, where computing time is not critical, and the value of finding the true global solution is very high. One example from engineering design is worst-case analysis or verifica- tion of a high value or safety-critical system. Here the variables represent uncertain parameters, that can vary during manufacturing, or with the environment or op- erating condition.

The objective function is a utility function, i.e., one for which smaller values are worse than larger values, and the constraints represent prior knowledge about the possible parameter values. The optimization problem (1.1) is the problem of finding the worst-case values of the parameters. If the worst-case value is acceptable, we can certify the system as safe or reliable (with respect to the parameter variations).

A local optimization method can rapidly find a set of parameter values that is bad, but not guaranteed to be the absolute worst possible. If a local optimiza- tion method finds parameter values that yield unacceptable performance, it has succeeded in determining that the system is not reliable. But a local optimization method cannot certify the system as reliable; it can only fail to find bad parameter values. A global optimization method, in contrast, will find the absolute worst val- ues of the parameters, and if the associated performance is acceptable, can certify the system as safe. The cost is computation time, which can be very large, even for a relatively small number of parameters. But it may be worth it in cases where the value of certifying the performance is high, or the cost of being wrong about the reliability or safety is high.

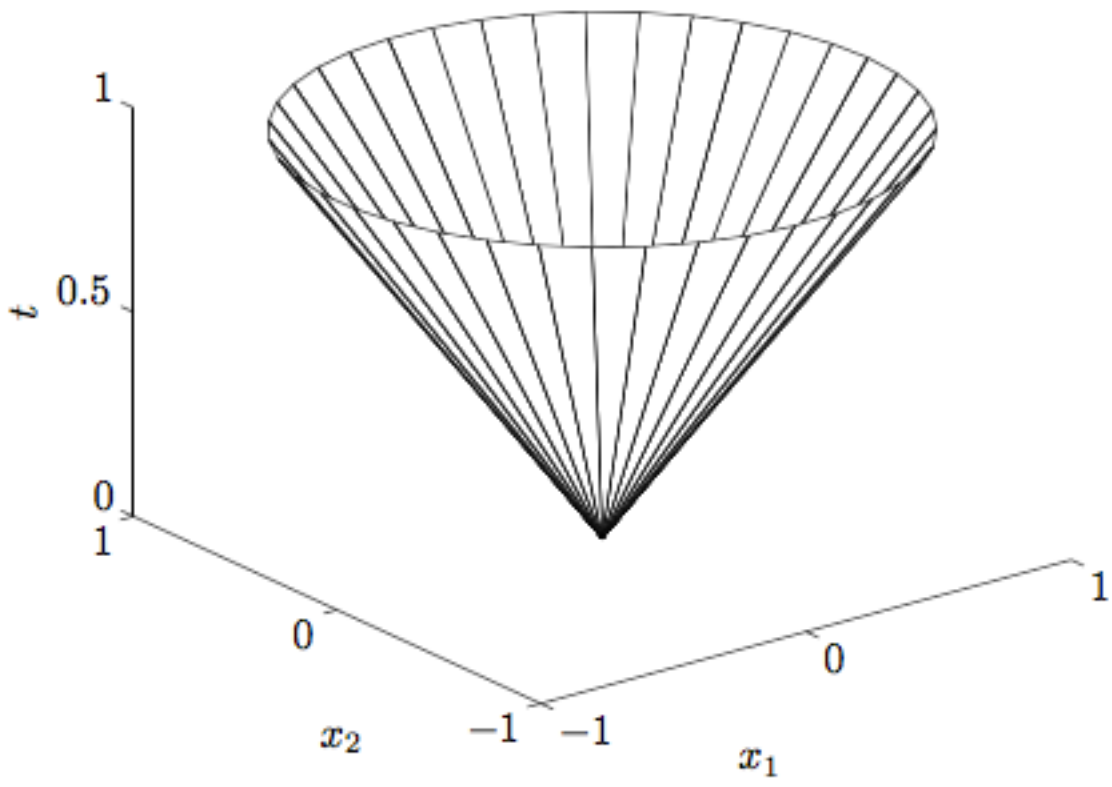


Fig 4 Boundary of second-order cone in R^3, $\{(x_1,x_2,t) \mid (x^2{}_1+x^2)^{1/2} \leq t\}$.

The second-order cone is also known by several other names. It is called the quadratic cone, since it is defined by a quadratic inequality. It is also called the Lorentz cone or ice-cream cone. Figure 2.10 shows the second-order cone in R^3.

4.5 Second Order Cone Programming

The pulse compression concept in radar systems appears as a solution to the dichotomous problem of simultaneously obtaining high transmitted pulse energy, in order to achieve long range, along with high local energy concentration after processing in the radar receiver, to yield high range resolution. Of course, using matched filter to compress signal with large time and frequency band, the output pulse can be narrow. Although pulse compression waveforms bear low side-lobes in their autocorrelations, the side-lobe levels can not satisfy the needs in many practical applications. So, further side-lobe reduction is desired. According to different signals we use variant side-lobes reduction techniques, including two kinds of configuration: the first kind is to add side-lobes reduction filter after matched filter, the other kind is to design mismatched filter directly, realising pulse compression and side-lobes suppression at the same time.

There are many pulse compression signals including linear chirp signal, nonlinear chirp signal, bi-phase codes and polyphase codes etc. Different classes of pulse compression waveforms have different pulse compression properties, and polyphase-pulse-compression codes have many useful features. The Frank, P1, P2 codes which have been derived from step-approximately-to-linear-frequency-modulation pulse compression waveforms and P3, P4 codes which have been de-rived from linear-frequency-modulation pulse compression waveforms are typical polyphase codes, and the P4 code which possess these features such as low range-time-side-lobe, ease of implementation, low cross-correlation between codes, large Doppler tolerance and compatibility with band-pass limited receivers at the same time is the best of them.

In respect to side-lobe reduction for P4 code, several techniques available can be adopted including the classical window function amplitude weighting, the post-compression sliding window 2-sample processing and second-order cone programming amplitude and phase weighting. Of course, using these techniques can substantially reduce the side-lobe levels of the compressed pulses of the codes. However, higher distance resolution and lower side-lobe levels in many special applications are needed, for the polyphase codes with arbitrary phase, optimal design method combined with pulse compression filters was proposed and obtained good results

Classical window function amplitude weighting in time domain and post compression, 2-sample Averager processing are two kind s of classical side-lobe suppression methods specific to polyphase codes, the two methods are belong to the structure of using side-lobe suppression filter after matching pulse compression. In addition, we can suppress the side-lobe through designing mismatched filter directly for p4 code, there are lots of ways to design mismatched filter, such as including the least square method, linear programming and neural network, etc.

They are able to suppress range sidelobes well but have some certain shortcomings, among them the least square method is able to get the optimal filter of minimum integral sidelobes, but it need to iterate for a lot of times and it’s hard to control iteration times and convergence; Generally the linear programming method isn’t suitable for polyphase codes; Convergence speed of neural network method is slow, so it affects its practical application.

However, using above methods to design mismatched filter is only considering to minimise the side-lobes of compressed pulse, but not considering SNR loss. Document proposed a method to design side-lobe suppression filter based on Second-order cone programming, this method translates the design of sidelobe suppression filter under the condition of biggest SNR loss into the second-order cone issue , and using interior point method to solve it efficiently. The specific details will be discussed as follows.

We can get the coefficient f of side-lobe suppression filter from the following formula

$\min \| R^H f \|_m$, s.t. $s^H f = 1$, $\|f\| \le 10^{(\varepsilon/20)}$ 4.1

ε is the biggest SNR loss, the unit is dB.

The formula above can be explained like this, under the constraints of the maximum SNR loss , solve the minimum peak side-lobe filter when the main-lobe value of comp-ressed pulse is set to 1. R is a matrix consists of signal sequence s and zeros, shown as follows.

$$R = \begin{bmatrix} 0 & \cdots & 0 & s_1 & \cdots & s_{M-1} \\ 0 & \cdots & s_0 & s_2 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ s_0 & \cdots & s_{M-2} & 0 & \cdots & 0 \end{bmatrix}_{M\times(2M-2)}$$

where s denotes the signal sequence which filled with zero to both ends , and the middle of the sequence is the code sequence, and the length of is equal to f .

4.6. Optimal design method combined arbitrary phase codes with side-lobe suppression filters optimization

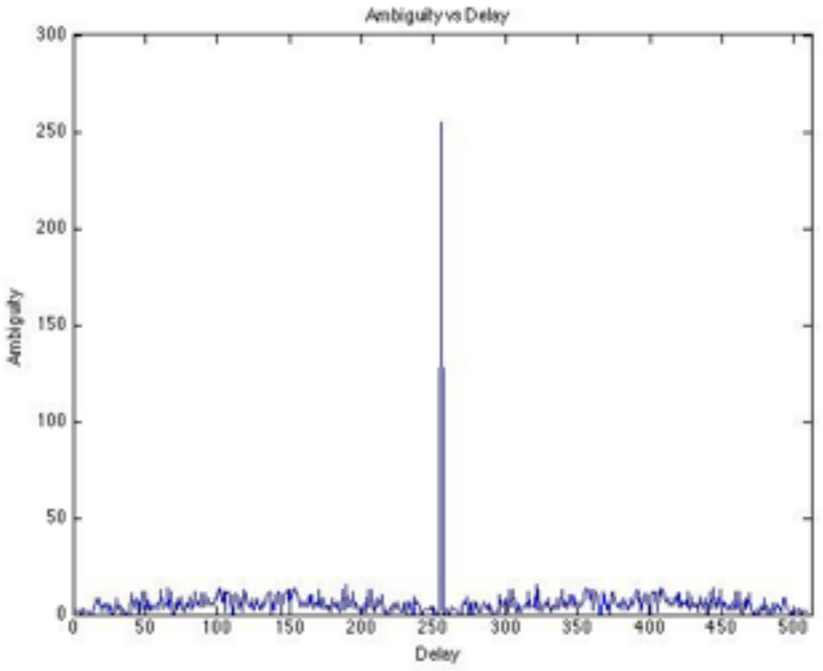
This method finds the required codes and mismatch filters through applying the optimal algorithm to joint optimization of the arbitrary phase codes and the mismatch filter. Genetic algorithms(GA), simulated annealing algorithms(SA), gradient search procedure and nonlinear constrained optimization method are common optimization algorithms. Of course, these methods above reflect a certain advantages and disadvantages when used in the combined optimization.

Inspired by the optimization waveform design method of the mutual information, document put forward a kind of optimal design method combined arbitrary phase codes with side-lobe suppression filters optimization, it could be seen as an improvement of the method based on second-order cone programming.

This method can obtain the final mismatch filter and the corresponding optimization code through lots of iterations. the core ideas of this method is, do some certain operations to the filter coefficients and the initial input signals at this time's iteration, then form a new phase encoding signal as the input signal for the next iteration, it can further reduce the range side-lobes of pulse compression results through iterating for a lot of times.

Using this method, under the condition of the maximum iteration times is 200 and maximum SNR loss is 0.75 dB, to design the optimized codes and the corresponding filter for the p4 codes of different length, we can get the results as illustrated in table 2. Among them, the practical optimized side-lobe refers to the highest side-lobe obtained from the compressed pulse by using joint optimization, the theoretic optimized side-lobe refers to the highest side-lobe obtained according to the matching theory.

From the data of table 2, we can know that this method can get the best side-lobe close to matching filter theory only at the cost of small SNR loss, and filter order equals to the input code, so it will not increase the implement complexity of the practical system.



4.1 Ambiguity vs Delay

These thus discuss and show the results of the pulse compression performance of Polyphase code under different side-lobe reduction methods. The classical window function amplitude weightings can largely reduce the side-lobes but result in a SNR loss and degraded resolution similar to that of LFM waveform.

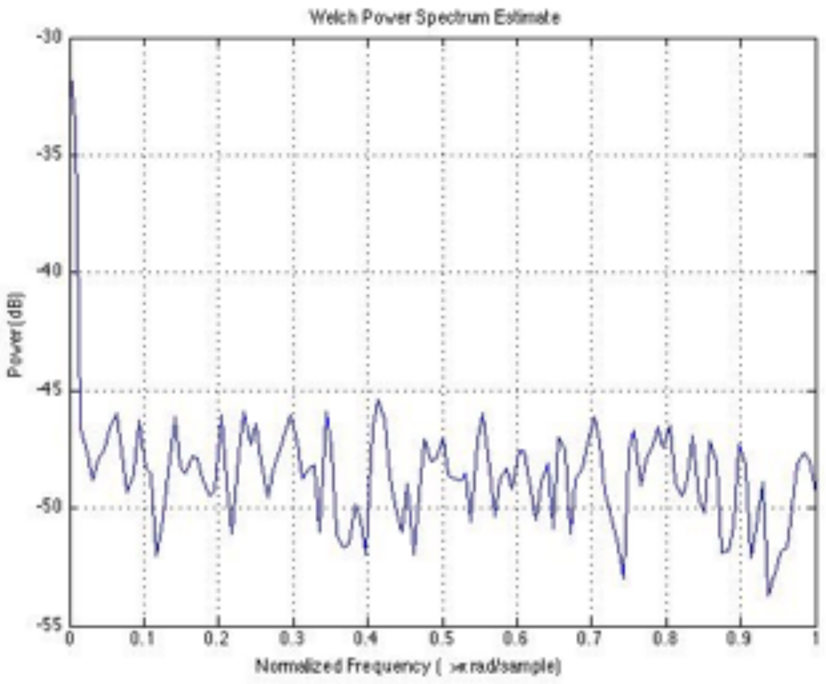


Fig 4.2 Welch Power spectrum with side-lobe suppression filters optimisation

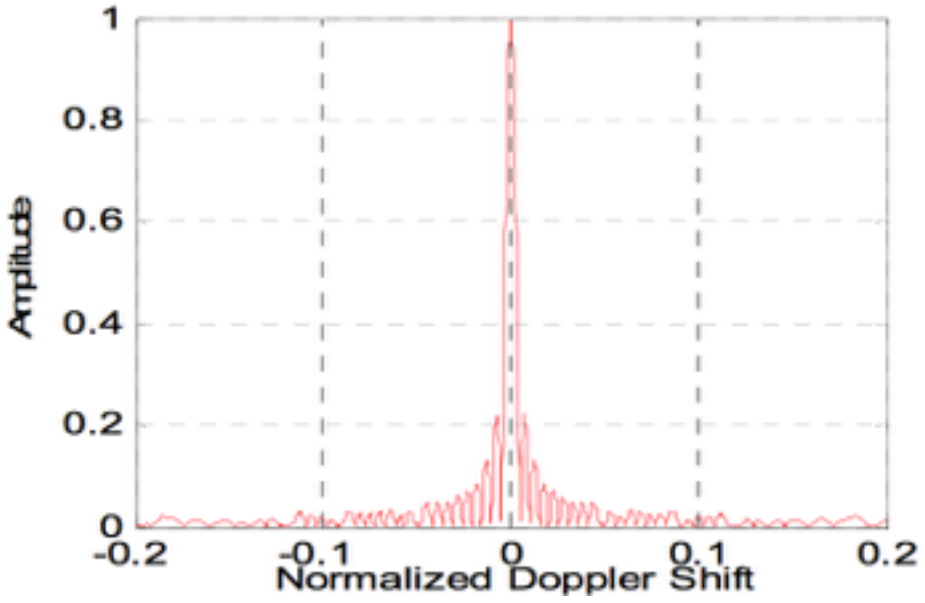


Fig 4.3 Amplitude vsNormalise Doppler shift

4.3.Performance Characteristics

Table 2: The results of compression Pulse comparison under optimised code and filter and windowing techniques

WIndow Name	Peak Side Lobe level(dB)	SNR loss
Hann	-39.69	1.2935
Hamming	-37.29	1.8046
Blackman	-38.24	1.3758
Kaiser(beta=2)	-26.48	2.416
Kaiser(beta=3)	-30.18	0.57735
Kaiser(beta=4)	-34.31	0.98957

Table 3. The result of compressed pulse under optimized code and filter

Code Length	Peak Sidelobe(dB)	SNR loss(dB)
50	-34.5014	≤0.75
100	-39.9917	≤0.75
150	-41.6021	≤0.75
200	-42.5585	≤0.75

Chapter 5

Conclusion and scope for future work

5.1.Conclusion

This thesis discusses and shows the results of the pulse compression performance of P4 code under different side-lobe reduction methods. The classical window function amplitude weightings can largely reduce the side-lobes but result in a SNR loss and degraded resolution similar to that of LFM waveform. The reduced side-lobes and the range resolution are higher than classical window function weighting, and good Doppler tolerance is maintained which is similar to classical window function weighting. It was shown that, the mismatch filter based on second order cone programming result in low PSL and very low SNR loss, the higher the mismatch filter order is, the lower the PSL we can get.

The design method combined arbitrary phase codes with mismatch filters optimization result in the lowest PSL, the lowest SNR loss and the highest range resolution in these methods.. It is to deserved to be noted that, for classical window function weighting method, the side-lobes decrease as the number of P4 code elements increase which is not similar to LFM waveforms.

In fact, we could not find the best method to reduce side-lobes for pulse compression because all methods exist certain defect. According to the practical applications, we should choose the suitable method to play up strengths and avoid weakness, of course, do some corresponding improvements when needed, finally, we could realize the design better.

5.2. Scope of Future Work

The work can be extended by improving PSL performance, SNR performance and specially Doppler shift interference by implementing the sidelobe cancellation technique which exactly cancels all the sidelobes as in the case of complementary code. There is a scope of designing a polyphase code which has lower sidelobes and is more Doppler tolerant than the codes discussed in the thesis by using the P4 code concept. Convex optimization can further help to improve Doppler tolerance and work can be extended to improve PSL and SNR with Doppler shift introduced. Other forms of convex optimisations may also be considered and used depending upon the need and speed of processing.

APPENDIX

MATLAB CODE:

P1 Autocorrelation

```
clear all;
A=1;
f = 1030e6;
fs= 2*f;
j= sqrt(-1);
m=10;
cpp= 1;
tb =1/(fs);

N=m;
for i=1:m
    for j=1:m
        phi(i,j)= (-pi/N)*[N-(2*j-1)]*[(j-1)*N+(i-1)];
    end
end

index=0;

for i=1:m
    for j=1:m
        I(index+1)= A*cos(2*pi*f*tb+phi(i,j)); %In phase component
        Q(index+1)= A*sin(2*pi*f*tb+phi(i,j)); %Quadrature phase component

        time(index+1)=index*tb;
        index = index+1;
    end
end


figure,
corr_signal = xcorr(I);
plot(corr_signal./max(max(corr_signal)));
grid on;
title('P1 Code length-64 Autocorrelation Function')
xlabel('Correlation Indices')
ylabel('Matched Filter Output(dB)');
axis tight;
```

P1 Code Phase values

```
% P1 PHASE
A = zeros(10,10);
B=[];
N=10;
```

```
for m = 1:N
    for n = 1:N
        A(m,n) = wrapTo2Pi(-(pi/N)*[N-(2*n-1)]*[((n-1)*N)+(m-1)]);

        B(end+1) = A(m,n);
    end

end

stem(B);
title('P1 Phase Values');
xlabel('Samples');
ylabel('Radians');
```

P1 code Power spectrum

```
P1 code Power Spectrum:
%generated by Matlab wradar Waveform analyser
h = phased.PhaseCodedWaveform;
% Set the values of the properties
h.SampleRate = 3000000;
h.Code = 'P1';
h.PRF = 10e+3;
h.NumPulses = 10;
h.ChipWidth = 1e-06;
h.NumChips = 4;
% Generate and scale the plot
Fs = h.SampleRate;
x = step(h);
[X, f] = pwelch(x, [], [], [], Fs, 'twosided');
[~, scale, Units] = engunits(max(f));
plot((f-Fs/2)*scale, fftshift(X), 'b');
axis('tight')
xlabel(sprintf('Frequency (%sHz)',Units));
ylabel('Power/Frequency (dB/Hz)');
title('Power Spectral Density of Baseband Signal');
grid on;
```

P2 Code Autocorrelation

```
clc;
clear all;
close all;
A=1;
f = 1030e6;
fs= 2*f;
j= sqrt(-1);
m=10;
```

```
cpp= 1;
tb =1/(fs);

N=10;
for i=1:m
    for j=1:m
        phi(i,j)= (pi/(2*N))*[N-(2*i)+1]*[N-(2*j)+1];
    end
end

index=0;

for i=1:m
    for j=1:m
        I(index+1)= A*cos(2*pi*f*tb+phi(i,j)); %In phase component
        Q(index+1)= A*sin(2*pi*f*tb+phi(i,j)); %Quadrature phase component

        time(index+1)=index*tb;
        index = index+1;
    end
end

figure,
corr_signal = xcorr(I);
plot(corr_signal./max(max(corr_signal)));
grid on;
axis tight;
```

P2 Phase values for each sample

```
%P2 PHASE
A = zeros(10,10);
B=[];
N=10;
for m = 1:N
    for n = 1:N
        A(m,n)  = wrapTo2Pi((pi/(2*N))*[N-(2*m)+1]*[N-(2*n)+1]);

        B(end+1)  = A(m,n);
    end

end

stem(B);
title('P2 PHASE VALUES','color','B');
xlabel('Samples');
ylabel('Radians');
```

P2 Power Spectrum

```
%generated by Matlab wradar Waveform analyser
h = phased.PhaseCodedWaveform;
```

```
% Set the values of the properties
h.SampleRate = 3000000;
h.Code = 'P2';
h.PRF = 10e+3;
h.NumPulses = 3;
h.ChipWidth = 1e-06;
h.NumChips = 4;
% Generate and scale the plot
Fs = h.SampleRate;
x = step(h);
y= filter(Hamming,x);
[X, f] = pwelch(y, [], [], [], Fs, 'twosided');
[~, scale, Units] = engunits(max(f));
plot((f-Fs/2)*scale, fftshift(X), 'b');
axis('tight')
xlabel(sprintf('Frequency (%sHz)',Units));
ylabel('Power/Frequency (dB/Hz)');
title('Power Spectral Density of Baseband Signal');
grid on;
```

P3 Autocorrelation function:

```
clear all;
A=1;
f = 1030e6;
fs= 2*f;
j= sqrt(-1);
m=100;
cpp= 1;
tb =1/(fs);

N=16;
for i=1:m
        phi(i)= (-pi/N)*[(i-1)^2];

end

index=0;
y=sort(abs(phi));
for i=1:m
        I(index+1)= A*cos(2*pi*f*tb+y(i)); %In phase component
        Q(index+1)= A*sin(2*pi*f*tb+y(i)); %Quadrature phase component

        time(index+1)=index*tb;
        index = index+1;

end


figure,
corr_signal = xcorr(I);
plot(corr_signal./max(max(corr_signal)));
grid on;
title('P1 Code length-64 Autocorrelation Function')
```

```
xlabel('Correlation Indices')
ylabel('Matched Filter Output(dB)');
axis tight;
```

P3 Power Spectrum

```
%generated by Matlab wradar Waveform analyser

h = phased.PhaseCodedWaveform;
% Set the values of the properties
h.SampleRate = 3000000;
h.Code = 'P3';
h.PRF = 10e+3;
h.NumPulses = 100;
h.ChipWidth = 1e-06;
h.NumChips = 4;
% Generate and scale the plot
Fs = h.SampleRate;
x = step(h);
[X, f] = pwelch(x, [], [], [], Fs, 'twosided');
[~, scale, Units] = engunits(max(f));
plot((f-Fs/2)*scale, fftshift(X), 'b');
axis('tight')
xlabel(sprintf('Frequency (%sHz)',Units));
ylabel('Power/Frequency (dB/Hz)');
title('Power Spectral Density of Baseband Signal');
grid on;
```

P4 Autocorrelation function

```
clc;
clear all;
close all;
A=1;
f = 1030e6;
fs= 2*f;
j= sqrt(-1);
m=100;
cpp= 1;
tb =1/(fs);
```

```
N=16;
for i=1:m
        phi(i)= (pi/(N))*(i-1)*(i-N-1);

end


index=0;
for i=1:m
        I(index+1)= A*cos(2*pi*f*tb+phi(i)); %In phase component
        Q(index+1)= A*sin(2*pi*f*tb+phi(i)); %Quadrature phase component

        time(index+1)=index*tb;
        index = index+1;

end

figure,
corr_signal = xcorr(I);
plot(corr_signal./max(max(corr_signal)));
grid on;
title('P4 Code length-100 Autocorrelation Function')
xlabel('Correlation Indices')
ylabel('Matched Filter Output');
legend('PSL = 20*log10(0.04083/1) = -26.32 dB');
axis tight;
```

P4 Power Spectrum

```
%generated by Matlab wradar Waveform analyser
h = phased.PhaseCodedWaveform;
% Set the values of the properties
h.SampleRate = 3000000;
h.Code = 'P4';
h.PRF = 10e+3;
h.NumPulses = 3;
h.ChipWidth = 1e-06;
h.NumChips = 4;
% Generate and scale the plot
Fs = h.SampleRate;
x = step(h);
[X, f] = pwelch(x, [], [], [], Fs, 'twosided');
[~, scale, Units] = engunits(max(f));
plot((f-Fs/2)*scale, fftshift(X), 'b');
axis('tight')
xlabel(sprintf('Frequency (%sHz)',Units));
ylabel('Power/Frequency (dB/Hz)');
title('Power Spectral Density of Baseband Signal');
grid on;
```

Barker Code Autocorrelation function

```
% generated by Radar waveform analyser
h = phased.PhaseCodedWaveform;
% Set the values of the properties
h.SampleRate = 3000000;
h.Code = 'Barker';
h.PRF = 10e+3;
h.NumPulses = 1;
h.ChipWidth = 7e-06;
h.NumChips = 13;
% Generate and scale the plot
Fs = h.SampleRate;
x = step(h);
if isa(h, 'phased.FMCWWaveform')
    prf = 1/h.SweepTime;
else
    prf = h.PRF;
end
figure(1)
ambgfun(x,Fs,prf,'Cut','Doppler');
ylabel('Amplitude');
title('Autocorrelation Function/Matched Filter Response');
hold;
```

Frank Code Autocorrelation

```
% F AC
clear all;
A=1;
f = 1030e6;
fs= 2*f;
j= sqrt(-1);
m=10;
cpp= 1;
tb =1/(fs);

N=m;
for i=1:m
    for j=1:m
        phi(i,j)= (2*pi/N)*(j-1)*(i-1);
    end
end
```

```
index=0;

for i=1:m
    for j=1:m
        I(index+1)= A*cos(2*pi*f*tb+phi(i,j)); %In phase component
        Q(index+1)= A*sin(2*pi*f*tb+phi(i,j)); %Quadrature phase component

        time(index+1)=index*tb;
        index = index+1;
    end
end


figure,
corr_signal = xcorr(I);
plot(corr_signal./max(max(corr_signal)));
grid on;
title('Frank Code length-100 Autocorrelation Function')
xlabel('Correlation Indices')
ylabel('Matched Filter Output(dB)');
axis tight;
```

Frank Code Phase values:

```
% F PHASE
A = zeros(10,10);
B=[];
for m = 1:10
    for n = 1:10
        A(m,n) = wrapTo2Pi((2*pi/10)*(m-1)*(n-1));

        B(end+1) = A(m,n);
    end

end

stem(B);
title('Phase Values');
xlabel('Samples');
ylabel('Radians');
```

Matched Filter Function:

```
%nscat is the number of scattering points within window(none)
%rmin min range of receiving window(Km)
%rrec range receiving window(m)
%taup(uncompressed pulse width(seconds)
%f()chirp start frequency (Hz)
%b chirp Bandwidth (Hz)
%scat_range vector of scattered range(Km)
%scat_rsc RCS vector scatteres(m^2)
%win 0- no window(none)
%    1-Hamming
%    2-Kaiser
%    3.chebychev
%y compressed output(volts)
function [y] = matched__filter(nscat, taup, f0, b, rmin, rrec,..
scat_range,scat_rcs, winid)

eps = 1.0e-16;
htau = 0.005e-3 / 2.;
c = 3.e8;
b= b;
n = fix(2. * 0.005e-3 * b);
m = power_integer_2(n);
nfft = 2.^m;
nscat=nscat;
x(nscat,1:nfft) = 0.;
y(1:nfft) = 0.;
replica(1:nfft) = 0.;
winid =winid;%Kaiser
if( winid == 0.)
    win(1:nfft) = 1.;
    win =win';
elseif(winid == 1.)
    win = hamming(nfft);
elseif( winid == 2.)
    win = kaiser(nfft,pi);
elseif(winid == 3.)
    win = chebwin(nfft,60);
end
deltar = c / 2. / b;
max_rrec = deltar * nfft / 2.;
rmin =rmin;
scat_range= scat_range;
maxr = max(scat_range) - rmin;
rrec =rrec;

if(rrec > max_rrec | maxr >= rrec )
'Error. Receive window is too large; or scatterers fall outside window'

end
trec = 2. * rrec / c;
```

```
taup = taup;
deltat = taup / nfft;
t = 0: deltat:taup-eps;
uplimit = max(size(t));
replica(1:uplimit) = exp(1i * 2.* pi * (.5 * (b/taup) .* t.^2));
figure(3)
subplot(2,1,1)
plot(real(replica))
title('Matched filter time domain response')
subplot(2,1,2)
plot(fftshift(abs(fft(replica))));
title('Matched filter frequency domain response')
for j = 1:1:nscat
t_tgt = 2. * (scat_range(j) - rmin) / c +htau;
scat_rcs =scat_rcs;
x(j,1:uplimit) = scat_rcs(j) .* exp(1i * 2.* pi * ...
(.5 * (b/taup) .* (t+t_tgt).^2));
y = y + x(j,:);
end
figure(1)
plot(t,real(y),'k')
xlabel ('Relative delay - seconds')
ylabel ('Uncompressed echo')
title ('Zero delay coincide with minimum range')
rfft = fft(replica,nfft);
yfft = fft(y,nfft);
out= abs(ifft((rfft .* conj(yfft)) .* win' )) ./ (nfft);
figure(2)
time = -htau:deltat:htau-eps;
plot(time,out,'k')
xlabel ('Relative delay - seconds')
ylabel ('Compressed echo')
title ('Zero delay coincide with minimum range')
grid
```

Function to compute PSL, ISL, Merit factor:

```
% ==========================================================================
%
%   Compute estimate Peak Sidelobe Level and Integrated Sidelobe Level
%            of ambiguity function by zero Doppler velocity and
% in case partialy minimize bounds on the volume of ambiguity function with
%                           Doppler velocity.
%  =========================================================================
%" The ambiguity function over the Doppler velocity interval [0,-50]m/s would
be identical ".
function
[ISLakf,PSLakf,MeritFactor,ISLamb,PSLamb]=Compare_Sidelobes(A,B,fftPoint)
%(ï¿½)-must be complex signal (example [-1 -1 -1 -1 -1 1 1 -1 -1 1 -1 1 -1] )
% and (B) the other signal, when we investigation complementary codes. If
% you have only one code for investigation you must type "B=[0]".
% Function return ISL(Integrated Sidelobe Level), PSL(Peak Sidelobe Leve),
Merit Factor and Peak Sidelobes Power for
% differently complex signals.

akf1=xcorr(A);
akf2=xcorr(B);
A1=akf1;
A2=akf2;

 figure
 subplot(2,1,1),plot(A)
 title (' Phase coded transmit waveform')
 %subplot(2,1,1),hold,plot(A1),plot(A2,'r')
 %title (' Pulse Compression of Zcomplementary sets')
 %legend('Zcomlplementary 1' , 'Zcomplementary 2' )

akf=akf1+akf2;
akf=abs(akf);
m = max(akf);
akf=akf./m;
AKF=akf;

s=sum(AKF);
mm=max(AKF);


ISLakf=10*log10((s-mm)/m)

      subplot(2,1,2),plot(AKF)
      title  ( ' Resulting Pulse Compression')
      text (20,0.9,'main lobe power')
      text (2,0.2,'range sidelobes')


    [row,col]=max(AKF);
    maxi=AKF;
    maxi(row,col)=0;
    mm1=max(maxi);

    PSLakf=10*log10(mm1/m)
    MeritFactor=m/(s-mm)

amb1=ambiguity(A);
amb2=ambiguity(B);
amb=amb1+amb2;
x = amb;
[n,m] = size(x);
V1 = 0;
a = m/4;
b = n/4;
V1 = zeros(n,m);

for j=1:m
    for i=1:n
        if (j>(m/2-a)) && (j<(m/2+a))
            if (i>(n/2-b)) && (i<(n/2+b))
                V1(i,j) = x(i,j);
            end
        end
    end
end


B=fft(V1,fftPoint);
V1=abs(B);
V1=abs(V1);
V1=V1.^2;
M=max(V1);
MM=max(M);      % unfixed max value from the matrix "V1"
V2=V1;
V2=V2./MM;      % fixed value on the matrix "V2"
V3=V2;

[M,N] = size(V3);                       %
V3_vector = reshape(V3,1,M*N);
[Value,Index] = max(V3_vector);
V3_vector(Index) = 0;
V3 = reshape(V3_vector,M,N);
M1=max(max(V3));                  % max value on sidelobe level



M2=max(V2);
MM2=max(M2);    % fixed max value from the matrix "V2"

S=sum(V2);
SS=sum(S);

ISLamb=10*log10((SS-MM2)/MM)
PSLamb=10*log10(M1/MM)
disp(ISLamb);
disp(PSLamb);

 figure
 mesh(V2)
 title  ( ' Ambiguity function')
 %text (10,10,1.2,' x(t,f)')
axes_handle = xlabel('t/T');
set(axes_handle,'FontName','Symbol');
axes_handle = ylabel('nT');
set(axes_handle,'FontName','Symbol');
axes_handle = zlabel('|c(t,n)|');
set(axes_handle,'FontName','Symbol');
```

Function to design windows

```
%one can also design using the matlab filter design app.

function h=tftb_window(N,name,param,param2);
%tftb_window    Window generation.
%   H=tftb_window(N,NAME,PARAM,PARAM2)
%   yields a window of length N with a given shape.
%
%   N      : length of the window
%   NAME   : name of the window shape (default : Hamming)
%   PARAM  : optional parameter
%   PARAM2 : second optional parameters
%
%   Possible names are :
%   'Hamming', 'Hanning', 'Nuttall',  'Papoulis', 'Harris',
%   'Rect',    'Triang',  'Bartlett', 'BartHann', 'Blackman'
%   'Gauss',   'Parzen',  'Kaiser',   'Dolph',    'Hanna'.
%   'Nutbess', 'spline',  'Flattop'
%
%   For the gaussian window, an optionnal parameter K
%   sets the value at both extremities. The default value is 0.005
%
%   For the Kaiser-Bessel window, an optionnal parameter
%   sets the scale. The default value is 3*pi.
%
%   For the Spline windows, h=tftb_window(N,'spline',nfreq,p)
%   yields a spline weighting function of order p and frequency
%   bandwidth proportional to nfreq.
%
%       Example:
%        h=tftb_window(256,'Gauss',0.005);
%        plot(0:255, h); axis([0,255,-0.1,1.1]); grid
%
%
if (nargin==0), error ( 'at least 1 parameter is required' ); end;
if (N<=0), error('N should be strictly positive.'); end;
if (nargin==1), name= 'Hamming'; end ;
name=upper(name);
if strcmp(name,'RECTANG') | strcmp(name,'RECT'),
 h=ones(N,1);
elseif strcmp(name,'HAMMING'),
 h=0.54 - 0.46*cos(2.0*pi*(1:N)'/(N+1));
elseif strcmp(name,'HANNING') | strcmp(name,'HANN'),
 h=0.50 - 0.50*cos(2.0*pi*(1:N)'/(N+1));
elseif strcmp(name,'KAISER'),
 if (nargin==3), beta=param; else beta=3.0*pi; end;
 ind=(-(N-1)/2:(N-1)/2)' *2/N; beta=3.0*pi;
 h=bessel(0,j*beta*sqrt(1.0-ind.^2))/real(bessel(0,j*beta));
elseif strcmp(name,'NUTTALL'),
 ind=(-(N-1)/2:(N-1)/2)' *2.0*pi/N;
 h=+0.3635819 ...
   +0.4891775*cos(    ind) ...
   +0.1363995*cos(2.0*ind) ...
   +0.0106411*cos(3.0*ind) ;
elseif strcmp(name,'BLACKMAN'),
 ind=(-(N-1)/2:(N-1)/2)' *2.0*pi/N;
 h= +0.42 + 0.50*cos(ind) + 0.08*cos(2.0*ind) ;
elseif strcmp(name,'HARRIS'),
 ind=(1:N)' *2.0*pi/(N+1);
 h=+0.35875 ...
   -0.48829 *cos(    ind) ...
   +0.14128 *cos(2.0*ind) ...
   -0.01168 *cos(3.0*ind);
elseif strcmp(name,'BARTLETT') | strcmp(name,'TRIANG'),
 h=2.0*min((1:N),(N:-1:1))'/(N+1);
elseif strcmp(name,'BARTHANN'),
 h=  0.38 * (1.0-cos(2.0*pi*(1:N)/(N+1))') ...
   + 0.48 * min((1:N),(N:-1:1))'/(N+1);
elseif strcmp(name,'PAPOULIS'),
 ind=(1:N)'*pi/(N+1); h=sin(ind);
elseif strcmp(name,'GAUSS'),
 if (nargin==3), K=param; else K=0.005; end;
 h= exp(log(K) * linspace(-1,1,N)'.^2 );
elseif strcmp(name,'PARZEN'),
 ind=abs(-(N-1)/2:(N-1)/2)'*2/N; temp=2*(1.0-ind).^3;
 h= min(temp-(1-2.0*ind).^3,temp);
elseif strcmp(name,'HANNA'),
 if (nargin==3), L=param; else L=1; end;
 ind=(0:N-1)';h=sin((2*ind+1)*pi/(2*N)).^(2*L);
elseif strcmp(name,'DOLPH') | strcmp(name,'DOLF'),
 if (rem(N,2)==0), oddN=1; N=2*N+1; else oddN=0; end;
 if (nargin==3), A=10^(param/20); else A=1e-3; end;
 K=N-1; Z0=cosh(acosh(1.0/A)/K); x0=acos(1/Z0)/pi; x=(0:K)/N;
 indices1=find((x<x0)|(x>1-x0));
 indices2=find((x>=x0)&(x<=1-x0));
 h(indices1)= cosh(K*acosh(Z0*cos(pi*x(indices1))));
 h(indices2)= cos(K*acos(Z0*cos(pi*x(indices2))));
 h=fftshift(real(ifft(A*real(h))));h=h'/h(K/2+1);
 if oddN, h=h(2:2:K); end;
elseif strcmp(name,'NUTBESS'),
 if (nargin==3), beta=param; nu=0.5;
 elseif (nargin==4), beta=param; nu=param2;
 else beta=3*pi; nu=0.5;
 end;
 ind=(-(N-1)/2:(N-1)/2)' *2/N;
 h=sqrt(1-ind.^2).^nu .* ...
   real(bessel(nu,j*beta*sqrt(1.0-ind.^2)))/real(bessel(nu,j*beta));
elseif strcmp(name,'SPLINE'),
 if (nargin < 3),
  error('Three or four parameters required for spline windows');
 elseif (nargin==3),
  nfreq=param; p=pi*N*nfreq/10.0;
 else nfreq=param; p=param2;
 end;
  ind=(-(N-1)/2:(N-1)/2)';
  h=sinc((0.5*nfreq/p)*ind) .^ p;
elseif strcmp(name,'FLATTOP'),
 ind=(-(N-1)/2:(N-1)/2)' *2.0*pi/(N-1);
 h=+0.2810639 ...
   +0.5208972*cos(    ind) ...
   +0.1980399*cos(2.0*ind) ;
else error('unknown window name');
end;
```

Function to generate n length sequence for optimization

```
%Notes ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
! !
%Here is a function of m sequence generated when using them should " call " ;
here called " call " is to give the
%Function of an input parameter ( this program refers to the n), m sequence is
then generated by the function can be assigned to a vector ,
%And does not require initialization vector . For example , this time going to
generate a sequence of length 127 m , as long as typing at the command line :
%a = m_gen (4) can be, where to store the sequence of vectors m (here a) the
name can be arbitrarily defined ; input parameters n = 4
% Corresponds to a sequence of length 127 m . If you want to generate a
sequence of other m length , simply by changing the input parameters to the
program
% Of the comments section has specified no corresponding n- m length of the
sequence . The last part of the instructions in order to generate a sequence of
m
% Correctness verified, you can not

function mesq=m_gen(n)
%------------- M sequence generation -------------------------%
eps=0.00000001;
switch n
    case 1
        connections=[0 0 1 1];                      % 15 yards long
        register=[1 0 1 0];
    case 2
        connections=[0 0 1 0 1];                    % 31 yards long
        register=[1 0 1 0 1];
    case 3
        connections=[0 0 0 0 1 1];                  % 63ï¿½ë³¤
        register=[1 0 1 0 1 0];
    case 4
        connections=[0 0 0 0 0 1 1];                % 127ï¿½ë³¤
        register=[1 0 1 0 1 0 1];
    case 5
        connections=[0 1 1 1 0 0 0 1];              % 255ï¿½ë³¤
        register=[1 0 1 0 1 0 1 0];
    case 6
```

```
        connections=[0 0 0 1 0 0 0 0 1];                % 511ï¿½ë³¤
        register=[1 0 1 0 1 0 1 0 1];
    case 7
        connections=[0 0 1 0 0 0 0 0 0 1];              % 1023ï¿½ë³¤
        register=[1 0 1 0 1 0 1 0 1 0];
    case 8
        connections=[0 1 0 0 0 0 0 0 0 0 1];            % 2047
        register=[1 0 1 0 1 0 1 0 1 0 1];
    case 9
        connections=[0 0 0 0 0 1 0 1 0 0 1 1];          % 4095ï¿½ë³¤
2011.02.18
        register=[1 0 1 0 1 0 1 0 1 0 1 0];
    otherwise
        connections=[0 0 0 0 0 0 0 0 1 1 0 1 1];        % 8191 yards long
2011.02.18
        register=[1 0 1 0 1 0 1 0 1 0 1 0 1];
end
n=length(connections);
N=2^n-1;                                    %Code length
mesq(1)=register(n);
for i=2:N
    newregister(1)=mod(sum(connections.*register),2);
    for j=2:n
        newregister(j)=register(j-1);
    end
    register=newregister;
    mesq(i)=register(n);
end
mesq=2*mesq-1;
```

Function to generate Ambiguity vs Delay

```
function[mm]=mxu(cn)  %cn Coefficients for the shift register
len=length(cn);%Desired length of the shift register
L=2^len-1;%m length sequence
an=[1,zeros(1,len-2),1];%The initial register contents
m(1)=an(len);%The first sequence of output symbols m
for i=2:L
 for j=2:len
an1(j)=an(j-1);
an1(1)=mod(sum(cn.*an),2);%Register with mode feedback 2 and
 end
an=an1;%After the shift register
m(i)=an(len);%The new register output
end
figure(1);
stairs(m);%Drawing on the m-sequence
axis([0 1200 -2 2]);
mm=2*m-1;
%For m sequence verification
Y=abs(conv(fliplr(mm),mm));
figure(2);
```

```
plot(Y);

xlabel('Delay')
ylabel('Ambiguity')
title('Ambiguity vs Delay')
```

Function to generate Ambiguity vs Delay
for some initial sequence cn:

```
%used for comparison
close all;
clear all;
clc;

cn=[0 1 1 1 0 0 0 1];
%cn=[0 1 1 1 0 0 0 1];
sn=mxu(cn);
sn1=abs(conv(fliplr(sn),sn));


q=0;%The maximum gain loss constraint
w=w_w(q);

so=(conv(fliplr(sn),w));
so1=(so)/max(abs(so));

figure(1);
plot(sn1);

figure(2);
plot(so1);
title('Ambiguity vs Delay ')
axis tight
```

Function to suppress sidelobes using second order cone programming:

```
function w=w_w(q)

cn=[0 0 0 1 1 1 1 ];
%cn=[0 1 1 1 0 0 0 1];
m1=mxu(cn);%m--1023
N=length(m1);
m=m1/sqrt(N);M=2*N+1;%2047
M2=2*M-1;%4093
N1=(M-N)/2;
N2=M-N-N1;
A1=zeros(1,N1);
A2=zeros(1,N2);
S=[A1 m A2];%S--2047
X=zeros(M,M2);
A5=zeros(1,M-1);
X(1,:)=[A5,S];
X(M,:)=[S,A5];
for k=2:M-1
    A3=zeros(1,M-k);
    A4=zeros(1,k-1);
    X1=[A3,S,A4];
    X(k,:)=X1;
end
  XX=X;
  XX(:,M)=[];
  A=XX;
cvx_begin  %convex programming where variable(fucntion) is subjected to
optimization under subjective functions
    variable w(M);
    minimize (max(abs(A'*w)))
    subject to
        S*w == 1;
        (w')*w<10^(q/10)
cvx_end
```

References

Chapter-1

Merrill I. Skolnik, *Introduction to radar systems*, McGraw Hill Book Company Inc.,1962.

Nadav Levanon, Eli Mozeson, “*Radar Signals*”, 1.st Editon Wiley-Interscience, 2004.

Carpentier, Michel H., "Evolution of Pulse Compression in the Radar Field," *Microwave Conference, 1979. 9th European*, vol., no., pp.45-53, 17-20 Sept. 1979

R. L. Frank, “Polyphase Codes with Good Non periodic Correlation Properties”, *IEEE Trans. on Information Theory*, vol. IT-9, pp. 43-45, Jan. 1963.

B. L. Lewis, F. F. Kretschmer Jr., “Linear Frequency Modulation Derived Polyphase Pulse Compression Codes”, *IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-18, no. 5, pp. 637-641, Sep. 1982.

W.K.Lee, H.D.Griffiths. “Pulse compression filters generating optimal uniform range sidelobe level”. *Electronic Letter* 1999, Vol. 35.No.11.

Woo-Kyung lee. “A Pair of asymmetrical weighting receivers and polyphase codes for

efficient aperiodic correlations.” *IEEE Communication Letters* Vol.10, No.5, May 2006.

Chapter-2

Merrill I. Skolnik, *Introduction to radar systems*, McGraw Hill Book Company Inc.,1962. [2.2] F.E. Nathanson, J. P. Reilly and M. N. Cohen, *“Radar Design Principles Signal*

Processing and the Environment”, 2nd ed. New York: McGraw-Hill, 1999, chapt. 1 & 8. [2.3] W. Siebert, “A Radar Detection Philosophy,” *IRE Trans.,* vol.IT-2, no. 3, pp. 204-221, Sept. 1956.

M.N.Cohen, J.M. Baden, and P. E. Cohen, “Biphase Codes with Minimum Peak Sidelobe,” IEEE *International Radar Conf*, 1989, pp. 62-66.

Nadav Levanon, Eli Mozeson, “Radar Signals”, 1.st Editon Wiley-Interscience, 2004. M.G. Parker, K.G. Paterson & C. Tellambura, “Golay Complementary Sequences”, in Wiley Encyclopedia of Telecommunications, John G. Proakis, ed., Wiley, 2003
S. Searle, S. Howard, “A Novel Polyphase Code for Sidelobe Suppression”, Invited Paper, *IEEE Trans. On Waveform Diversity and Design*, 2007.

Talal Darwich, “*High Resolution Detection Systems using Low Sidelobe Pulse Compression Techniques*”, Ph.D. Thesis, University of Louisiana, 2007.
R. L. Frank, “Polyphase Codes with Good Nonperiodic Correlation Properties”, *IEEE Trans. on Information Theory*, vol. IT-9, pp. 43-45, Jan. 1963

B. L. Lewis, F. F. Kretschmer Jr., “Linear Frequency Modulation Derived Polyphase Pulse Compression Codes*”, IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-18, no. 5, pp. 637-641, Sep. 1982

B. L. Lewis, “Range-Time-Sidelobe Reduction Technique for FM-Derived Polyphase PC Codes*”, IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-29, no. 3, pp. 834-840, July 1993

Chapter-3

B. L. Lewis, F. F. Kretschmer Jr., “A New Class of Polyphase Pulse Compression Codes and Techniques”, IEEE *Trans. on Aerospace and Electronic Systems*, vol. AES-17, no. 3, pp. 364-372, May 1981

B. L. Lewis, “Range-Time-Sidelobe Reduction Technique for FM-Derived Polyphase

PC Codes*”, IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-29, no. 3, pp. 834-840, July 1993

Woo-Kyung lee, Hugh D.Griffiths. “A new pulse compression Techniques Generating Optimal uniform Range sidelobe and reducing integrated sidelobe level.” *IEEE International Radar Conference* 2000

F. F. Kretschmer Jr., L. R. Welch, “Sidelobe Reduction Techniques for Polyphase Pulse

Compression Codes”, *IEEE International Radar Conference*, pp. 416-421, May 2000. M. Luszczyk, D. Mucha, "Kaiser-Bessel window weighting function for polyphase pulse

compression code," *Microwaves, Radar and Wireless Communications, 2008. MIKON 2008. 17th International Conference on* , vol., no., pp.1-4, 19-21 May 2008.

Oppenheim, A.V., and R.W. Schafer, *“Discrete-Time Signal Processing*”, Prentice-Hall, 1989, pp. 447-448.

Frank F. Kretschmer and Laurence R. Welch, “Sidelobe reduction techniques for polyphase pulse compression codes” *IEEE International Radar Conference* Pp.416-421.

Kaiser, J.F., "Nonrecursive Digital Filter Design Using the I0- sinh Window Function," *Proc. 1974 IEEE on Symp. Circuits and Systems*, (April 1974), pp.20-23.

Chapter-4

Lewis B L,Kretschmer F F Jr. A New Class of Polyphase Pulse Compression Codes and Techniques[J]. IEEE Trans on AES,1981,17(3):364-372.

Tao Haihong,Liao Guisheng. Bi-phase Encoded Waveform Design to Deal With the Range Ambiguities for Sparse Space-based Radar Systems[J]. Journal of Astronautics,2005,26(5), 657-662.

Deng H. Polyphase Code Design for Orthogonal Netted Radar Systems[J]. IEEE Trans on Signal Processing,2004,52(11):3126-3135.

Xue-hui,WU Zhao-ping,Su Tao,WU Shun-jun. Optimal design method combined arbitrary phase codes with pulse compression filters optimization. Key Lab. of Radar Signal Processing ,Xidian Univ,2009,36(6),1027-1032.

ZHANG Liping,PENG Yingning,WANG Xiutan,WANG Xiqin. Slide-window side-lobe suppression filter for LFM and LFM-derived polyphase coded waveforms. J Tsinghua Univ (Sci & Tech),2001,41(1),20-23

Xue-hui,WU Zhao-ping,Su Tao,WU Shun-jun. Design of peak sidelobe suppression filter based on second- order cone programming. Systems Engineering and Electronics,2009,31(11),2567-2570.