Eyung W. Kang

radar system

analysis, design, and simulation

CD-ROM Included

Radar System Analysis, Design, and Simulation

DISCLAIMER OF WARRANTY

The technical descriptions, procedures, and computer programs in this book have been developed with the greatest of care and they have been useful to the author in a broad range of applications; however, they are provided as is, without warranty of any kind. Artech House, Inc. and the author and editors of the book titled *Radar System Analysis, Design, and Simulation* make no warranties, expressed or implied, that the equations, programs, and procedures in this book or its associated software are free of error, or are consistent with any particular standard of merchantability, or will meet your requirements for any particular application. They should not be relied upon for solving a problem whose incorrect solution could result in injury to a person or loss of property. Any use of the programs or procedures in such a manner is at the user's own risk. The editors, author, and publisher disclaim all liability for direct, incidental, or consequent damages resulting from use of the programs or procedures in this book or the associated software.

For a listing of recent titles in the *Artech House Radar Library*, turn to the back of this book.

Radar System Analysis, Design, and Simulation

Eyung W. Kang

ARTECH
HOUSE
BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalog record for this book is available from the British Library.

ISBN-13: 978-1-59693-347-7

685 Canton Street
Norwood, MA 02062

10 9 8 7 6 5 4 3 2 1

Disclaimer:
This eBook does not include the ancillary media that was packaged with the original printed version of the book.

Contents

Preface

This book aims to reach radar system engineers who strive to optimize system performance and graduate students who wish to learn radar system design. It addresses the need to master system analysis and design skills and to verify that the analysis is correct and that the design is optimal through simulation on a digital computer. The prerequisites for understanding most of topics in this book are rather minimal: an elementary knowledge of probability theory and algebraic matrix operations and a basic familiarity with the computer.

Achieving the right balance between the depth and breadth of each chapter subject has been a difficult task. Readers may recognize that full coverage of each chapter title could fill its own a separate book of substantial volume. An attractive feature of this book is that it allows readers to build a solid foundation and increase their level of sophistication as their experience grows.

This book is intended for those readers who have an impatient urge to learn radar system analysis and design and C++ programming with applications. All examples are explained in detail in the text, and the numerical results are either displayed on screen or stored in .DAT files and shown in graphic form when appropriate. This book is not C++ for dummies; this book is for smart students and diligent engineers who are overloaded with other burdens.

So let's get started! The mountaintop is not that far away, and the path to be traversed to reach the top is not too arduous a climb.

Acknowledgments

This introductory book will help readers to build a solid foundation for further explorations in their chosen topics.

Many friends and colleagues helped me and encouraged me to complete the manuscript. I owe them a large debt of gratitude. I wish I could acknowledge them all by name, but they are too numerous.

I am most grateful to Mrs. Mary K. Darcy who patiently taught me how to type the mathematic equations clearly and concisely. Her indefatigable tutoring in presenting the graphs and diagrams made this book stand apart from all others with a similar title.

Introduction

Chapter 1: Matrix, Vector, and Linear Equations

Chapter 1 introduces matrix-vector operations. We start with how to solve a set of simultaneous linear equations by Gauss's elimination methods of the back-substitution and forward substitution that we learned in earlier years in school. When the dimension of equations is two or three we can solve for the unknowns with pencil and paper. When the dimension is higher than, say, four or higher we would rather resort to a computer program.

We encounter immediately the task of matrix inversion. Matrix inversion is usually permissible, but sometime it is not, depending upon the structure of the matrix. We learn when it is permissible by means of matrix factorization (decomposition). The factored matrices would clearly indicate whether the inversion is permissible. Several popular factorization routines and the corresponding inversions of factored matrices are programmed.

Vector operation is treated as a subset of matrix operations. Two header files, VECTOR.H and MATRIX.H, are constructed in order to determine when the matrix-vector operations are called for in the main driver.

Chapter 2: Pseudorandom Number (PRN), Noise, and Clutter Generation

The goal of this chapter is to generate various noises and clutters we would encounter in signal processing. The noise and clutter are characterized by the probability density function and its mean and variance. (For those who prefer electrical terms mean is DC voltage level and variance is AC power.) The basic building components of noise and clutter are unit uniform random variables, and the unit uniform variables are, in turn, generated from pseudo-random numbers (PRNs). There are a few prepackaged random number generators; however, careless use of these generators causes some intractable confusion in the results of signal processing.

We start with PRN generation by mixed congruential method. This method generates a random number vector without missing data nor duplicated data in a specified population N after a correction. It is contiguous in the given range. Some of the off-the-shelf packages seldom meet the requirement of contiguity.

The typical noise and clutter we encounter most often in signal processing, such as Gaussian, Rayleigh, Rician, exponential, and chi-squared, and lognormal, Weibull clutters, are generated. The computation of noise power or clutter power is emphasized. These noises or clutters are used in applications programs in such areas as filter design, pulse compression, Kalman filters, and the Monte Carlo technique.

Chapter 3: Filters, FIR, and IIR

In this chapter we program filter designs. Filters are broadly classified into two groups: finite duration impulse response filters, and infinite duration impulse response filters. The word duration is deleted, and we call them the finite impulse response (FIR) filters, or infinite impulse response (IIR) filters for short.

The design of a FIR filter is based on the inverse Fourier transform of the impulse response of lowpass, highpass, bandpass, and bandstop. The structure of the FIR filter is nonrecursive, and the phase response is a linear function of frequency. The design of the IIR filter is based on the analog filter designs abundantly available in textbooks and tables. The analog frequency response is transformed to the z-domain using bilinear transform and prewarping the response. The structure is recursive, and the phase versus frequency is nonlinear.

Several window functions (the envelope functions) to control the level of sidelobes are described in detail and incorporated in the design, and the advantages and disadvantages of FIR and IIR are discussed.

Chapter 4: Fast Fourier Transform (FFT) and IFFT

The real-time processing of Fourier transforms became possible when the fast Fourier transform algorithm was discovered in the early 1960s. Prior to this discovery the "real-time on-line" computation of frequency content had been impractical.

The fast Fourier transform is implemented either by decimation-in-time (DIT) or decimation-in-frequency (DIF) algorithm, with a suitable bit-reversal operation. The proper sequence of the bit-reverse operation is shown in a block diagram to show the proper order of bit-reversal in DIT and DIF.

An improper sampling of signals would produce spectral leakage and unfaithful reproduction of time function. A detailed discussion to remedy these problems are given, and examples are shown.

Applications of the fast Fourier transform and inverse transform in signal processing are numerous and include determining how to extract a signal buried in noise, how to compress a frequency-modulated pulse to improve resolution, and how to correct an unbalanced and mismatched I/Q channels of a coherent receiver. We present a few of these interesting applications.

A clear explanation together with a system block diagram will help readers to understand the concept involved. The results of programs are stored in .DAT files, and the corresponding graphs are shown.

Chapter 5: Ambiguity Function

This chapter is written for those who would like to explore the pulse compression in depth. We start with a rectangular pulse with constant carrier frequency and learn basic characteristics of a simple pulsed signal.

A popular signal for pulse compression is a coherent linear frequency-modulated (CLFM) pulse. Most high-performance radar systems have adopted the CLFM pulse, and the returned signal is compressed via FFT and IFFT algorithms in real time.

Analysis of the ambiguity function has indicated that there are many more modulation schemes to increase compression gain, to reduce sidelobe level, to improve resolution, and to maintain the low probability of intercept.

One ideal modulated pulse is the Costas-coded frequency-hopping modulation. We have programmed the Costas-coded signal and concluded that the Costas-coded signal is very nearly an ideal pulse. The correct sequence of frequency hopping is analyzed, and the result is presented in an ambiguity surface in three dimensions.

Chapter 6: Array Antennas

This chapter presents the array antenna design. A simple line array antenna is analyzed and programmed. A circular or elliptical array design is an extension of a line array design. The amplitude distributions of array elements are controlled by window functions such as Hamming, Chebyshev, Taylor, and Lambda. The maximum gain, 3-dB beamwidth, and the sidelobe levels are compared with a rectangle window. A similarity between the window functions in filter design is mentioned. A monopulse array antenna, an essential component of a tracking radar system, is followed.

Elimination (or cancelation) of mutual reactance coupling among array elements is excluded from the discussion, for the coupling is entirely dependent upon the physical structure of the element of radiation source and the geometric distribution of elements on the array aperture.

This chapter does not cover the intricacies and difficulties in designing and manufacturing an array antenna. However, it provides, under an ideal condition, what we expect of the maximum gain, the beamwidth, and the sidelobe levels when the dimension of the array aperture and window function are specified. The references are cited for those problems we have not addressed.

Chapter 7: Target Detection

This chapter presents the theory of the probability of detection and the probability of false alarm. The detection probability and the false alarm probability are introduced heuristically at the beginning in order to put readers at ease.

The detection processing involved in the heuristic model is shown in a recirculating accumulator. The relationship between the detection probability $P_{d,}$ the false alarm probability P_{fa} and the threshold level V_{th} is programmed and discussed. The threshold level V_{th} (or bias level y_b) is a function of the number of pulse N to be accumulated (summed) in the recirculator-accumulator; the larger the number of pulses the lower the V_{th} (or y_b) for a specified P_{fa}.

The mathematic expression for P_{fa} is formally derived as a function of the bias level y_b. An expression of P_d is derived, in turn, as a function of both the number of pulses N and the signal-to-noise ratio S/N (in power) per pulse.

Four factors, P_d, P_{fa}, N, and S/N, maintain an interlocked relationship; that is, if we specify any combination of three factors, the fourth will be determined uniquely. One remaining problem is how to characterize the target signals. The retuned signals are grouped into five Marcum's and Swerling's targets in honor of the pioneer investigators.

Marcum's target is nonfluctuating, a constant cross-section target, like a large metallic sphere of several wavelengths in diameter. A large spherical target reflects a constant power irrespective of the aspect angle of the antenna beam. Two programs are written for Marcum's target, and the results are shown in a table as well as in the following graphic forms:

- Pd_(0)_1.CPP Detection probability, Marcum's target, N=1
- Pd_(0)_N.CPP Detection probability, Marcum's target, N≥2

Swerling has grouped the fluctuating targets in four different types: target model 1, 2, 3, and 4.

The probability density function of the target cross-section is assumed to be Rayleigh-distributed for target models 1 and 2, and the rate of fluctuation is either slow or rapid. Rayleigh-slow is Swerling's target 1, and Rayleigh-rapid is Swerling target 2.

The target cross-section of Swerling's 3 and 4 is assumed to be distributed chi-squared with four degrees of freedom, and the fluctuation is either slow or rapid. Slow or rapid fluctuation is a relative term with respect to the dynamics of target and the radar operation parameters such as transmitting frequency, the pulse repetition frequency (f_{PRF}), and the antenna rotation rate.

Eight programs are written for Swerling's targets. The false alarm probability is set at 1.0E-6, adjustable to 1.0E-5. The detection probability is computed as a function of S/N per pulse with the number of pulse N integrable as a variable parameter. The eight programs are listed as follows:

- Pd_(l)_1.CPP: Swerling target 1, N=1;
- Pd_(l)_N.CPP: Swerling target 1, N=2, 4, 8, 16, . . . ;
- Pd_(2)_1.CPP: Swerling target 2, N=1;
- Pd_(2)_N.CPP: Swerling target 2, N=2, 4, 8, 16, . . . ;
- Pd_(3)_1.CPP: Swerling target 3, N=1;
- Pd_(3)_N.CPP: Swerling target 3, N=2, 4, 8, 16, . . . ;
- Pd_(4)_1.CPP: Swerling target 4, N=1;
- Pd_(4)_N.CPP: Swerling target 4, N=2, 4, 8, 16, ...

The noise is for all programs Gaussian with zero mean and unity variance, which transformed to exponential after a square-law detector as presented in Chapter 2.

An example of a preliminary system design is given at the end. For instance, when we specify the transmitter frequency and power, the antenna gain, the hardware

plumbing loss the RF section and the noise figure, the predetection bandwidth of the receiver, the number of pulses N integrable, at what distance might we be able to detect a target of $10m^2$ with a detection probability of 90% (or 50%) if the target is a nonfluctuating Marcum's target? We answer this question clearly. The detection range for the fluctuating Swerling's targets can be deduced from the programs given above.

The detection probability when the Gaussian noise is replaced by clutter returns is separately analyzed and programmed in Chapter 10, since the detection probability under clutter intrusion requires an additional theoretical development and processing implementation.

Chapter 8: Kalman Filter

We define the symbols and notations used in this chapter at the outset. This is to eliminate the confusion and frustration that might stem from reading a few textbooks and research papers by various authors.

The bold letters represent vectors or matrices. Some vectors and matrices have an overhead symbol, a caret (^) or a tilde (~). The caret signifies that the vector or matrix is an estimate. The tilde, on the other hand, signifies the difference between an estimate and the true state. In addition, the vector and matrix have subscript k–1, k, or k+1. The k–1 connotes the vector or matrix is a state of immediate past, the k the present. The k+1 is a predicted state, one sample time in the future. The vector or matrix without a subscript is time-invariant, and the vector or matrix without an overhead symbol is the true state, unknowable to the observer.

Example 1

We derive seven equations of Kalman filter through an example. A simple example of Kalman filter processing is a radar system that tracks a passenger airliner whose flight trajectory is a straight line at a constant altitude without any abrupt maneuver. We assume that the airliner experiences a small random deviation from an ideal straight line due to atmospheric disturbance and nonuniform engine thrust. We define the state equation of the airliner and the measurement (observation) equation by a radar. The atmospheric disturbance and nonuniform thrust are assumed to be small in magnitude, and the probability distribution is Gaussian. Two sources of measurement errors are due to finite transmitter pulsewidth (range error) and finite antenna beamwidth (azimuth error). The probability density function of the measurement errors is a unit uniform; that is, the error is most likely distributed equally and uniformly. Numerical examples are given whenever a new vector or matrix is introduced.

Three programs are written: an equation of a straight-line trajectory in x-y coordinates; a Gaussian deviation of zero mean and variance of 0.5g, which is added to the straight line to simulate the actual flight path; and the measurement equation by the radar. The estimated and predicted error covariance matrices $\mathbf{P}_k^+$ and $\mathbf{P}_{k+1}^-$ are derived as well as the Kalman gain matrix $\mathbf{k}_k$. The estimated state and predicted state, $\mathbf{x}_k^+$ and $\mathbf{x}_{k+1}^-$ (the position of airliner) are derived, with the Kalman gain matrix $\mathbf{k}_k$ as an input.

The filter processing is shown in a flow diagram. The filter has two recursive loops; one computes the estimated and predicted error covariance matrices, and the other loop computes the estimated and predicted states. The two loops are connected by a Kalman gain matrix.

A post-flight analysis is programmed to evaluate the performance of the Kalman filter. The error between the true trajectory and the estimated and the predicted positions of the airliner are presented for discussion. An improved initialization of the error covariance matrix and that of the state vector are discussed.

Example 2

The second example is air traffic control (ATC) radar. There are two types of ATC radars: airport surveillance radar (ASR) and air route surveillance radar (ARSR). The former is for a short range, the latter for a longer range. An ASR system with a Kalman tracking filter is programmed in the line-of-sight (LOS) coordinates while an airliner approaches an airport.

The transmitter pulsewidth (0.83 μs) and the antenna beamwidth (1.5 degrees), the sources of range measurement error, and the azimuth error are incorporated in the numerical example to demonstrate how a practical Kalman filter really works.

The elements of estimated and predicted error covariance matrices are in terms of LOS coordinates, as are the estimated and predicted airliner's state. A post-flight error analysis is programmed, and the results are presented in graphic form. The advantages and disadvantages of formulating the filter in LOS coordinates system are discussed.

Example 3 and 4

The third and fourth examples are that of a short-range ground-based air defense radar; one operates in the LOS coordinates and the other in the Cartesian coordinates system (CCS). The target is a fighter-bomber on a ground-attack mission with a "turn-dive-attack and turn-climb-escape" maneuver with a maximum acceleration of ± 3g. The random acceleration noise matrix $\mathbf{Q}_k$ and the measurement error matrix $\mathbf{R}_k$ are derived, and the coordinate transformation between LOS and CCS is given. The initialization of the estimate of the error covariance matrix $P_k(k=2)$ instead of $P_k(k=0)$ is discussed in detail.

The tracking performances in the LOS and CCS systems are presented for comparison. The recursive processing sequences are shown in a block diagram.

Example 5

The fifth example emphasizes the problems associated with a matrix inversion, and a Kalman filter without a matrix inversion is presented. We note the special characteristic of the error covariance matrix P_k; it is always symmetric and positive definite. The covariance matrix is factored (decomposed), and we recognize that an inversion is equivalent to a scalar division after factorization.

The Kalman filter without a matrix inversion is programmed. The target is identical to example 3 or 4. The result shows that the difference between "with inversion" and "without inversion" is practically nil. The computation load of no-inversion is, however, slightly higher.

A block diagram of "no-matrix-inversion" is given. The basic processing sequences are not altered; they are two recursive loops, one for the error covariance matrices and the other for state vectors, connected by Kalman gain matrix.

Chapter 9: Monte Carlo Method and Function Integration

In this chapter we learn the basic principles of the Monte Carlo method through examples. We begin with the most simple technique of the "hit-or-miss" method. From this method we learn that the number of sampled data must be on the order of tens of thousands or more to obtain an acceptable level of precision.

The hit-or-miss method is followed by the "ordered sample method" and the "sample mean method" to reduce the number of replications. We present the subject of the coefficient of dispersion (CD) to determine the replication number required for a desired level of precision. The last method we study is the "importance sampling method," a most popular technique among researchers in the field. Mathematics involved in the importance sampling are derived in detail, and two examples are given.

This chapter concludes with several function integration routines found in the elementary calculus books, since the Monte Carlo technique is a numerical experimental mathematic process of integration and a probabilistic determination of a rare event occurrence out of a large number of trials.

Chapter 10: Constant False Alarm Rate (CFAR) Processing

Equations are derived to show the interlocking relationship among the probability of detection, the probability of false alarm, the threshold level, and the number of samples stored in the reference window.

We entertain a scenario of radar in surveillance mode when an antenna sweeps from one sector to another. The antenna will receive different noise or clutter power locally varying. The threshold (or bias level) must be adjusted automatically to maintain the false alarm at a constant level. In this chapter we derive mathematic equations for the threshold when statistically stationary and uniformly distributed noise or clutter is encountered. The crux of the solution is an accurate estimate of the noise or clutter power.

Example 1: CA-CFAR

We present the "cell-average" technique in estimating the noise power in detail. The noise is assumed to be Gaussian-distributed with zero mean and unknown variance. CA-CFAR processing is simple in concept and easy to implement; however, it has a sluggish response to a rapid change in noise level. The CA-CFAR incurs a processing loss inversely proportional to the number of reference cells employed in estimating the noise power. The loss in decibels as a function of the number of reference cells is derived. The most detrimental to a successful operation is the absence

of an adequate means to deal with multiple targets in the reference cells, since these spurious targets would raise the threshold erroneously higher.

Example 2: OS-CFAR
We investigate a CFAR technique with a censoring mechanism, called "order-statistics" (OS) processing, or OS-CFAR for short, to handle the multiple-target situation. The received noise data stored in the reference window is rank-ordered (rearranged) by increasing order of magnitude, and we censor (discard) one, two, three, or four of the highest data in estimating the noise power. The underlying principle comes from the theory of order statistics. The distribution of noise after the censor has been known to us. We have applied the theory to the problem.

We have derived the equations of the false alarm probability, the threshold multiplication factor T, and the signal-to-noise (S/N) ratio (in power) per pulse. The results indicate that OS-CFAR has eliminated the problem of multiple targets in the reference window with a small fraction of additional processing loss. We also found that the response to a sudden change in noise level is faster than that of CA-CFAR. The OS-CFAR offers the most robust processing, provided that the noise (or clutter) is Gaussian-distributed, variance-unknown.

CFAR processing in clutter. The reflected power from the land and sea are reported to be more likely distributed as log-normal or Weibull rather than a benign Gaussian. The log-normal distribution is characterized by tall spiky waveforms. The probability density function has the longest tail.

The Weibull is a two-parameter distribution: the shape parameter c and the scale parameter b. Depending on the shape parameter, the Weibull probability density function may be an exponential, or Rayleigh, or a Gaussian look-alike with a small skew to the right. Weibull is a versatile function to characterize the clutter returns that vary spatially or temporally.

We study some statistics of Weibull distribution (i.e., mean, variance, moments, median, mode, and quantiles). Weibull clutters that pass through a square-law detector remain Weibulls, though the shape parameter is halved and the scale parameter is squared. We take advantage of this unique transformation of Weibull in CFAR processing.

Example 3: Weber-Haykin CFAR, WH-CFAR
Weber and Haykin proposed CFAR processing in Weibull clutter. We follow their analysis and present a few numerical results, since they have not provided any. The task is to estimate the two parameters simultaneously from the clutter returns stored in the reference window. The false alarm probability and the threshold are programmed. We present two results with uncensored and censored algorithms. The processing losses are compared with CA-CFAR and OS-CFAR, and we found that WH-CFAR exacts a very high loss in Weibull clutter.

Example 4: Maximum Likelihood CFAR, ML-CFAR
The principle of maximum likelihood estimation is applied to CFAR processing. Mathematic expressions for an estimate of the shape parameter and the scale parameter are given. An estimate of the shape parameter can only be obtained through

an iterative procedure since the expression is in a transcendental form. An estimate of scale parameter is obtained by using the iteratively estimated shape parameter.

The shortcoming of the maximum likelihood estimate of the shape parameter is a large biased result when the number of sampled data (the stored clutter samples in the reference window) is relatively small (i.e., 8, 16, or 32).

The biased estimate must be corrected before computing the scale parameter.

The bias is negligible when the samples are larger than 100. (We rarely have a window length of 100.) We have programmed ML-CFAR without a censor and found that the loss is less than that of WH-CFAR; however, it is doubtful that real-time on-line implementation due to the computation burden imposed by the iterative procedure.

Example 5: Minimum Mean Square Error CFAR, MMSE-CFAR

To avoid the laborious iterations in ML-CFAR we apply the minimum mean square error analysis to estimate the shape and scale parameter of Weibull-distributed clutters. The received clutter data stored in the reference window are rank-ordered (rearranged) in ascending order of magnitude. The rank-ordered clutter samples in a natural logarithm is a linear function of the parameters with small deviations from a straight line. A regressive straight line with the minimum mean squared error discloses two parameters; the slope of the straight line is an estimate of the shape parameter, the intercept point an estimate of the scale parameter. The computations involved are simple enough for a real-time implementation.

Chapter 11: Moving Target Indicator (MTI)

A moving target indicator (MTI) is one of perhaps half a dozen indispensable signal processing methods for the successful operation of a surveillance or tracking radar. A MTI is designed on the principle of Doppler frequency detection (the phase differential with respect to time). All moving objects produce Doppler frequency proportional to the relative velocity of the target and the observer.

We present the implementations of recursive and nonrecursive delay line cancelers of various orders and discuss their performance. We mention the origin of blind speed and design approaches to mitigate it by staggering the pulse repetition frequency, f_{PRF}. An example of ATC radar is given for stagger management.

The definition of clutter attenuation (CA) and the improvement factor (I) in decibels is given to evaluate the performance of various MTI filters. The clutter attenuation and the improvement factor would deteriorate when the clutter distribution has a nonzero mean velocity and a broader variance. The system instabilities also cause deterioration. Two examples are given.

Chapter 12: Miscellaneous Program Routines

This chapter has collected two scores of C++ programs that do not fit with the chapter title or with the program routines in preliminary preparation for the main program. I hope that readers will benefit from the collections in this chapter.

CHAPTER 1

Matrix, Vector, and Linear Equations

1.1 Introduction

Matrix and vector operations have wide applications in everyday engineering problems, yet the courses offered at the college and university level have become too abstract. Thus many students struggle with the abstract concepts, and seldom have opportunities to apply them to problem solving.

We know how to solve a set of simultaneous linear equations by Cramer's rule but when the dimension of the equations gets larger, a pencil and paper is far from adequate, and a solution by computer algorithm is in order.

Commercially available software packages at the elementary, intermediate, and advanced levels continue to grow in sophistication; however, students and engineers learn how to use these convenient tools but not how to construct a solution algorithm.

We give a quick review on the solutions of a set of simultaneous linear equations by Gaussian elimination with backsubstitution and forward substitution and rapidly culminate to a construction of matrix header file. The header file contains almost all the matrix operations required in engineering and general science problems.

We do not claim that our matrix header file is encyclopedic; however, it is more than adequate for the matrix operations covered in this textbook.

1.2 Simultaneous Linear Equation

An example of a set of simultaneous linear equations is given below.

$$\begin{cases} a_{00}x_0 + a_{01}x_1 + a_{02}x_2 = b_0 \\ a_{10}x_0 + a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{20}x_0 + a_{21}x_1 + a_{22}x_2 = b_2 \end{cases} \tag{1.1}$$

where a_{ij} and b_i are given

x_i are unknown, to be solved.

Eq (1.1) can be written in a vector-matrix form,

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

A more compact form is,

$$\mathbf{AX} = \mathbf{B} \tag{1.2}$$

We want to solve for the column vector **X** given a matrix **A** and a column vector **B**. There are two ways to solve for **X**; one is using Cramer's rule and the other by the elimination technique. When the dimension of **A** is fairly large the determinant and cofactors required for Cramer's rule become very involved and tedious. The elimination technique is deceptively simple and straightforward.

Before we study the elimination technique we have to delve into the so-called consistency of (1.2). An illustration shows what we mean by consistency.

Consider the following simultaneous equations.

$$\begin{cases} x_1 + x_2 = 5 \\ 2x_1 + 2x_2 = 11 \end{cases}$$

Obviously if one is true the other cannot be, since the second equation is incompatible or inconsistent with the first one. The inconsistency is visually apparent in this simple case. Consider the following simultaneous equations. A visual inspection for consistency is difficult if not impossible.

$$\begin{cases} 2x_0 + 3x_1 + x_2 = 14 & [1] \\ x_0 + x_1 + x_2 = 6 & [2] \\ 3x_0 + 5x_1 + x_2 = 10 & [3] \end{cases} \tag{1.3}$$

When we do the following operation, we find that 0 = 12.

$$2\cdot[1] - \{[2]+[3]\}$$

Equation (1.3) is inconsistent; therefore, we do not have a solution for vector **X**.

Imagine that the dimension of **A** is larger than, say ten! The test of consistency involves the rank test on matrix **A**. Numerous good textbooks spend multiple pages on the definition, theorem, and proof of the rank test. Beginners would immediately loose interest, become discouraged, and abandon the matrix theory—a tragic loss.

We dispense with a lengthy dissertation on the rank test. Instead we recommend a test on the determinant of **A** for consistency. If the determinant is nonzero, the simultaneous equations are consistent. Readers may protest that the computation of the determinant is as difficult as the rank test. We therefore strive for a method of computation of the determinant as simple as practicable through matrix factorization and simpler inversion of the factorized (decomposed) matrices. First we study how to solve simultaneous linear equations by the elimination technique.

1.2.1 Gaussian Elimination with Backsubstitution

We plan to solve the following simultaneous linear equations by backsubstitution. The consistency test is, for the moment, assumed to be satisfied.

$$\begin{cases} 2x_0 + x_1 + 3x_2 = 11 & [1] \\ 4x_0 + 3x_1 + 10x_2 = 28 & [2] \\ 2x_0 + 4x_1 + 17x_2 = 31 & [3] \end{cases} \tag{1.4}$$

The following table describes the steps taken for back-substitution.

Step taken	a_0	a_1	a_2	b_i	*[eq]*
	2	1	3	11	[1]
	4	3	10	28	[2]
	2	4	17	31	[3]
[1]	2	1	3	11	[4]
[2] – 2[1]	0	1	4	6	[5]
[3] – [1]	0	3	14	20	[6]
[4]	2	1	3	11	[7]
[5]	0	1	4	6	[8]
[6] – 3[5]	0	0	2	2	[9]

When we write equations [7]–[9] appear as follows:

$$\begin{aligned} 2x_0 + x_1 + 3x_2 &= 11 \quad [7] \\ x_1 + 4x_2 &= 6 \quad [8] \\ 2x_2 &= 2 \quad [9] \end{aligned}$$

From equation [9] we obtain x_2=1. The result is substituted into equation [8], and we obtain x_1=2. The results are again substituted into equation [7], and we obtain x_0=3. The substitution is always backward: x_2 first, x_1 next, and x_0 last. We note that the original square matrix **A** is transformed to an upper triangle matrix **U**, and the column vector **B** is altered.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \rightarrow \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ & a'_{11} & a'_{12} \\ & & a''_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b'_1 \\ b''_2 \end{bmatrix}$$

A simple deduction gives the elements of an upper triangle matrix **U** in terms of the original matrix **A**: finding the relationship between the primed elements in **U** in terms of the elements in **A**. (The first row of **A** and the first element of **B** are unchanged). The conversion of **A** to **U** is given by,

(intermediate steps)

$$\begin{cases} a'_{11} = a_{11} - (a_{10}/a_{00})a_{01} \\ a'_{12} = a_{12} - (a_{10}/a_{00})a_{02} \\ b'_1 = b_1 - (a_{10}/a_{00})b_0 \end{cases} \begin{cases} a'_{21} = a_{21} - (a_{20}/a_{00})a_{01} \\ a'_{22} = a_{22} - (a_{20}/a_{00})a_{22} \\ b'_2 = b_2 - (a_{20}/a_{00})b_0 \end{cases} \begin{cases} a''_{22} = a'_{22} - (a'_{21}/a'_{11})\,a'_{12} \\ b''_2 = b'_2 - (a'_{21}/a'_{11})\,b'_2 \end{cases}$$

We note that the quotient multipliers (a_{ij}/a_{jj}) are undefined when the denominator a_{00} or a'_{11} is zero. When a_{00} or a'_{11} is zero, the backsubstitution method fails. There are many simultaneous linear equations that may have a null element in the diagonal position, yet must have a perfect solution. We rearrange a raw of matrix **A** so as to move the null element to an off-diagonal position, or, we transform **A** to a lower triangle matrix **L**. We shall cover the transform of **A** to **L** in the next section.

A demonstration program is written in GAUSSBCK.CPP to solve (1.4) by the Gaussian backsubstitution technique (**A** to **U**). The solution of **X** is

$$\mathbf{X} = [x_0, x_1, x_2]^T = [3, 2, 1]^T$$

The key lesson in this section is a transform of square matrix **A** to an upper triangle matrix **U**. Read the footnotes in GAUSSBCK.CPP in the file.

1.2.2 Gaussian Elimination with Forward Substitution

Simultaneous linear equations can be solved by forward substitution as well.

$$\begin{cases} a_{00}x_0 + a_{01}x_1 + a_{02}x_2 = b_0 \\ a_{10}x_0 + a_{11}x_1 + a_{12}x_2 = b_1 \\ a_{20}x_0 + a_{21}x_1 + a_{22}x_2 = b_2 \end{cases}$$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \rightarrow \begin{bmatrix} a''_{00} & & \\ a'_{10} & a'_{11} & \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b''_0 \\ b'_1 \\ b'_2 \end{bmatrix}$$

The elements of lower triangle matrix **L** can be expressed in terms of the elements of the original matrix **A**. They are:

$$\begin{cases} a'_{00} = a_{10} - (a_{12}/a_{02})a_{00} \\ a'_{01} = a_{11} - (a_{12}/a_{02})a_{01} \\ b'_0 = b_1 - (a_{12}/a_{02})b_0 \end{cases} \begin{cases} a'_{10} = a_{20} - (a_{22}/a_{12})a_{10} \\ a'_{11} = a_{21} - (a_{22}/a_{12})a_{11} \\ b'_1 = b_2 - (a_{22}/a_{12})b_1 \end{cases} \begin{cases} a''_{00} = a'_{10} - \left(a'_{11}/a'_{01}\right)a'_{00} \\ \\ b''_0 = b'_1 - \left(a'_{11}/a'_{01}\right)b'_0 \end{cases}$$

We note that the pivot elements a_{02}, a_{12} and a'_{01} in the denominator of the quotient multipliers must not be zero. Should that happen, a row of **A** should be rearranged as mentioned earlier.

The Gaussian elimination technique with forward substitution is programmed in GAUSSFWD.CPP. The solution for **X** is same as in GAUSSBCK.CPP.

We have demonstrated that simultaneous linear equations can be solved by matrix transform from a square matrix **A** to either an upper triangle matrix **U** or a lower triangle matrix **L**, backsubstitution or forward substitution. The computation of determinant and adjoint matrices is not needed as in the Cramer rule.

In the next section we learn how to eliminate the computation of the altered vector **B** through matrix factorization (decomposition).

1.3 Matrix Factorization

In this section we learn how to solve linear simultaneous equations through various matrix factorizations. For Gaussian elimination methods we have to transform a square matrix **A** to either an upper triangle matrix **U** or a lower triangle matrix **L**, recompute the column vector **B**, and apply a backsubstitution or forward substitution to solve for unknown column vector **X**.

We shall eliminate the recomputation of the column vector **B** through matrix factorization and take advantage of the special properties of the factorized matrices to compute the determinant and the inverse of **A**.

1.3.1 LU Factorization

Given a set of linear simultaneous equations,

$$\mathbf{AX} = \mathbf{B} \tag{1.5}$$

The square matrix **A** is factored into a unit lower triangle matrix **L** and an upper triangle matrix

$$\mathbf{A} = \mathbf{LU} \tag{1.6}$$

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} 1 & & \\ l_{10} & 1 & \\ l_{20} & l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{00} & u_{01} & u_{02} \\ & u_{11} & u_{12} \\ & & u_{22} \end{bmatrix} \tag{1.7}$$

$$\mathbf{AX} = \mathbf{B}$$

$$[\mathbf{LU}]\mathbf{X} = \mathbf{B} \tag{1.8}$$

Regrouping (1.8),

$$\mathbf{L}[\mathbf{UX}] = \mathbf{B} \tag{1.9}$$

let

$$[\mathbf{UX}] = \mathbf{Y} \tag{1.10}$$

we have

$$\mathbf{LY} = \mathbf{B} \tag{1.11}$$

The unknown column vector **Y** can be solved by forward substitution. The solution for **Y** in hand, we solve (1.10) for the column vector **X**. The column vector **B** remains unchanged throughout the operations. All we have to do is factorization of **A** into **L** and **U**.

From (1.7) we identify l_{ij} and u_{ij} in term of a_{ij} by multiplying **L** and **U**.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} u_{00} & u_{01} & u_{02} \\ l_{10}u_{00} & l_{10}u_{10}+u_{11} & l_{10}u_{20}+u_{12} \\ l_{20}u_{00} & l_{20}u_{01}+l_{21}u_{11} & l_{20}u_{22}+l_{21}u_{12}+u_{22} \end{bmatrix}$$

We have,

$$\begin{cases} a_{00} = u_{00} \\ a_{01} = u_{01} \\ a_{02} = u_{02} \end{cases}$$

$$\begin{cases} a_{10} = l_{10}u_{00} \\ a_{11} = l_{10}u_{10} + u_{11} \\ a_{12} = l_{10}u_{20} + u_{12} \end{cases}$$

$$\begin{cases} a_{20} = l_{20}u_{20} \\ a_{21} = l_{20}u_{01} + l_{21}u_{11} \\ a_{22} = l_{20}u_{22} + l_{21}u_{12} + u_{22} \end{cases}$$

A fragment of coding in C++ for the factorization is shown below.

```
for(i=0; i<m; i++)
  for(j=i; j<m; j++)
    for(k=0; k<i; k++)
      {
        u[i][j]=a[i][j]+ ∑k l[i][j]*u[k][j]

        l[i][j]=a[i][j]+ ∑k l[j][k]*u[k][j]
      }
```

There are two different **LU** factorizations, Doolittle's and Crout's. Doolittle's algorithm makes all the diagonal elements of **L** unity, whereas Crout's algorithm makes all the diagonal elements of **U** unity. There is no difference in computation speed nor memory requirement but a minor index adjustment.

A demonstration program is written in LU_FCTR.CPP.

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 2 & 5 \\ 4 & 9 & 2 & 11 \\ 2 & 0 & 2 & 3 \\ 8 & 17 & -10 & 26 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 1 & -4 & 1 & \\ 4 & 1 & 2 & 1 \end{bmatrix} \quad \mathbf{U} = \begin{bmatrix} 2 & 4 & 2 & 5 \\ & 1 & -2 & 1 \\ & & -8 & 2 \\ & & & 1 \end{bmatrix}$$

The solution for $\mathbf{X} = [\,1, 3, 5, 7]^T$

1.3.2 LL^T Factorization (Cholesky)

A square matrix **A** can be factored to a lower triangle matrix **L** and its transpose $\mathbf{L}^T$ provided that **A** is symmetric and positive definite.

$$\mathbf{A} = \mathbf{L}\mathbf{L}^T \tag{1.12}$$

We encounter symmetric and positive definite matrix frequently in engineering problems—error covariance matrix in the Kalman filter, for example. Compared with **LU** factorization, Cholesky factorization needs half as many computations. This is a major advantage. The disadvantage is that it requires "sqrt" operations where a round-off error may become problematic. We shall mitigate the "sqrt" operations in the so-called modified Cholesky factorization later.

$$\mathbf{A}\mathbf{X} = \mathbf{B} \tag{1.13}$$

$$\mathbf{L}\mathbf{L}^T\mathbf{X} = \mathbf{B} \tag{1.14}$$

Let

$$\mathbf{L}^T\mathbf{X} = \mathbf{Y} \tag{1.15}$$

Then

$$\mathbf{L}\mathbf{Y} = \mathbf{B} \tag{1.16}$$

Solve for **Y** in (1.16) through forward substitution. In turn, solve for **X** in (1.15) by backsubstitution since $\mathbf{L}^T$ is an upper triangle.

We note that the diagonal elements involve squared terms. Equating elements a_{ij} to the corresponding elements of $\mathbf{L}\mathbf{L}^T$, we obtain,

$$\begin{cases} a_{00} = l_{00}^2 \\ a_{10} = l_{10}l_{00} \\ a_{20} = l_{20}l_{00} \end{cases} \qquad \begin{cases} l_{00} = (a_{00})^{1/2} \\ l_{10} = a_{10}/l_{00} \\ l_{20} = a_{20}/l_{00} \end{cases}$$

$$\begin{cases} a_{11} = l_{10}^2 + l_{11}^2 \\ a_{21} = l_{20}l_{10} + l_{21}l_{11} \end{cases} \qquad \begin{cases} l_{11} = (a_{11} - l_{10}^2)^{1/2} \\ l_{21} = (a_{21} - l_{20}l_{10})/l_{11} \end{cases}$$

$$a_{22} = l_{20}^2 + l_{21}^2 + l_{22}^2 \qquad l_{22} = \left(a_{22} - l_{20}^2 - l_{21}^2\right)^{1/2}$$

By algebraic deduction, the diagonal elements are given by,

$$l_{ii} = \left[a_{ii} - \sum_{k=0}^{i-1} (l_{ik})^2\right]^{1/2} \tag{1.17}$$

and the off-diagonal terms are given by,

$$I_{ij} = \frac{1}{I_{jj}} \left[a_{ij} - \sum_{k=0}^{j-1} (l_{ik} l_{ik}) \right]^{1/2} \tag{1.18}$$

A demonstration program is written in LLT_FCTR.CPP using (1.17) and (1.18). Given **A**, a symmetric and positive definitive matrix, we obtain **L** and $\mathbf{L}^T$, and the solution for **X**.

$$\mathbf{A} = \begin{bmatrix} 36 & 30 & 24 \\ 30 & 34 & 26 \\ 24 & 26 & 21 \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} 6 & & \\ 5 & 3 & \\ 4 & 2 & 1 \end{bmatrix} \quad \mathbf{L}^T = \begin{bmatrix} 6 & 5 & 4 \\ & 3 & 2 \\ & & 1 \end{bmatrix}$$

$$\mathbf{X} = [-1 \quad -2 \quad -3]^T$$

In general, the computation of the determinant and inversion of a square matrix involves a lengthy error-prone procedure when the dimension of matrix **A** is relatively large. With $\mathbf{LL}^T$ factorization the computation of the determinant becomes a simple algebraic operation.

$$\begin{aligned} \text{del } \mathbf{A} &= \det(\mathbf{L}\mathbf{L}^T) \\ &= \det \mathbf{L} \cdot \det \mathbf{L}^T \\ &= \prod_i l_{ii} \cdot \prod_j l_{jj} \end{aligned}$$

If any of l_{ii} or l_{jj} is a null the determinant is zero, matrix **A** is singular, and inversion of **A** is not permissible. The consistency test we have mentioned earlier would become a simple visual inspection. When none of l_{ii} and l_{jj} is null, an inversion of **A** is given by,

$$\mathbf{A}^{-1} = (\mathbf{L}\mathbf{L}^T)^{-1} = (\mathbf{L}^T)^{-1} \cdot \mathbf{L}^{-1}$$

The inversion of a lower triangle matrix **L** and its transpose is discussed in Section 1.4.

1.3.3 $\mathbf{LDL}^T$ Factorization (Modified Cholesky)

The "sqrt" operation in $\mathbf{LL}^T$ factorization can be avoided if matrix **A** is symmetric and positive definite. The matrix **A** will be factored as,

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{1.19}$$

The $\mathbf{LDL}^T$ factorization is sometimes called "modified" Cholesky. **L** is a unit lower triangle, and **D** is a diagonal metrix. A set of linear simultaneous equations would be solved by a similar procedure employed previously, by regrouping as shown below.

$$\mathbf{AX} = \mathbf{B}$$

$$(\mathbf{LDL}^T)\mathbf{X} = \mathbf{B} \tag{1.20}$$

Regrouping

$$(\mathbf{LD})(\mathbf{L}^T\mathbf{X}) = \mathbf{B} \tag{1.21}$$

and if we let

$$\mathbf{L}^T\mathbf{X} = \mathbf{Y} \tag{1.22}$$

(1.21) becomes

$$\mathbf{LDY} = \mathbf{B} \tag{1.23}$$

Again, if we let **LD** = **M,** (1.23) becomes **MY** = **B**

Matrix **M** is a lower triangle; therefore, a forward substitution would solve for the unknown column vector **Y**. The solution of **Y** in hand, we solve for **X** in (1.22) by backsubstitution. All we have to do is to find expressions for the elements of **L** and **D** in terms of the elements of **A**.

We shall write out (1.19) and equate the like elements in a_{ij} to the corresponding elements of the product of three matrices.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{10} & 1 & & \\ l_{20} & l_{21} & 1 & \\ l_{30} & l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} d_{00} & & & \\ & d_{11} & & \\ & & d_{22} & \\ & & & d_{33} \end{bmatrix} \begin{bmatrix} 1 & l_{10} & l_{20} & l_{30} \\ & 1 & l_{21} & l_{31} \\ & & 1 & l_{32} \\ & & & 1 \end{bmatrix}$$

By algebraic deduction the following identities are found.

$$d_{ii} = a_{ii} - \sum_{k=0}^{i-1} (l_{ik})^2 d_{kk} \tag{1.24}$$

$$l_{ij} = \frac{1}{d_{jj}} \left[a_{ij} - \sum_{k=0}^{i-1} l_{ik} l_{jk} d_{kk} \right] \tag{1.25}$$

A demonstration program is written in LDL_FCTR.CPP with **A** given.

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 6 & 10 \\ 4 & 11 & 24 & 38 \\ 6 & 24 & 70 & 130 \\ 10 & 38 & 130 & 359 \end{bmatrix}$$

By (1.24) and (1.25) we found **L** and **D**.

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ 2 & 1 & & \\ 3 & 4 & 1 & \\ 5 & 6 & 7 & 1 \end{bmatrix} \quad \mathbf{L}^T = \begin{bmatrix} 1 & 2 & 3 & 4 \\ & 1 & 4 & 5 \\ & & 1 & 6 \\ & & & 1 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} 2 & & & \\ & 3 & & \\ & & 4 & \\ & & & 5 \end{bmatrix}$$

The determinant is the product of diagonal elements of **D**, and the consistency test is easily accomplished visually.

$$\begin{aligned} \det \mathbf{A} &= \det \mathbf{L} \cdot \det \mathbf{D} \cdot \det \mathbf{L}^T \\ &= (1)\,(d_{00} \cdot d_{11} \cdot d_{22} \cdot d_{33})\,(1) \\ &= \prod_i d_{ii} \end{aligned}$$

Inversion of **A** is the product of the inverse of three factored matrices in the reversed order.

$$\mathbf{A}^{-1} = (\mathbf{L}^T)^{-1}\,\mathbf{D}^{-1}\,\mathbf{L}^{-1}$$

We made all diagonals of **L** unity in our demonstration. We could have a lower triangle with nonunity diagonals. This would be a general $\mathbf{LDL}^T$ factorization; however, we don't see any added benefit.

The solution for **X** will be obtained without Gaussian elimination:

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$$

Readers will recognize, the power of factorization in solving linear simultaneous equations.

1.3.4 $\mathbf{UDU}^T$ Factorization

A square, symmetric, positive definite matrix **A** can be factorized into three matrices **U**, **D** and $\mathbf{U}^T$. **U** is an upper triangle matrix with unity diagonals.

$$\mathbf{A} = \mathbf{UDU}^T$$

The procedure to obtain the elements of **U** and **D** is similar to $\mathbf{LDL}^T$ factorization discussed in the previous section.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & u_{01} & u_{02} & u_{03} \\ & 1 & u_{12} & u_{13} \\ & & 1 & u_{23} \\ & & & 1 \end{bmatrix} \begin{bmatrix} d_{00} & & & \\ & d_{11} & & \\ & & d_{22} & \\ & & & d_{33} \end{bmatrix} \begin{bmatrix} 1 & & & \\ u_{01} & 1 & & \\ u_{02} & u_{12} & 1 & \\ u_{03} & u_{13} & u_{23} & 1 \end{bmatrix}$$

Equating the elements of **A** to the corresponding elements of the product of three matrices, we obtain the following identities.

$$d_{jj} = a_{jj} - \sum_{k=0}^{i-1} (u_{jk})^2 d_{kk} \tag{1.26}$$

$$u_{ij} = \left[a_{ij} - \sum_{k=0}^{i-1} u_{ik}\ d_{kk}\ u_{jk} \right] / d_{jj} \tag{1.27}$$

A demonstration program is written in UDU_FCTR.CPP with **A** given.

$$\mathbf{A} = \begin{bmatrix} 130 & 186 & 152 & 20 \\ 186 & 283 & 230 & 30 \\ 152 & 230 & 249 & 30 \\ 20 & 30 & 35 & 5 \end{bmatrix}$$

and found **U** and **D** through (1.26) and (1.27).

$$\mathbf{U} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ & 1 & 5 & 6 \\ & & 1 & 7 \\ & & & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} 2 & & & \\ & 3 & & \\ & & 4 & \\ & & & 5 \end{bmatrix}$$

The determinant is given by the product of the diagonal elements of **D**, and the consistency will be ascertained by visual inspection. The unknown column vector **X** will be obtained without Gaussian eliminations, once we compute the inverse of **A**.

1.3.5 QR Factorization

There are many matrix factorizations in addition to the few we have studied. An interesting one is **QR** factorization.

$$\mathbf{A} = \mathbf{QR} \tag{1.28}$$

where

A: May be a square or rectangular matrix;
Q: All columns must be orthonormal. When **A** is a square **Q** would be orthogonal. The orthogonality implies that,

$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I}, \quad \text{or} \quad \mathbf{Q}^{-1} = \mathbf{Q}^T \tag{1.29}$$

An inversion of **Q** is therefore a transpose of itself.

R: An upper triangle, invertible, and nonzero diagonal.

As with other factorizations we have studied, the **QR** factorization solves for the unknown column vector **X** of,

$$\mathbf{AX} = \mathbf{B} \tag{1.30}$$

Premultiply both sides by $\mathbf{Q}^T$.

$$\mathbf{Q}^T\mathbf{AX} = \mathbf{Q}^T\mathbf{B} \tag{1.31}$$

Since $\mathbf{Q}^T\mathbf{A} = \mathbf{Q}^T\mathbf{QR} = \mathbf{R}$, (1.31) becomes

$$\mathbf{RX} = \mathbf{Q}^T\mathbf{B} \tag{1.32}$$

We note that **R** is an upper triangle that employs Gaussian elimination with back-subsitution to solve for **X**. Better still **X** can be obtained by,

$$\mathbf{X} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{B} \tag{1.33}$$

There are a half dozen different methods to obtain **QR** factorization. They are, listed as follows:

1. Gram-Schmidt method;
2. Modified Gram-Schmidt;
3. Householder method;
4. Block householder;
5. Givens method;
6. Fast Givens.

Each method has certain merit such as numerical stability or fewer computation steps. We illustrate the classic Gram-Schmidt method.

$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} q_{00} & q_{01} & q_{02} & q_{03} \\ q_{10} & q_{11} & q_{12} & q_{13} \\ q_{20} & q_{21} & q_{22} & q_{23} \\ q_{30} & q_{31} & q_{32} & q_{33} \end{bmatrix} \begin{bmatrix} r_{00} & r_{01} & r_{02} & r_{03} \\ & r_{11} & r_{12} & r_{13} \\ & & r_{22} & r_{23} \\ & & & r_{33} \end{bmatrix}$$

All column vectors of **Q** must be orthonormal:

$$q_{i0} = \frac{v_0}{v_0}, \; q_{i1} = \frac{v_1}{v_1}, \ldots\ldots, q_{\;im} = \frac{v_m}{v_m} \tag{1.34}$$

The column vectors v_0, v_1, v_2, . . . , v_m are obtained by the orthogonality principle.

$$
\begin{aligned}
v_0 &= a_0 \\
v_1 &= a_1 - \left(\frac{v_0{}^T a_1}{v_0{}^T v_0}\right) v_0 \\
v_2 &= a_2 - \left(\frac{v_0{}^T a_2}{v_0{}^T v_0}\right) v_0 - \left(\frac{v_0{}^T a_2}{v_1{}^T v_1}\right) v_1 \\
v_3 &= a_3 - \left(\frac{v_0{}^T a_3}{v_0{}^T v_0}\right) v_0 - \left(\frac{v_0{}^T a_3}{v_0{}^T v_0}\right) v_1 - \left(\frac{v_0{}^T a_3}{v_0{}^T v_0}\right) v_2 \\
&\cdot \\
&\cdot \\
&\cdot \\
v_m &= a_m - \left(\frac{v_0{}^T a_m}{v_0{}^T v_0}\right) v_0 - \left(\frac{v_0{}^T a_m}{v_0{}^T v_0}\right) v_1 \cdots\cdots - \left(\frac{v_0{}^T a_m}{v_0{}^T v_0}\right) v_{m-1}
\end{aligned}
\tag{1.35}
$$

A demonstration program is written in QR_FCTR.CPP with a square matrix **A.**

$$
\mathbf{A} = \begin{bmatrix} 5.0 & 7.5 & 8.0 & 7.0 \\ 5.0 & 7.5 & 5.0 & 4.0 \\ 5.0 & 1.5 & 0.0 & 1.0 \\ 5.0 & 1.5 & 3.0 & 2.0 \end{bmatrix}
$$

The orthonormal matrix **Q** is obtained through (1.35), and the upper triangle matrix **R** is computed by,

$$\mathbf{R} = \mathbf{Q}^{-1}\mathbf{A} = \mathbf{Q}^{T}\mathbf{A}$$

Note that **A** has a null element on diagonal position in our example. For the Gaussian elimination technique we have to rearrange the row so as to push the null element to a off-diagonal position. In **QR** factorization the rearrangement is not necessary.

The inversion of **A** is computed by,

$$\mathbf{A}^{-1} = [\mathbf{QR}]^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{-1} = \mathbf{R}^{-1}\mathbf{Q}^{T}$$

The inversion of **R**, a triangle matrix, is much easier than an inversion of the square matrix **A**. A demonstration program shows that

$$\mathbf{Q} = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & -0.5 & -0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & -0.5 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 10 & 9 & 8 & 7 \\ & 6 & 5 & 4 \\ & & 3 & 2 \\ & & & 1 \end{bmatrix}$$

We will present a routine to test whether **A** = **QR** is executed through a header file MATRIX.H shortly.

When **A** is rectangular, we assume that **A**mxn, m>n, the linear simultaneous equations are an over-determined case. An over-determined case has an interesting application in production cost control.

We shall show an example to embellish the hard subject of matrix theory, an example of the everyday application of matrix theory. Suppose the daily performance of a production line is monitored as shown in Table 1.1.

Two unknown parameters in Table 1.1 are the unit cost and the fixed overhead cost. The data collected can be cast as a set of linear simultaneous equations with unknown unit cost as x and the overhead cost as h.

$$3x + h = 5$$

$$5x + h = 5$$

$$7x + h = 7$$

$$10x + h = 9$$

A vector-matrix form is

$$\mathbf{AX} = \mathbf{B}, \qquad \mathbf{A} = \begin{bmatrix} 3 & 1 \\ 5 & 1 \\ 7 & 1 \\ 10 & 1 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x \\ h \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 5 \\ 5 \\ 7 \\ 9 \end{bmatrix}$$

We want to compute the column vector **X** by **QR** factorization of **A**. A demonstration program is written in QR_MMSE.CPP. The result is,

$$v_0 = a_0 = [\,3, 5, 7, 10]^T$$

$$q_0 = \frac{v_0}{v_0} = [\,0.2218,\ 0.3696,\ 0.5175,\ 0.7392\,]^T$$

Table 1.1

	Cost of Production	*Number of Units Produced*
First day	$5.00	3
Second day	$5.00	5
Third day	$7.00	7
Fourth day	$9.00	10

$$v_1 = a_1 - \left(\frac{v_0{}^T a_1}{v_0{}^T v_0}\right) v_0 = \left[\, 0.5902,\ 0.3168,\ 0.0437,\ -0.3661 \,\right]^T$$

$$q_1 = \frac{v_1}{v_1} = [\, 0.7718,\ 0.4145,\ 0.0572,\ -0.4788 \,]^T$$

therefore,

$$\mathbf{Q} = \begin{bmatrix} 0.2218 & 0.7718 \\ 0.3696 & 0.4145 \\ 0.5175 & 0.0572 \\ 0.7392 & -0.4788 \end{bmatrix}$$

$$\mathbf{Q}^T = \begin{bmatrix} 0.2218 & 0.3696 & 0.5175 & 0.7392 \\ 0.7718 & 0.4145 & 0.0572 & -0.4788 \end{bmatrix}$$

$$\mathbf{R} = \mathbf{Q}^T\mathbf{A} = \begin{bmatrix} 13.5277 & 1.8481 \\ 0.0000 & 0.7647 \end{bmatrix}$$

Finally,

$$\mathbf{RX} = \mathbf{Q}^T\mathbf{B}$$

and

$$\mathbf{X} = R^{-1}\mathbf{Q}^T\mathbf{B} = \begin{bmatrix} 0.6168 \\ 2.6449 \end{bmatrix} = \begin{bmatrix} \text{unit cost} \\ \text{overhead cost} \end{bmatrix}$$

The demonstration above is shown graphically in Figure 1.1. The **QR** factorization performed a minimum mean square estimate (MMSE) of x and h. The slope of the dashed line is the unit cost; the intercept point with the y-axis is the overhead cost. The MMSE will be discussed in Chapter 10 in detail.

1.4 Matrix Inversion

In this section we learn how to invert various factored matrices. The matrices obtained by factorization are a lower or upper triangle matrix, their transpose, or a diagonal matrix—never a square. We have mentioned that the inversion of a square matrix is cumbersome and error prone. Proving nonsingularity involved the determinant, and computation of the determinant is not an easy task.

We recommend the inversion of a matrix through factorization. After factorization the test of singularity can be accomplished visually. When the diagonal elements

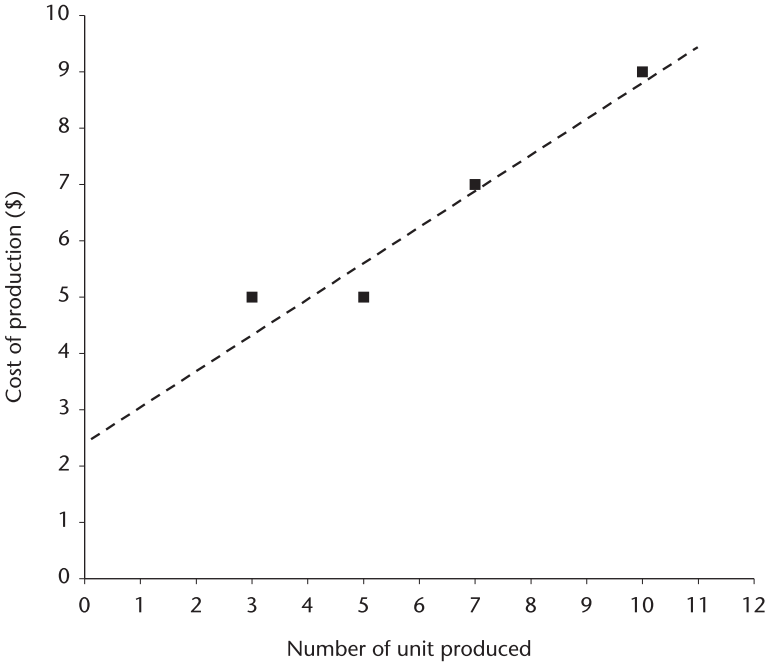


Figure 1.1 QR factorization and minimum-mean-square-error.

of factored matrices are nonzero the determinant of the original matrix is nonzero, and the matrix is nonsingular; thus the matrix is invertible.

Suppose a square matrix **A** is factored into **L** and **U**. The determinant of **A** is a product of the determinant of **L** and **U**.

$$\mathbf{A} = \mathbf{LU}$$

$$\det \mathbf{A} = \det \mathbf{L} \cdot \det \mathbf{U}$$

The determinant of **L** and **U** is a product of diagonal elements since both matrices are triangle. Inversion of **A** is obtained by inverted matrices **L** and **U** and multiplied in reversed order:

$$\mathbf{A}^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}$$

In this section we learn how to invert unit lower matrix $\mathbf{L}_1$, a general lower matrix $\mathbf{L}_x$, a unit upper matrix $\mathbf{U}_1$ and a general upper matrix $\mathbf{U}_x$. We include inversion of a diagonal matrix **D** and a rectangle matrix **Q**.

1.4.1 $\mathbf{L}_1^{-1}$

When a lower triangle matrix has unity diagonal elements, an inversion is a fairly simple task as the following illustration shows.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1_{10}+x_{10} & 1 & 0 & 0 \\ 1_{20}+1_{21}x_{10}+x_{20} & 1_{21}+x_{21} & 1 & 0 \\ 1_{30}+1_{31}x_{10}+1_{32}x_{20}+x_{30} & 1_{31}+1_{32}x_{21}+x_{31} & 1_{32}+x_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$$\begin{matrix} 1_{20}+1_{21}x_{10}+x_{20} & 1_{21}+x_{21} & 1 & 0 & 1 \\ 1_{30}+1_{31}x_{10}+1_{32}x_{20}+x_{30} & 1_{31}+1_{32}x_{21}+x_{31} & 1_{32}+x_{32} & 1 & \quad 1 \end{matrix}$$

Equating the elements on both sides, the identities are obtained.

$$\begin{cases} 1_{10}+x_{10}=0 \\ 1_{20}+1_{21}x_{10}+x_{20}=0 \\ 1_{30}+1_{31}x_{10}+1_{32}x_{20}+x_{30}=0 \end{cases} \quad \begin{cases} 1_{21}+x_{21}=0 \\ 1_{31}+1_{32}x_{21}+x_{31}=0 \end{cases} \quad 1_{32}+x_{32}=0$$

Solving for x_{ij} we have the elements of $\mathbf{L}_1^{-1}$by the identities

$$\begin{cases} x_{10}=-1_{10} \\ x_{20}=-1_{20}-1_{21}x_{10} \\ x_{30}=-1_{30}-1_{31}x_{10}-1_{32}x_{20} \end{cases} \quad \begin{cases} x_{21}=-1_{21} \\ x_{31}=-1_{31}-1_{32}x_{21} \end{cases} \quad x_{32}=-1_{32}$$

Using three integer indices, i, j, k, the inversion of **L**1 is found by

```
for(i=1; i<n; i++)
  for(j=0; j<i; j++)
   {
     L[i][j]= -L[i][j];
     for(k=j+1; k<i; k++)
       L[i][j]=L[i][j]-L[i][k]*L[k][j];
   }
```

A program is written in L(1)_INV.CPP. The original **L** is destroyed or altered. It is wise to reserve additional memory to preserve the original **L** if you wish to test that $\mathbf{L}\mathbf{L}^{-1} = \mathbf{I}$

1.4.2 $\mathbf{L}_x^{-1}$

An inversion of a general lower triangle matrix whose diagonal elements are not unity will be programmed. As we have done for a unit lower triangle we use an identity relationship to compute the inverse.

$$\mathbf{L}_x\,\mathbf{L}_x^{-1}=\mathbf{I}$$

$$\begin{bmatrix} 1_{00} & & & \\ 1_{10} & 1_{11} & & \\ 1_{20} & 1_{21} & 1_{22} & \\ 1_{30} & 1_{31} & 1_{32} & 1_{33} \end{bmatrix} \begin{bmatrix} x_{00} & & & \\ x_{10} & x_{11} & & \\ x_{20} & x_{21} & x_{22} & \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

Multiplying two lower triangle matrices we obtain identities, and the unknown elements x_{ij} are sequentially computed.

$$\begin{cases} x_{00} = 1/l_{00} \\ x_{10} = -1/l_{11}(l_{10}x_{00}) \\ x_{20} = -1/l_{22}(l_{21}x_{10}+l_{20}x_{00}) \\ x_{30} = -1/l_{33}(l_{32}x_{20}+l_{31}x_{10}+l_{30}l_{00}) \end{cases}$$

$$\begin{cases} x_{11} = 1/l_{11} \\ x_{21} = -1/l_{22}(l_{21}x_{11}) \\ x_{31} = -1/l_{33}(l_{32}x_{21}+l_{31}x_{11}) \end{cases}$$

$$\begin{cases} x_{22} = 1/l_{22} \\ x_{32} = -1/l_{33}(l_{32}x_{22}) \end{cases}$$

$$x_{33} = 1/l_{33}$$

A fragmented, nonexecutable coding to compute the inverse is

```
for(j=0; j<n-1; j++)
  for(i=j+1; i<n; i++)
    {
      sum=0.0;
      for(k=j; k<i; k++)
        sum+=L[i][k]*x[k][j];
      x[i][j]=-1/L[i][j]*sum;
    }
```

A demonstration program is written in L(X)_INV.CPP. Even though we have declared array elements "double," we note a loss of accuracy when tested

$$\mathbf{L}_x \mathbf{L}_x^{-1} = \mathbf{I}$$

We have deliberately chosen $\mathbf{L}_x$ so that a round-off error can be observed in this program,

$$1/3 = 0.333333..... = 0.333330$$

$$1/6 = 0.166666..... = 0.166667$$

The roundoff error occurs in a multiplication of $l_{ik}l_{kj}$ as well as when one element is rounded off to begin with.

1.4.3 U_1^{-1}

An inversion of a unit upper triangle matrix will be computed from an identity relationship:

$$\mathbf{U}\,\mathbf{U}^{-1} = \mathbf{I}$$

$$\begin{bmatrix} 1 & u_{01} & u_{02} & u_{03} \\ & 1 & u_{12} & u_{13} \\ & & 1 & u_{23} \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & x_{01} & x_{02} & x_{03} \\ & 1 & x_{12} & x_{13} \\ & & 1 & x_{23} \\ & & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & x_{01}+u_{01} & x_{02}+u_{01}x_{12}+u_{02} & x_{03}+u_{01}x_{13}+u_{02}x_{23}+u_{03} \\ 0 & 1 & x_{12}+u_{12} & x_{13}+u_{12}x_{23}+u_{13} \\ 0 & 0 & 1 & x_{23}+u_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

Equating the corresponding elements on both sides,

$$x_{01}+u_{01}=0 \quad \begin{cases} x_{02}+u_{01}x_{12}+u_{02}=0 \\ x_{12}+u_{12}=0 \end{cases} \quad \begin{cases} x_{03}+u_{01}x_{13}+u_{02}x_{23}+u_{03}=0 \\ x_{13}+u_{12}x_{23}+u_{13}=0 \\ x_{23}+u_{23}=0 \end{cases}$$

The unknown elements x_{ij} are obtained from above identities.

$$\begin{cases} x_{01} = -u_{01} \\ x_{12} = -u_{12} \\ x_{23} = -u_{23} \end{cases}$$

$$\begin{cases} x_{02} = -u_{02} - u_{01}x_{12} \\ x_{13} = -u_{13} - u_{12}x_{23} \\ x_{03} = -u_{03} - u_{01}x_{13} - u_{02}x_{23} \end{cases}$$

An examination of indices of x_{ij} and u_{ij} reveals an algorithm for coding.

```
for(i=n-2; i>=0; i--)
   for(j=n-1; j>=i+1; j--)
    {
      u[i][j]= -u[i][j];
      for(k=i+1; k<j; k++)
        u[i][j]=u[i][j]- u[i][k]*u[k[j];
    }
```

A demonstration program is shown in U(X)_INV.CPP. Round-off error is not encountered in this demonstration for a fortuitous selection of elements, and no division is involved.

1.4.4 U_x^{-1}

Inversion of an upper triangle matrix whose diagonal elements are not unity will be computed by a procedures similar to $\mathbf{U}_1^{-1}$.

$$\begin{bmatrix} u_{00} & u_{01} & u_{02} & u_{03} \\ & u_{11} & u_{12} & u_{13} \\ & & u_{22} & u_{23} \\ & & & u_{33} \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ & x_{11} & x_{12} & x_{13} \\ & & x_{22} & x_{23} \\ & & & x_{33} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

The unknown elements x_{ij} are obtained from the following ten identities.

$$\begin{cases} x_{33} = 1/u_{33} \\ x_{22} = 1/u_{22} \\ x_{11} = 1/u_{11} \\ x_{00} = 1/u_{00} \end{cases}$$

$$\begin{cases} x_{23} = -1/u_{22}(u_{23}x_{33}) \\ x_{12} = -1/u_{11}(u_{12}x_{22}) \\ x_{01} = -1/u_{00}(u_{01}x_{11}) \end{cases}$$

$$\begin{cases} x_{13} = -1/u_{11}(u_{13}x_{33}+u_{23}x_{23}) \\ x_{02} = -1/u_{00}(u_{03}x_{22}+u_{01}x_{12}) \end{cases}$$

$$x_{03} = -1/u_{00}(u_{03}x_{33}+u_{03}x_{23}+u_{01}x_{13})$$

A careful examination of indices of x_{ij} and u_{jk} reveals that the following algorithm would compute the inverse of $\mathbf{U}_x$.

```
for(i=n-2; i>=0; i--)
  for(j=n-1; j>i; j--)
    {
      for(k=j; k>i; k--)
        x[i][j]=x[i][j]+u[i][k]*x[k][i];
     x[i][j]=-x[i][j]/u[i][j];
    }
```

A demonstration program is written in U(X)_INV.CPP. The round-off error problem would not be observed since the reciprocals of diagonal elements are exact.

$$\mathbf{U}_x = \begin{bmatrix} 1 & 2 & 3 & 4 \\ & 5 & 6 & 7 \\ & & 8 & 9 \\ & & & 10 \end{bmatrix}$$

$$\mathbf{U}_x^{-1} = \begin{bmatrix} 1.000 & -0.4000 & -0.0750 & -0.0525 \\ & 0.2000 & -1.5000 & -0.0050 \\ & & 0.1250 & -0.1125 \\ & & & 0.1000 \end{bmatrix}$$

1.4.5 D^{-1}

Inversion of a diagonal matrix **D** is obtained by the reciprocal of the diagonal elements.

$$\mathbf{D} = \begin{bmatrix} d_{00} & & & \\ & d_{11} & & \\ & & d_{22} & \\ & & & d_{33} \end{bmatrix} \quad \mathbf{D}^{-1} = \begin{bmatrix} 1/d_{00} & & & \\ & 1/d_{11} & & \\ & & 1/d_{22} & \\ & & & 1/d_3 \end{bmatrix}$$

1.4.6 Q^{-1}

An inversion of orthogonal matrix **Q** is obtained by transposition as we have discussed in **QR** factorization.

$$Q^{-1} = Q^T$$

1.5 Vector Operations

So far we have associated a vector as an adjunct to a set of linear simultaneous equations as a column vector **B** or **X**.

$$\mathbf{AX} = \mathbf{B}$$

A vector could be a column vector or row vector, and the vectors have their own algebraic operations: Two vectors can be added, subtracted one from the other, multiplied by one another, for example.

A header file is compiled VECTOR.H to place most basic operations in a single file so that when we need vector operations we just link to this file. In the file we show how to construct a vector without size specified (empty vector) or a vector of specified size "n," but all the elements are zero (zero vector). We shall show how to construct a vector of size "n" with all the element entries specified in array form or from keyboard inputs.

Furthermore, the header file includes various unary operations such as shown below, where **V** and **W** are two vectors and "c" is a constant.

$$\mathbf{V} + \mathbf{W}$$

$$\mathbf{V} - \mathbf{W}$$

$$\mathbf{V} + c$$

$$\mathbf{V} - c$$

$$-\mathbf{V} \qquad \text{(negation)}$$

$$\mathbf{V}^*\mathbf{W}^1$$

1. **V** must be a row vector and **W** must be a column vector, and the result is a scalar. Other variants of multiplications will be handled in MATRIX.

V*c

V/c

We have added equality and inequality statements.

V == **W**

V != **W**

We have also included a check on the size of a vector. We have added a convenient step to extract a designated element out of a vector by array index.

The reader will have noticed the phrase we have added or included often. The header file could have included every conceivable vector operation. One person's complete file is another person's incomplete file and vice versa. We have not included in the file the overloaded binary operations such as (+=), (–=), (*=) or (/=). We shall include these in MATRIX.H.

The VECTOR.H is a precursor of MATRIX.H, an abbreviated file. Two exercise programs, VCTR_EX1.CPP and VCTR_EX2.CPP, are written to familiarize with the header file. Read the header file closely. The construction of the header file is an exciting and rewarding experience in C++ programming.

1.6 Matrix Operations

Matrix operations are a natural extension of vector operations, for a vector is a subset of a matrix. We have constructed a header file MATRIX.H, to compile most basic matrix operations.

We begin with construction of an empty matrix, followed by element entries by an array. The elements may be integer, float, or double; internally the integers and floats are converted to double. This may be overkill; however, prudence is necessary for successful operations. (Readers may eliminate these conversions to double if they wish.)

We have included a check on the number of rows and columns of the index. We have added a step to extract an element out of a matrix by index designation of an array A[i][j], or a matrix M(i, j). The first index is for row and the second for column as is the customary rule in the textbook on matrix theory.

Thereafter, eight unary operations followed; **M** and **P** are two matrices and "f" a constant.

operator=(const Matrix &M); **M=P** assign

operator+(const Matrix &M); **M+P**

operator+(const Matrix &f); **M**+f

operator–(const Matrix &M); **M**–f

operator–( ); **–M** negation

operator*(const double f); **M***f

operator*(const Matrix &M); **M*P**

The last operator executes the following five multiplications.

[(1-by-n) row vector] * [(n-by-1)column vector] = scalar

[(n-by-1) column vector] * [(1-by-n) row vector] = matrix(n, n)

[(m-by-n) matrix] * [(n-by-m) matrix] = matrix (m, m)

[(1-by-n) row vector] * [(n-by-k) matrix] = row vector (k)

[(m-by-k) matrix] * [(k-by-1) column vector] = column vector (m)

Last we have added three nonmember functions to the header file. They are listed as follows:

- Transpose: **M** to $\mathbf{M}^T$;
- Trace: Sum of diagonal elements;
- Identity: Unity diagonal matrix.

We do not claim the header file has compiled all conceivable matrix operations. Readers may add any useful member or nonmember functions to the file.

Five exercise programs are given in order to familiarize with matrix operations MTRX_EX1.CPP through MTRX_EX5.CPP using the header file. I hope that the exercises will elevate readers' confidence in handing matrix operations.

1.7 Conclusion

We have solved a set of linear simultaneous equations by Gaussian elimination with back-substitution and forward substitution.

The vector-matrix equation of the form,

$$\mathbf{AX} = \mathbf{B}$$

would have a unique solution when it is consistent. The consistency test is by the determinant of matrix **A**. The unique solution exists only when the determinant is not a null; **A** must be nonsingular. The determinant is easily computed by factorization of **A** into triangular matrices and a diagonal matrix. Inversion of the factored matrices is studied next. An inversion of a matrix is computed by the inverse of each constituent matrix multiplied in reversed order. We found that the inversion of a triangle matrix is much easier than the inversion of a square matrix.

List of Programs

Program	*Features*
(1) GAUSSBCK.CPP	Gaussian elimination with back-substitution
(2) GAUSSFWD.CPP	Gaussian elimination with forward-substitution
(3) LU_FCTR.CPP	**LU** factorization
(4) LLT_FCTR.CPP	$\mathbf{LL}^T$ factorization
(5) LDL_FCTR.CPP	$\mathbf{LDL}^T$ factorization
(6) UDU_FCTR.CPP	$\mathbf{UDU}^T$ factorization
(7) QR_FCTR.CPP	**QR** factorization
(8) QR_MMSE.CPP	Application of **QR** factorization
(9) L(1)_INV.CPP	Inverse of unit lower triangle matrix, $\mathbf{L}_1^{-1}$
(10) L(X)_INV.CPP	Inverse of a general lower triangle matrix, $\mathbf{L}_x^{-1}$
(11) U(1)_INV.CPP	Inverse of unit upper triangle matrix, $\mathbf{U}_1^{-1}$
(12) U(X)_INV.CPP	Inverse of a general upper triangle matrix, $\mathbf{U}_x^{-1}$
(13) VCTR_EX1.CPP	First exercise program on vector operation
(14) VCTR_EX2.CPP	Second exercise program on vector operation
(15) MATX_EX1.CPP	First exercise program on matrix operation
(16) MATX_EX2.CPP	Second exercise on matrix operation
(17) MATX_EX3.CPP	Third exercise on matrix operation
(18) MATX_EX4.CPP	Fourth exercise on matrix operation
(19) MATX_EX5.CPP	Fifth exercise on matrix operation
(20) VECTOR.H	Header file for vector operations
(21) MATRIX.H	Header file for matrix operations

We would not attempt to invert a square, nonsingular matrix by the direct method unless the dimension of the **A** matrix were two-by-two. We always factor it, take the inverse of the decomposed matrices, and multiply them in proper order. The unknown column vector X is given by without Gaussian elimination.

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$$

There are many excellent textbooks on linear algebra, matrix theory, and computation algorithms. The author is obliged to cite a few.

Selected Bibliography

Arthens, M., *Gradient, Matrix and Matrix Calculations,* Lexington, MA: Lincoln Lab., MIT, 1965 (AD 624426).

Golub, G. H., and C. F., Van Loan, *Matrix Computations*, Third ed., Baltimore, Maryland: The John Hopkin Univ. Press, 1996.

Kwak, J. H. and S. Hong, *Linear Algebra,* Boston, MA: Birkhauser, 1997.

Press, W. H., et al., *Numerical Recipes in C*, Second ed., Cambridge, MA: Cambridge Univ. Press, 1992.

Searle, A. R., *Matrix Algebra Useful for Statistics,* New York, John Wiley & Sons, 1982.

Strang, G., *Linear Algebra and Its Applications,* New York, Academic Press, 1980.

CHAPTER 2

Pseudorandom Number, Noise, and Clutter Generation

2.1 Introduction

In this chapter we learn how to generate random number sequences, noise, and clutter. All signals are invariably contaminated by noise and/or clutter. The contaminated signal is filtered or estimated to extract the signal by digital signal processing. Filtering will be covered in Chapter 3, the time-domain filtering by finite impulse response (FIR) or the infinite impulse response (IIR) filter. Filtering in the frequency domain will be covered in Chapter 4: fast Fourier transform (FFT) and inverse fast Fourier transform (IFFT). Estimation of signals will be covered in Chapter 8: Kalman filter.

There is no better way to understand signal processing required than by knowing the characteristics of noise or clutter—and no better way to understand noise and clutter than by generating them ourselves so that we have a thorough understanding of the noise and clutter we will encounter in the real world.

The kind of noise and clutter we have heard of, such as Gaussian noise, Rayleigh noise, exponential noise, chi-squared noise, lognormal clutter, and Weibull clutter, can be generated from unit uniform random variables. Unit uniform random variables are, in turn, generated from random number sequences.

True noise, a stochastic process, can never be manufactured (or generated) by a deterministic machine such as computer. In other words, true random number sequences can never be generated by computer programs, since the true random numbers do not repeat, do not have cyclic periods, do not have an end. However, we can generate a random number sequence as close to the a random sequence as we wish with the following conditions. The conditions are described as follows.

1. The pseudorandom number (PRN) sequence has a finite length N.
2. The PRN x_i, i = 1, 2, 3, . . . N, occurs only once in the total population N.
3. The PRN sequence must be contiguous in the range; that is, there shall be no missing nor duplicated number in N.
4. The sequence must not have periodicity in the range.

2.2 Pseudorandom Number and Unit Uniform Variables

A random number sequence that a computer program generates that meets the conditions above is called a PRN sequence. Some commercially available programs

claim to generate random numbers but fail to meet the conditions, especially condition (3), the contiguity.

Among many techniques of generating PRNs, we choose the mixed congruential method. The mixed congruential method is compact, portable, and easy-to-understand. The mixed congruential method is described by [1].

$$x_{i+1} = a \cdot x_i + c \quad \text{(modulo m)} \tag{2.1}$$

where

- a: Multiplier constant;
- x_i: Initial seed;
- c: Increment constant;
- m: Modulus constant, the total population of the sequence N, must be a prime number.

Equation (2.1) is programmed in PRN37A.CPP. The 37 is a prime number and the total population of the pseudorandom sequence. The PRNs generated are:

```
 9  12  27  28  33  21  35  31  11  22
 3  19  25  18  20  30   6  34  26  23
 8   7   2  14  37   4  24  13  32  15
10  17  19   5  29   1   9
```

The rules for specifying two integer constants "a" and "c" are mentioned in the footnote of the program. The rules are not very strict since any rule placed upon the randomness should not be binding. The rules should be taken as a guideline.

To check the correctness of the random number sequences, PRN_MISS.CPP is written. In the program the random numbers are rank-ordered in ascending order of magnitude for a quick check on the condition (3).

```
 1   2   3   4   5   6   7   8   9   9
10  11  12  13  14  15  16  17  18  19
20  21  22  23  24  25  26  27  28  29
30  31  32  33  34  35  37
```

The duplicated number is 9, and the missing number is 36. The PRN is corrected by replacing one of the duplicated numbers with the missing number. The corrected PRN sequence is now contiguous, distinctive, and uniformly distributed in the range [1, 37], with no missing or duplicated numbers.

```
 9  12  27  28  33  21  35  21  11  22
 3  19  25  18  20  30   6  34  26  23
 8   7   2  14  37   3  24  13  32  15
10  17  15   5  29   1  36
```

(2.2)

The mixed congruential method is versatile as well; another independent PRN sequence can be generated by changing the constant "a" and/or "c" and the initial seed x_i. A large number of independent PRN sequences would be obtained with the same population. As a demonstration we have written PRN37B.CPP by changing the increment constant "c."

7	37	2	12	25	16	8	5	27	26
21	33	19	23	6	32	14	35	29	36
34	24	11	20	28	31	9	10	15	3
17	13	30	4	22	1	18			

(2.3)

The unit uniform variables are obtained by simply scaling (2.2) or (2.3).

$$u_i = PRN_i / N$$

The probability density function of a unit uniform random variable is shown in Figure 2.1; no missing data or duplication of data is emphasized. Unit uniform random variables are the basic building components of white Gaussain noise. White Gaussian is, in turn, the building component of all other noise and clutter together with exponential noise, which we shall discuss shortly.

2.2.1 PRN Generation of an Arbitrary Population

We have employed the mixed congruential method in generating PRNs. The primary reason for selecting this method is that it is a simple algorithm, portable and versatile. A shortcoming may be that it requires a modulo operation on a prime number. The prime number is the total population of the PRN sequence. Suppose we wish

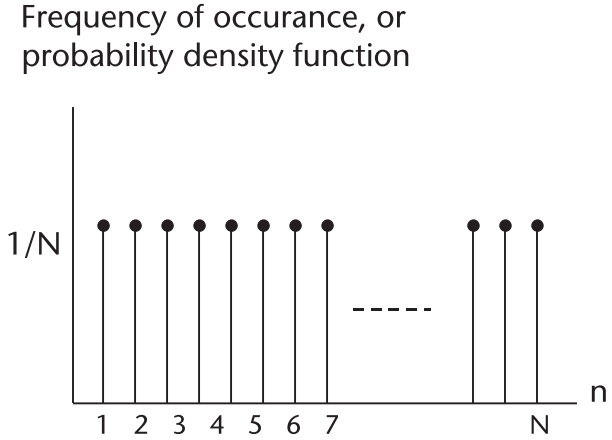


Figure 2.1 Probability density function, unit uniform random variables.

to generate random numbers different from the prime number, yet we demand a contiguity: no missing, no duplication. We follow the steps described below.

1. Select a prime number m immediately larger than the desired n.

Desired Population "n"	*Prime Number "m"*
32	37
64	67
100	101
128	131
256	257
.	.
.	.
512	521
1024	1031
.	.
.	.

2. Generate a PRN sequence by the mixed congruential method with modulo "m" and the proper choice of "a," "c" and the initial seed.

$$x_{i+1} = a \cdot x_i + c \qquad \text{(offset by one)}$$
$$x_{i+1} = (a \cdot x_i + c) \quad 1 \qquad \text{(offset by zero)}$$

3. Search and find the missing numbers and pair of duplicated numbers, and correct PRN sequence.
4. Discard the highest number(s) and form the "n" random sequence that is contiguous in the range.

Two demonstration programs are written in PRN64A.CPP and PRN64B.CPP, and a pair PRN 128A.CPP and PRN128B.CPP. We need a pair of unit uniform variables to generate Gaussian noise: 64A and 64B, 128A, and 128B. Four DAT files, PRN64A.DAT and PRN64B.DAT, PRN128A.DAT, and PRN128B.DAT, have stored the PRN sequences and the corresponding unit uniform random variables for later use.

2.3 White Gaussian Noise

White Gaussian noise is generated from a pair of independent unit uniform random variables $u_1(i)$ and $u_2(i)$.

$$G_{in}(i) = \text{sqrt}\{\ 2.0 \ln[u_1(i)] \cdot \cos[2\pi\, u_2(i)]\}$$
$$G_{qd}(i) = \text{sqrt}\{\ 2.0 \ln[u_1(i)] \cdot \sin[2\pi\, u_2(i)]\} \qquad (2.4)$$

$G_{in}(i)$ and $G_{qd}(i)$ are the in-phase and quadrature phase Gaussian noise components respectively. The "white" implies that the noise is uniform across the frequency

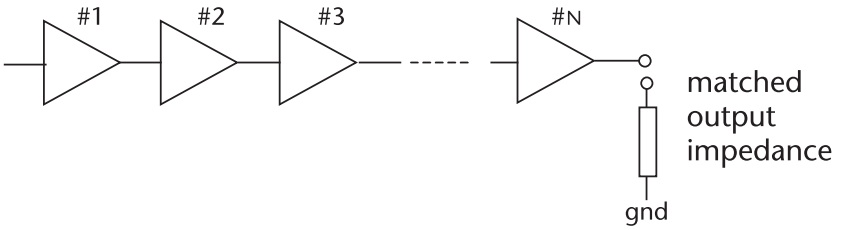


Figure 2.2 Cascaded amplifiers.

spectrum. "Gaussian" or sometimes "normal" refers to the amplitude distribution, the probability density function of the random variables, or the frequency of occurrence in the noise stream.

Consider the amplifier chain in Figure 2.2. The last amplifier is terminated with a matched impedance.

If we assume that the amplifiers are ideal and there is no band-limiting, the output is white Gaussian noise. When we intersperse bandpass filters between the amplifiers, the output noise is no longer white. We call the output a narrowband Gaussian noise. Consider a typical receiver chain as shown Figure 2.3. We call them the outputs I-channel and Q-channel baseband Gaussian noise, without the adjective "white."

Equation (2.4) is programmed in WGN64.CPP where $u_1(i) = (0.0, 1.0]$ and $u_2(i) = (0.0, 1.0]$. Both of them have a semiclosed range instead of a double-closed range of [0.0, 1.0]. The semiclosed range is a temporary necessity for the natural logarithm involved (natural logarithm of zero is undefined).

Since the noise power of the Gaussian distribution is given by,

$$\text{noise power} = \text{mean-squared} + \text{variance}$$

an error in nonzero mean and an ill-conditioned variance cause an intractable confusion in analysis of a result of signal processing. WGN64A.DAT are tested so that the mean and variance are exactly 0.0 and 1.0 respectively. Gaussian random variables with a nonzero mean and a specified variance other than unity are generated by,

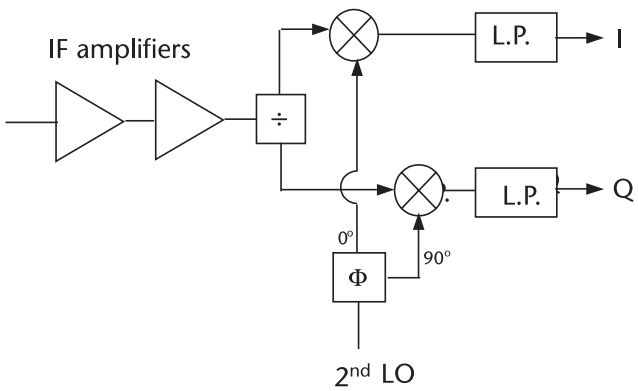


Figure 2.3 Typical receiver block diagram.

$$G(\text{mean, var}) = \text{mean} + \sigma \cdot G(0.0, 1.0)$$

The in-phase Gaussian noise component is shown in Figure 2.4. The probability density function in the bargraph is attached to the right. We expect the bar graph to approach to a normal when the number of samples is increased.

2.4 Rayleigh Noise

When narrowband Gaussian noise passes through a linear envelope detector, the output noise is said to be Rayleigh-distributed. A practical realization of an envelope detector is shown in Figure 2.5. The difference between a linear envelope detector and a square-law detector will be discussed later.

Rayleigh random variables are generated by,

$$R(i) = \text{sqrt}[G_{in}(i) \cdot G_{in}(i) + G_{qd}(i) \cdot G_{qd}(i)] \tag{2.5}$$

where $g_{in}(i)$ and $g_{qd}(i)$ are the in-phase and quadrature-phase components of Gaussian noise that we have generated in the previous section. Equation (2.5) is programmed in RAY64.CPP and shown in Figure 2.6.

The Rayleigh distribution has some interesting statistical characteristics:

$$f_r(x) = \frac{x}{\sigma^2} \exp\left\{\frac{-x^2}{2\sigma^2}\right\}$$

- Mean: $E\{x\} = \sigma \sqrt{\pi}\text{s.sqrt}(p/2)$;
- Second moment: $E\{x^2\} = 2\sigma^2$;

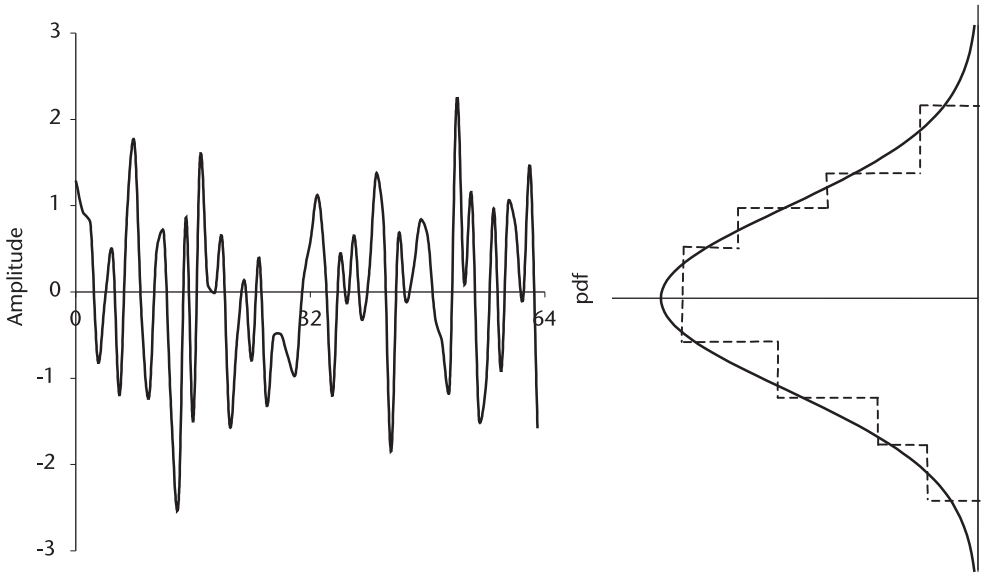


Figure 2.4 Gaussian noise, G(mean=0.0, var=1.0).

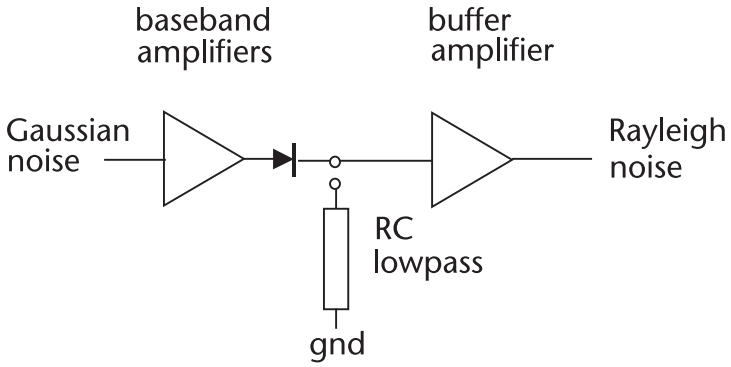


Figure 2.5 Gaussian noise detected by envelope detector.

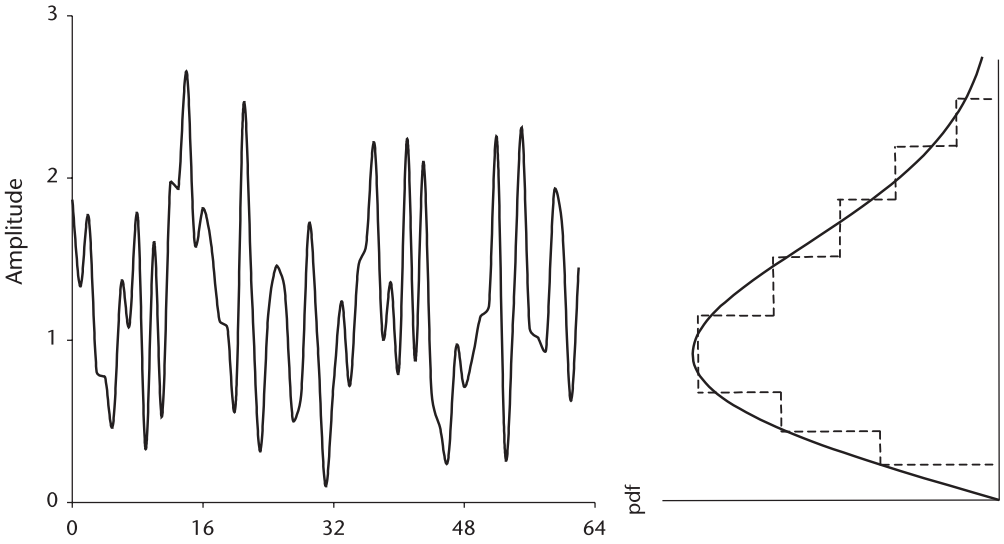


Figure 2.6 Rayleigh noise.

- Variance: $E\{x^2\} - [E\{x\}]^2 = \sigma^2(2-\pi/2)$;
- Mode: $x_{mod} = \sigma$;
- Median: $x_{med} = [-2\sigma^2 \ln(1/2)]$.

2.5 Rician Random Variables, Signal-to-Noise Ratio

When a sinusoidal signal plus narrowband Gaussian noise is envelope-detected, the output is said to be Rician-distributed. Two examples of the Rician distribution are a poor AM receiver with a high noise figure and a nonfluctuating target return added to the receiver thermal noise when demodulated by an envelope detector.

The Rician probability density function is given by

$$f_r(x) = \frac{x}{\sigma^2} \exp\left\{\frac{-1}{2\sigma^2}\ (x^2+A^2)\right\} I_o\left(\frac{xA}{\sigma^2}\right)$$

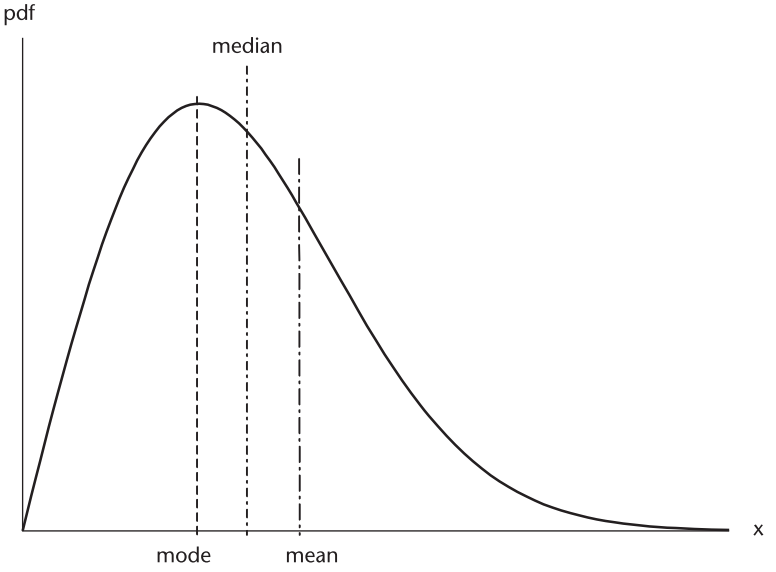


Figure 2.7 Rayleigh probability density function.

where

x: Rician ransom variable;
A: Amplitude of a sinusoidal signal;
I_o: Modified Bessel function of first kind, zero-order.

When the signal amplitude A is relatively high, the Rician probability function approaches a Gaussian with mean approximately equal to A. On the other hand, when the signal is absent the Rician pdf is identical to Rayleigh as it should be. The Rician pdf with various signal-to-noise ratios is shown in Figure 2.8. The Rayleigh pdf is added as a member of Rician.

Rician variates are generated by,

$$R(i) = \text{sqrt}\ \{[G_{in}(i) + A\cdot\cos\theta(i)]^2 + [G_{qd}(i) + A\cdot\sin\theta(i)]^2\} \tag{2.6}$$

where $G_{in}(i)$ and $G_{qd}(i)$ are the in-phase and quadrature-phase components of Gaussian noise with zero mean and unity variance. The angle $\theta(i)$ is assumed to be uniformly distributed in $[0, 2\pi]$. Equation (2.6) is programmed in RICE64.CPP and the results are shown in Figure 2.9.

Consider the case in which the circuit shown in Figure 2.5 is followed by a comparator with a threshold as shown below.

The comparator is assumed to be a baseband amplifier with a threshold rather than a binary "1" or "0" type. The comparator is to reduce the noise reaching an

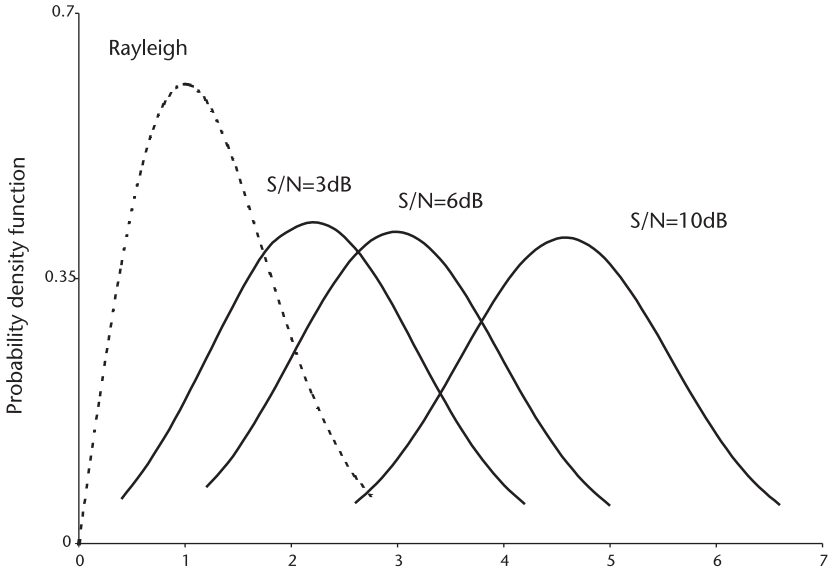


Figure 2.8 Rician pdf with various SNR values.

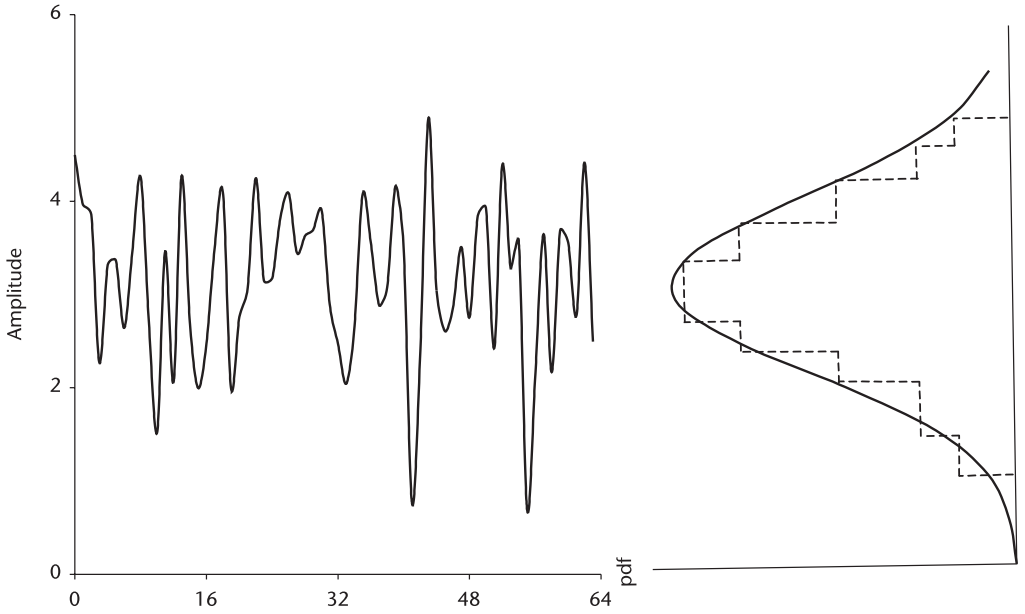
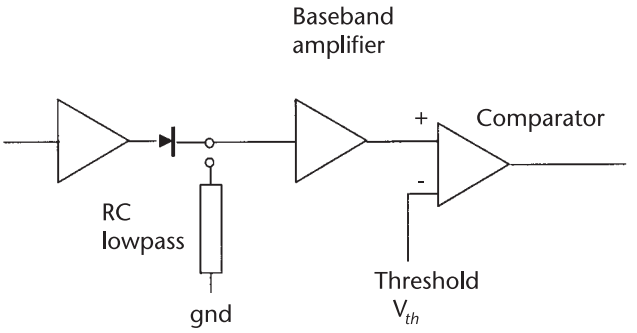


Figure 2.9 Rician distribution.

earphone, speaker, or some other signal presentation equipment. The operation of the comparator can be visualized as shown in Figure 2.10.



When the threshold is set as shown, the comparator will commit two kinds of errors:

The first error is that the noise is construed as a signal, and the second error is that the signal is construed as noise and discarded. The first error is called the probability of false alarm, and the second the detection loss. The determition of the threshold level requires a constrained optimization to reduce both errors.

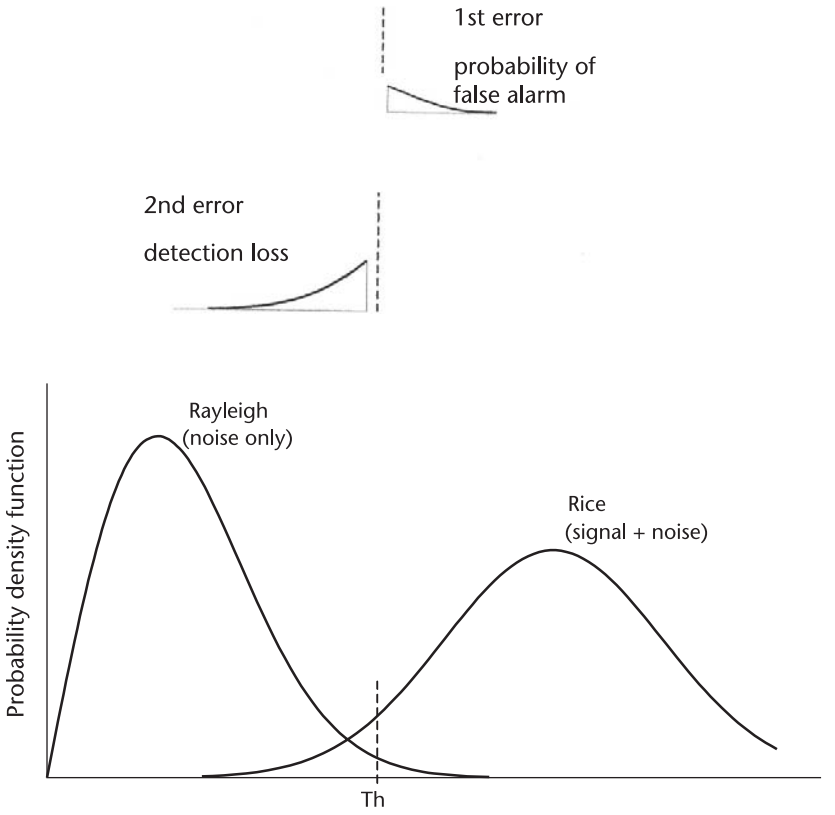


Figure 2.10 Comparator with threshold Th.

An example is in order. Suppose we specify the false alarm probability of 1.0E-6, one error in a million. We would set the threshold level such that the first error is 1.0×10^{-6}.

$$\text{first error} = \int_{\text{th}}^{\infty} \frac{x}{\sigma^2} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx \tag{2.7a}$$

$$= 1.0 - \int_{0}^{\text{th}} \frac{x}{\sigma^2} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx \tag{2.7b}$$

With the threshold level on hand, we compute the detection loss, the second error.

$$\text{second error} = \int_{0}^{\text{th}} \frac{x}{\sigma^2} \exp\left\{\frac{-1}{2\sigma^2}(x^2 + A^2)\right\} I_0\left(\frac{xA}{\sigma^2}\right) dx \tag{2.8a}$$

$$= 1.0 - \int_{\text{th}}^{\infty} \frac{x}{\sigma^2} \exp\left\{\frac{-1}{2\sigma^2}(x^2 + A^2)\right\} I_0\left(\frac{xA}{\sigma^2}\right) dx \tag{2.8b}$$

If the detection loss is relatively low, let's say 5% or less, the threshold level is accepted. If the detection loss is considered excessive, then we have to move the Rician pdf to the right until the detection loss is agreeably small: that is to increase the signal-to-noise ratio if we insist the false alarm probability should remain the same.

The complement of (2.8) is called Marcum's Q function [2] and extensive tables and graphs have been published on the threshold levels.

$$\text{detection probability} = \int_{0}^{\infty} \frac{x}{\sigma^2} \exp\left\{\frac{-1}{2\sigma^2}(x^2 + A^2)\right\} I_0\left(\frac{xA}{\sigma^2}\right) dx \tag{2.9}$$

A detailed discussion and analysis can be found in Chapters 7 and 9.

2.6 Chi-Squared Noise

In order to improve the detection performance, signal processing frequently employs so-called post-detection integration. A general scheme of a post-integration is shown in Figure 2.11.

When the input to a diode detector is narrowband Gaussian noise, the output of the summer is said to be chi-square distributed. Chi-squared random variables are the sums of n Gaussian noise samples squared:

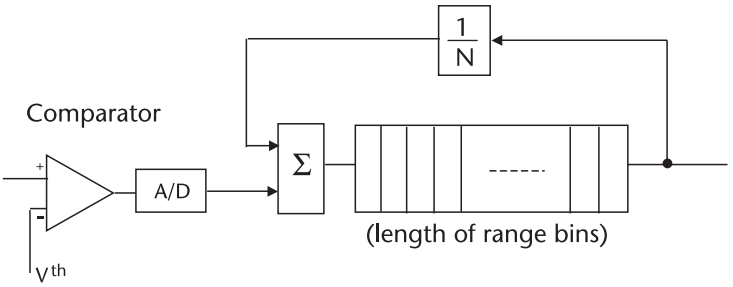


Figure 2.11 Post-detection integration, exponentially weighted.

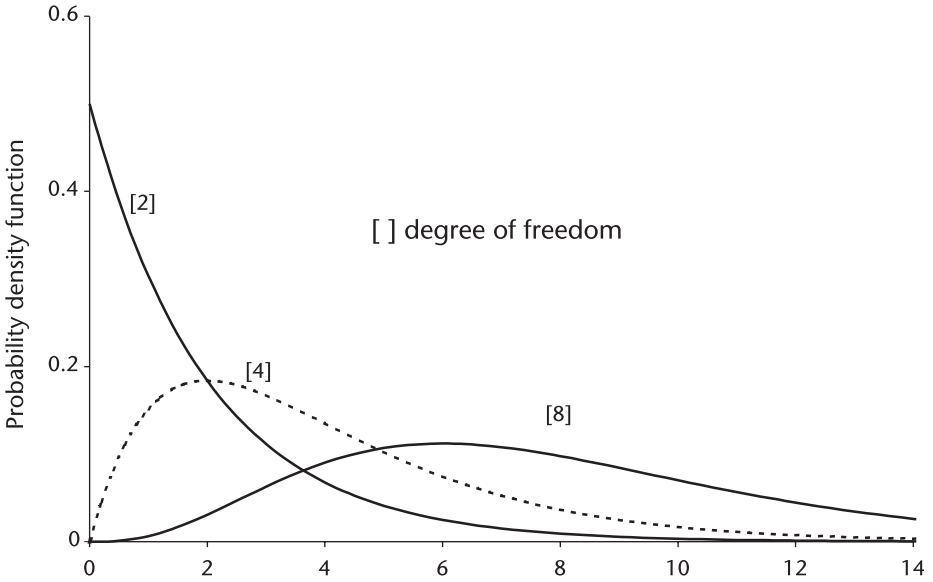


Figure 2.12 Probability density function, chi-squared, degree of freedom 2, 4, 8.

$$\chi^2(x) = G_1^2(x) + G_2^2(x) + G_3^2(x) + \ldots = \sum_{i=1}^{n} G_i^2(x)$$

The Gaussian noise vectors $G_i(x)$, i=1, 2, 3, . . . n, must have identical means and variances, and equal populations. The number of terms summed "n" is the degrees of freedom of the chi-squared random variable. The probability density functions of various degrees of freedom are shown in Figure 2.12. Degrees of freedom, 1 and 2 are exponential functions. The degree 1 has infinity value at x=0, the degree 2 is 0.5 at x=0. The degree 4 mimics a Rayleigh. As the degree of freedom increases further, the chi-squared approaches Gaussian.

2.7 Square-Law Detector

In this section we discuss the square-law detection of noise or signals, see Figure 2.13, where the current and voltage relationship of a diode is shown.

The i-v curve may be divided into two regions, the squared region and the linear region. The cutoff v_c for a silicon diode is 0.7V, 0.3V for a germanium, and 0.12V for a hot-carrier diode.

High-level noise will be detected in the linear region whereas low-level noise will be detected in the square region. When we contemplate inclusion of post-detection integration, Figure 2.11, an analysis for a linear envelope detector is quite difficult because in part the probability density function of sum of Rayleigh or Rician random variables is not known in a closed analytic form. The difference between a linear envelope detector and a square-law detector is less than 0.2 dB when the probability of false alarm is 1.0E-5 or less and the detection probability is 50% or higher. For this

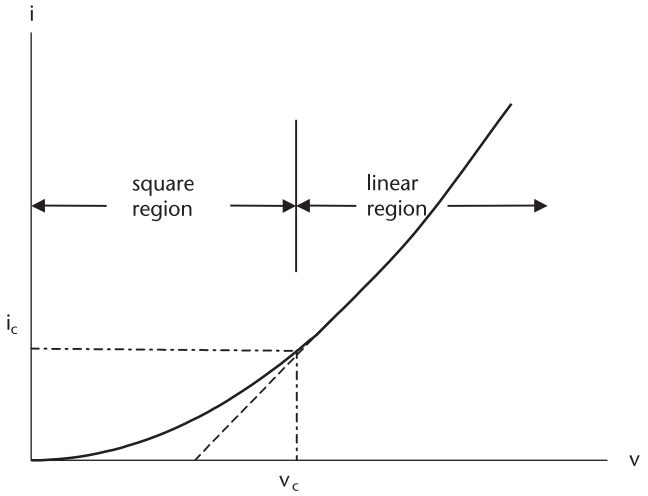


Figure 2.13 Current versus voltage characteristic of diode.

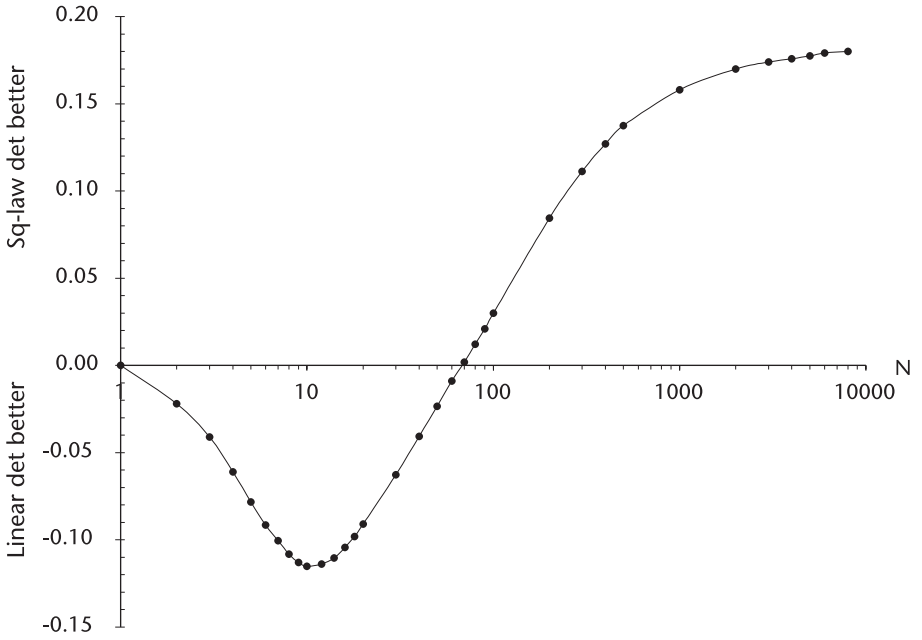


Figure 2.14 Comparison between linear and square-law Detector for N pulses integrated.

reason the analysis is exclusively executed on the assumption of a square-law detector (more about this in Chapter 7).

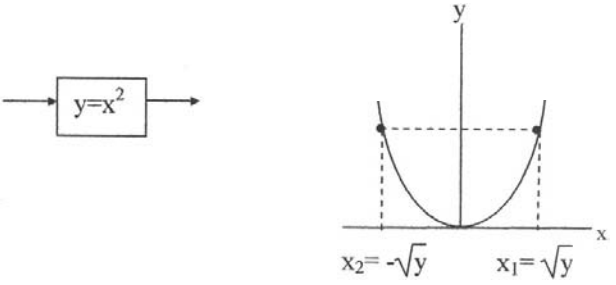
2.8 Exponential Noise

The probability density function out of a square-law detector when the input noise is Gaussian is given by [3],

$$f_Y(y) = \frac{f_x(x_1)}{|g'(x_1)|} + \frac{f_x(x_2)}{|g'(x_2)|} + \ldots + \frac{f_x(x_n)}{|g'(x_n)|} \tag{2.10}$$

where

$$g'(x) = \frac{dy}{dx} \quad \begin{cases} g'(x_1) = 2x_1 = 2\sqrt{y} \\ g'(x_2) = -2x_2 = -2\sqrt{y} \end{cases}$$



$$f_Y(y) = \frac{1}{2\sqrt{y}}\frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-y}{2\sigma^2}\right\} + \frac{1}{2\sqrt{y}}\frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-y}{2\sigma^2}\right\}$$

$$= \frac{1}{\sigma\sqrt{2\pi y}}\exp\left\{\frac{-y}{2\sigma^2}\right\} \tag{2.11}$$

The output noise is distributed as an exponential function, infinite at y=0. This is the chi-squared probability density function with one degree of freedom.

Narrowbanded Gaussian noise whose instantaneous amplitude is z can be considered as a vectorial sum of real and imaginary components. The instantaneous angle θ is assumed uniformly distributed in [0, 2π].

$$z = r\cos\theta + j(r\sin\theta) = x + j\,y$$

The probability distribution function $F_{xy}(x,y)$ is obtained by integrating the joint probability density function $f_{xy}(x,y)$.

$$F_z(z) = \int\int_{x^2+y^2 \le z} f_{xy}(x,y)dx\ dy$$

$$= \int\int \frac{1}{2\pi\sigma^2}\exp\left\{\frac{-(x^2+y^2)}{2\sigma^2}\right\}dx\ dy$$

$$= \int_0^{\sqrt{z}} \frac{2\pi r}{2\pi\sigma^2}\exp\left\{\frac{-r^2}{2\sigma^2}\right\}dr, \quad x = r\cos\ ,\quad y = r\sin$$

$$= 1.0 \quad \exp\{\frac{z}{2\sigma^2}\}$$

By differentiating the probability distribution function above we obtain the probability density function of z.

$$f_Z(z) = \frac{1}{2\sigma^2}\exp\{\frac{z}{2\sigma^2}\} \tag{2.12}$$

The output noise is distributed exponentially, $f_Z(z) = 0.5$ at $z = 0$, with variance σ^2=1. This is chi-squared with two degrees of freedom, see Figure 2.12. The mean (expectation) and variance of the exponential probability density function are

$$E\{x\} = \int_0^\infty z\ f_z(z)dz = \int_0^\infty \frac{-z}{2\sigma^2}\exp\left\{\frac{-z}{2\sigma^2}\right\}dz = 2\sigma^2$$

$$E\{x^2\} = \int_0^\infty z^2\ f_z(z)dz = \int_0^\infty \frac{-z^2}{2\sigma^2}\exp\left\{\frac{-z}{2\sigma^2}\right\}dz = 8\sigma^4$$

Therefore, the variance $\sigma^2 = E\{x^2\} \quad [E\{x\}]^2 = 4\sigma^4$, and the standard deviation and the median are given by

$$\text{standard deviation} = 2\sigma^2$$

$$\text{median} = \int_0^{zm} f_Z(z)\ dz = \frac{1}{2}, \quad z_m = 2\sigma^2\ [\ \ln(1/2)]$$

The generation of exponential random variables is by the inverse transform method [4].

$$u = F_Z(z) = 1 \quad \exp\left\{\frac{z}{2\sigma^2}\right\} \qquad \text{(u is the unit uniform random variable)}$$

$$1\text{–}u = \exp\left\{\frac{z}{2\sigma^2}\right\}, \quad \text{and } \ln(1\text{–}u) = \frac{z}{2\sigma^2}$$

Since (1–u) is identically distributed as u,

$$\ln u = \frac{z}{2\sigma^2}$$
$$z_i = 2\sigma^2 [\ \ln(u_i)] \tag{2.13}$$

Equation (2.13) is programmed in EXP128.CPP, and the result is shown in Figure 2.15.

2.9 Lognormal Clutter

The reflected power from the sea observed by a high-resolution radar at a low grazing angle is said to be distributed as logarithmic Gaussian, lognormal for short.

$$f_L(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-1}{2\sigma^2}(\ln x - m)^2\right\} \tag{2.14}$$

A harbor surveillance radar receives log-normal clutter when the sea state is high, a windy and choppy sea [5]. Lognormal probability density functions with a mean of 1, 2, and 4 with unity variance are shown in Figure 2.16. The mean and variance are referred to the parent Gaussian, not to the lognormal. The lognormal probability density function exhibits a longer tail than any other distribution. The longer tail implies a spiky amplitude of clutter in the time domain.

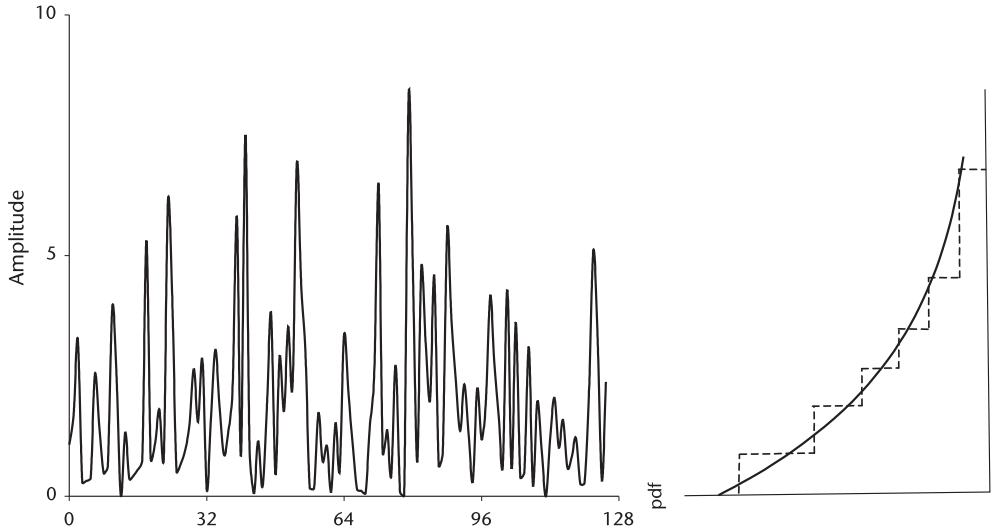


Figure 2.15 Exponential noise.

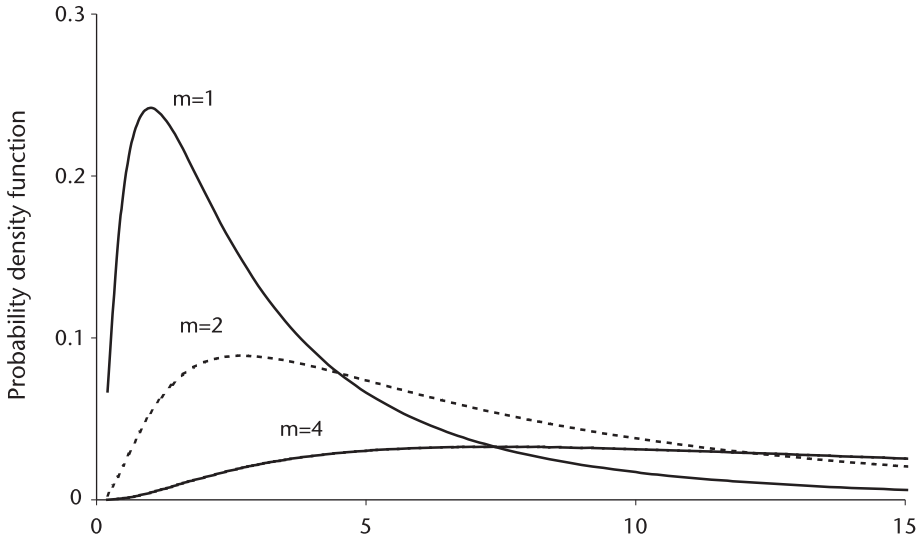


Figure 2.16 Lognormal probability density function.

Lognormal random variables are generated from Gaussian random variables:

1. Generate Gaussian random variables G_i(mean=0, var=1.0);
2. Let temporary variables y_i as,

$$y_i = \text{mean} + \sigma \cdot G_i(0.0, 1.0)$$

3. Lognormal clutters $x_i = \exp\{y_i\}$.

Lognormal clutter is generated in LOGNORM.CPP and shown in Figure 2.17.

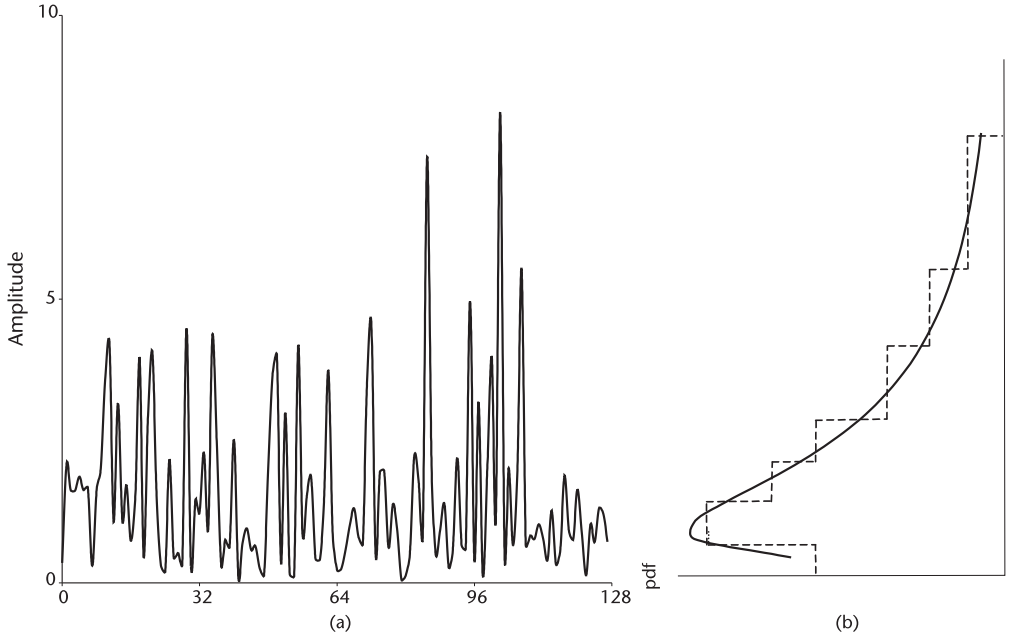


Figure 2.17 Lognormal clutter.

2.10 Weibull Clutter

The Weibull probability density function is given by,

$$f_W(x) = \left(\frac{c}{b}\right)\left(\frac{x}{b}\right)^{c-1} \exp\left\{-\left(\frac{x}{b}\right)^{c}\right\} \tag{2.15}$$

where

c: The shape parameter;
b: The scale parameter.

A Weibull pdf with various "c" parameters are shown in Figure 2.18. When "c" is unity and "b" arbitrary, the Weibull pdf is identical to the exponential pdf. When c=2 and $b=2\sigma^2$, the Weibull is identical to the Rayleigh while when c is greater than 2.5, the Weibull resembles Gaussian. The Weibull pdf is a flexible function.

The Weibull pdf (named for Swedish engineer Weibull) was originally applied to the analysis of strength of materials: fatigue and breakdown of mechanical components of a structure, or a life-test (longevity) of electrical devices such as incandescent bulbs.

Since the probability density function is so adaptable, the function is now applied to the analysis of radar clutter returns. A large volume of returned data collected and analyzed indicates that the land and sea returns are distributed as Weibull. A vegetated field and forest viewed by a low-resolution radar at a high grazing angle approaches Rayleigh. On the other hand, in a high-resolution radar at a low grazing angle on land and sea the return powers are distributed as Weibull of varying "c" [5].

Weibull random variables are generated by the inverse transform method. The probability distribution function $F_W(x)$ is obtained by integrating the probability density function.

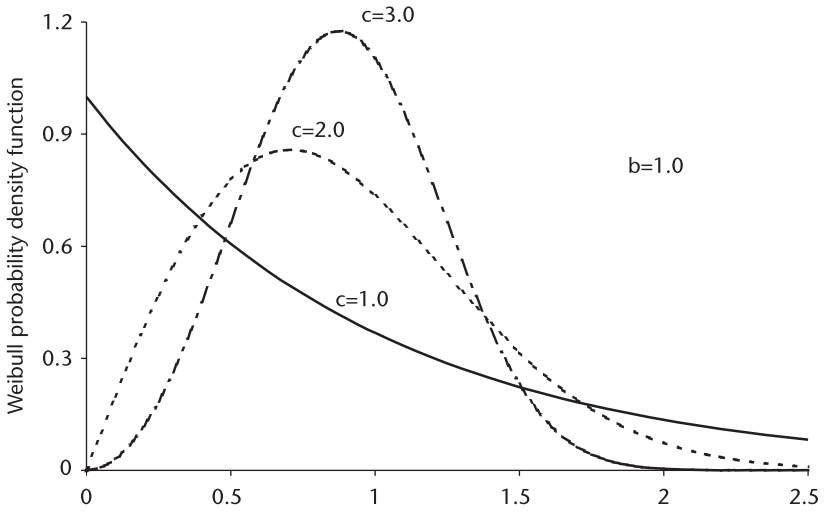


Fig 2.18 Weibull probability density function with various shape parameters.

$$F_W(x) = \int_0^x \left(\frac{c}{b}\right)\left(\frac{x}{b}\right)^{c-1} \exp\left\{-\left(\frac{x}{b}\right)^c\right\} dx$$
$$= 1 - \exp\left\{-\left(\frac{x}{b}\right)^c\right\}$$

The unit uniform random variable u(0.0, 1.0] is equated to the probability distribution function $F_W(x)$,

$$u = F_W(x) = 1 \quad \exp\left\{ \left(\frac{x}{b}\right)^c\right\}$$

so that

$$1-u = \exp\left\{ \left(\frac{x}{b}\right)^c\right\}, \quad \text{and} \quad \ln(1 \ \ u) = \left(\frac{x}{b}\right)^c$$

Since (1–u) is identically distributed as u, we set,

$$\ln(u) = \left(\frac{x}{b}\right)^c$$
$$\text{Finally,} \qquad x_i = b\cdot[\ \ \ln(u_i)]^{1/c}. \tag{2.16}$$

Equation (2.16) is programmed in WEIBULL.CPP and the result is shown in Figure 2.19. We have chosen c = 1.5 and b = 1.0, somewhere between exponential and Rayleigh in the program.

What would be the output distribution when Weibull clutters are detected by a square-law detector? We apply the transformation principle of the input-output relationship of the probability density function given by (2.10).

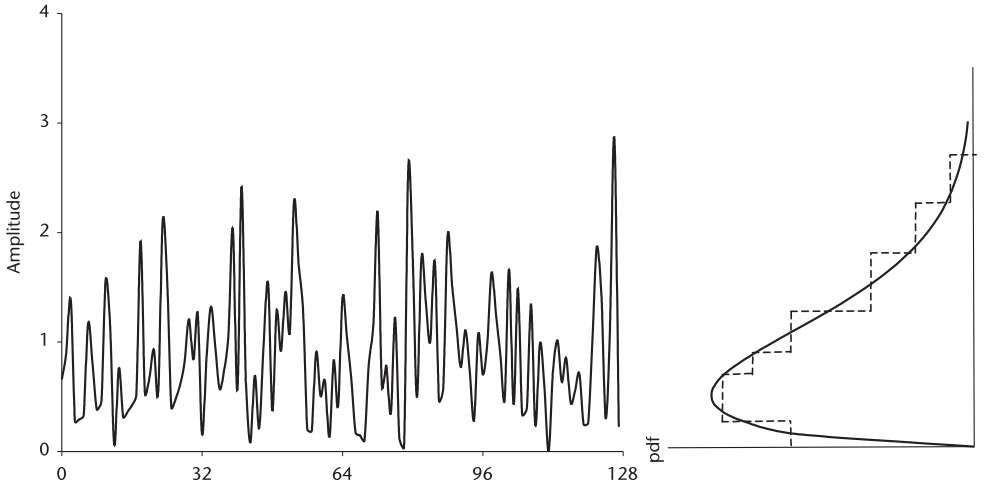
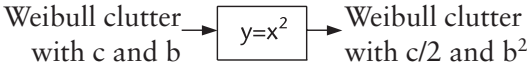


Figure 2.19 Weibull clutter and probability density function.

$$f_Y(y) = \frac{f_x(x_1)}{|g'(x_1)|} = \frac{1}{2\sqrt{y}}\left(\frac{c}{b}\right)\left(\frac{x}{b}\right)^{c-1}\exp\left\{-\left(\frac{x}{b}\right)^{c}\right\}\Big|_{x1=\sqrt{y}}$$

$$= \frac{c/2}{b^2}\left(\frac{y}{b^2}\right)^{c/2-1}\exp\left\{-\left(\frac{y}{b^2}\right)^{c/2}\right\}$$



We recognize that the output probability density function is another Weibull with altered c and b. This may be another reason we may prefer a square-law detector over a linear detector. The output shape parameter is one-half that of the input, and the output scale parameter is the square of the input. When the input scale parameter is less than unity, the corresponding output scale parameter decreases; when it is larger than unity it increases at the output.

The Weibull probability density function can be changed to an exponential, Rayleigh, or Gaussian by merely selecting a different value for the shape parameter. The parameters such as mean, variance, standard deviation, median, and mode are presented in Chapter 10.

2.11 Postulate of Probability Density Function from Sampled Data

We have generated various types of noise and clutter in the previous sections. In this section we study the inverse problem. Suppose we are given a set of sampled data of unknown distribution, and we wish to find the probability density function of the data. We present two demonstrations of how to find the pdf.

Demonstration 1

A group of electronic components are tested for their longevity under stress. The following data of their lifetime, in hours, is collected. We suspect the longevity may be distributed as Gaussian. In order to test the postulate we plot the data on normal probability paper. (We shall show how to construct the paper shortly). If a straight line is observed, our postulate is correct; if not, we postulate another distribution. The collected data follow (in hours):

77	21	18	51	65	48	83	37	43	48
29	55	13	77	30	23	29	55	72	28
49	12	73	56	33	78	47	50	59	19
44	50	41	39	63	66	19	94	34	24
34	85	57	30	45	59	52	40	56	33
30	59	50	45	55	86	56	56	53	82
80	62	52	97						

We construct the following table showing the rank-ordered subgroup, the observed frequency count, the cumulative frequency count, and the expected cumulative frequency as follows:

[i]	*rank-ordered subgroup*	*freq count*	*cumulative freq count*	*expected cumulative freq, 1/64 (column 5)*
1	10 – 14	2	2	0.031
2	15 – 19	3	5	0.078
3	20 – 24	2	7	0.109
4	25 – 29	3	10	0.156
5	30 – 34	8	18	0.281
6	35 – 39	2	20	0.313
7	40 – 44	4	24	0.375
8	45 – 49	6	30	0.460
9	50 – 54	7	37	0.578
10	55 – 59	11	48	0.750
11	60 – 64	3	51	0.797
12	65 – 69	2	53	0.828
13	70 – 74	2	55	0.858
14	75 – 79	3	58	0.906
15	80 – 84	2	60	0.938
16	85 – 89	2	62	0.969
17	90 – 94	1	63	0.984
18	95 – 99	1	64	1.000

Columns 2 and 5 are plotted on normal probability paper as shown in Figure 2.20 by taking the midpoint of column 2 as the abscissa, and column 5 as the ordinate. A straight line is drawn by visual adjustment. A straight line indicates that our initial postulate is correct: The sampled data are distributed as Gaussian. We note that the mean (0.5) is approximately 50 hours, and the standard deviation (0.159 and 0.841) is approximately 70–27=43. We can test the two parameters by the sample mean and variance; however, the plotting checks the correctness of our postulate.

$$\text{mean} = \frac{1}{N}\sum_{j=1}^{N} x_j$$

$$\text{variance} = \frac{1}{N-1}\sum_{j=1}^{N} (x_j - m)^2$$

When the sampled data are postulated as lognormally distributed, we still use normal probability paper after column 2 is converted to logarithm.

Demonstration 2

The proper function of electronic equipment depends critically on the weakest component. The weakest component on a circuit board may be a transistor, diode, resistor, inductor, or another constituent element. When the weakest component fails

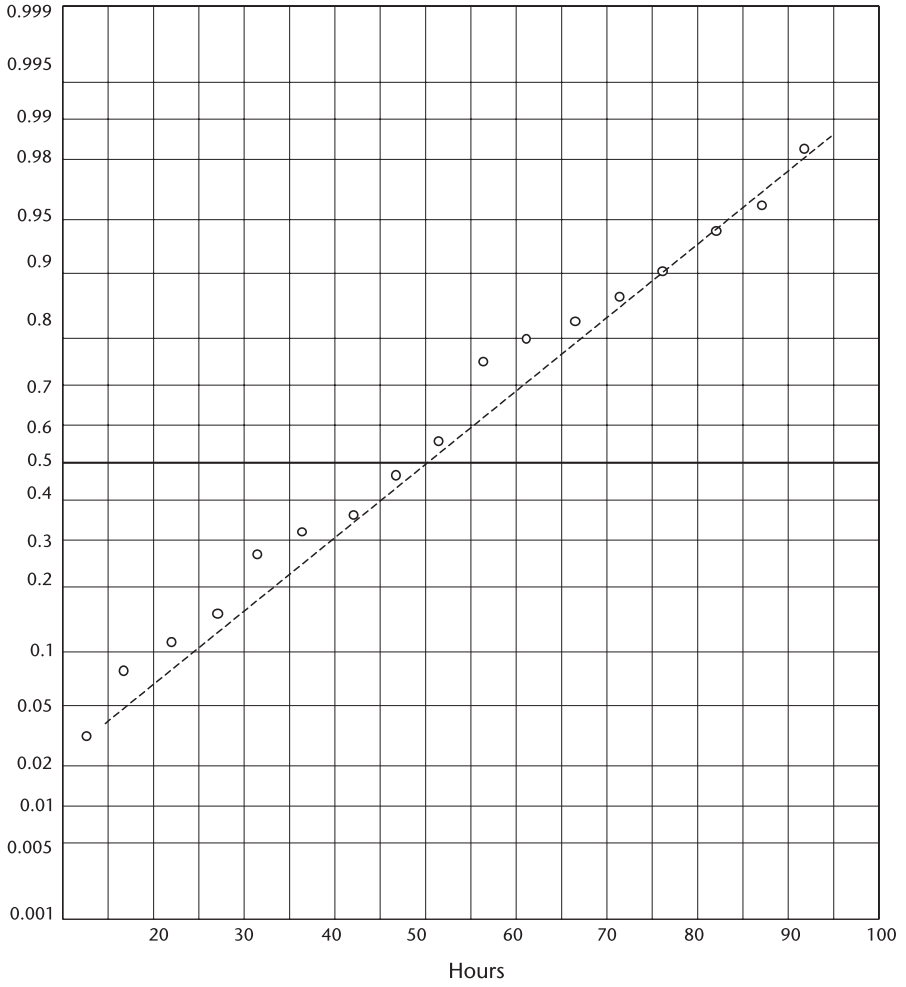


Figure 2.20 Sampled data postulated as gaussian.

the circuit board's function fails and so does the equipment. (No parallel redundancy is considered here.) The failure rate is postulated to be Weibull with unknown parameters. The Weibull probability density function and distribution function are repeated below for convenience.

$$f_W(x) = \left(\frac{c}{b}\right)\left(\frac{x}{b}\right)^{c\ 1} \exp\left\{\ \left(\frac{x}{b}\right)^{c}\right\} \tag{2.18}$$

$$F_W(x) = 1\quad \exp\{\ (xb)^{c}\} \tag{2.19}$$

Taking the natural logarithm of (2.19) twice, we have

$$\ln\left\{\ln\left[\frac{1}{1 - F_W(x)}\right]\right\} = c\ \ln(x) - c\ \ln(b) \tag{2.20}$$

If we equate the left-hand side to y_i, (2.20) leads us to a linear equation

$$y_i = c\ \ln(x_i) \quad c\ \ln(b)$$

The data plotted on a linear-linear graph should look as shown in Figure 2.21. The slope of the straight line would give us an approximation of the shape parameter c and the intercept point the scale parameter b. Our primary object in plotting is to test the postulate of the distribution of the sampled data.

A better estimate of two parameters can be obtained by the method of MMSE.

$$\text{minimize} \sum_{i=0}^{N} [y_i\ \ c\ \ln(x_i) + c\ \ln(b)]^2$$

Estimates of c and b are given by

$$\hat{c} = \frac{\sum y_i \ln(x_i) - E\{\ln x\}\ E\{y\}}{\sum (\ln x)^2 - [E\{\ln y\}\ ]^2} \tag{2.21}$$

$$\hat{b} = \exp\left\{ \frac{-E\{y\}}{\hat{c}} + E\{\ln x\} \right\} \tag{2.22}$$

where

$$E\{\ln x\} = \frac{1}{N} \sum \ln x_i;$$

$$E\{y\} = \frac{1}{N} \sum y_i;$$

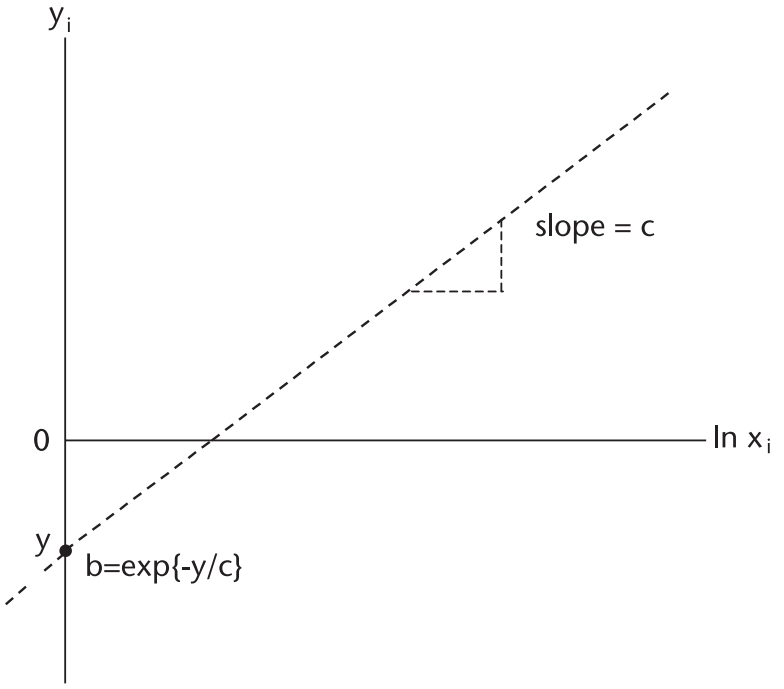


Figure 2.21 Linear Equation of (2.20). Sampled data postulated as Weibull.

x_i and y_i are a rank-ordered pair.

All that remains is to compute the left-hand side of (2.20) where $F_W(x)$ is involved, and we have equated it to y_i. An estimate of $F_W(x)$ is given by Bury [6] and Ross [7].

$$E\{F_W(x)\} = \frac{i}{N+1} \tag{2.23}$$

Therefore,

$$y_i = \ln\left\{\ln\left[\frac{1}{1 - F_W(x)}\right]\right\} = \ln\{-\ln[1 - F_W(x)]\}$$

$$= \ln\left\{-\ln\left[\frac{N + 1 - i}{N + 1}\right]\right\} \tag{2.24}$$

Numerical Demonstration 3

A group of batteries is tested under stress for longevity. Stress tests are often performed to shorten the testing time in laboratory. The following data (in months) is collected, and we postulate that the life-length is distributed as Weibull.

1.878	0.756	0.847	1.310	0.454
1.024	0.665	0.316	0.569	1.545
1.420	0.935	2.162	1.209	1.692
1.115				

After 16 failures the testing is terminated. We rank-order the longevities, (in months) in ascending order and construct the following table.

[i]	*rank-ordered* x_i	*ln* (x_i)	$y_i = ln\left\{-ln\left[\frac{N+1-i}{N+1}\right]\right\}$
1	0.316	-1.152	-2.803
2	0.454	-0.790	-2.078
3	0.569	-0.564	-1.639
4	0.665	-0.408	-1.316
5	0.758	-0.279	-1.055
6	0.847	-0.166	-0.832
7	0.935	-0.067	-0.634
8	1.024	0.024	-0.455
9	1.115	0.109	-0.283
10	1.209	0.190	-0.120
11	1.310	0.290	0.041
12	1.420	0.351	0.202
13	1.545	0.435	0.369
14	1.691	0.525	0.551
15	1.878	0.639	0.761
16	2.161	0.771	1.041

The data is plotted in Figure 2.22; $\ln(x_i)$ is the abscissa, y_i is the ordinate. A straight line is drawn visually. The straight line confirms our initial postulate of a Weibull distribution. The slope of the line, the shape parameter, is approximately 2. The intercept point indicates that the scale parameter is approximately 1.3. Once a straight line is observed the estimates of the parameters would be computed by MMSE via (2.20) and (2.21).

Plotting the sampled data on either Gaussian probability paper or linear-linear paper is a powerful tool to test a postulate on the distribution of the sampled data.

2.12 Construction of Gaussian (Normal) Probability Paper

Normal probability paper (Gaussian graph) is commercially available; however, we would like to construct one ourselves. See Figure 2.20 without the sample points. The Gaussian density function and distribution function are

$$f_G(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} \tag{2.25}$$

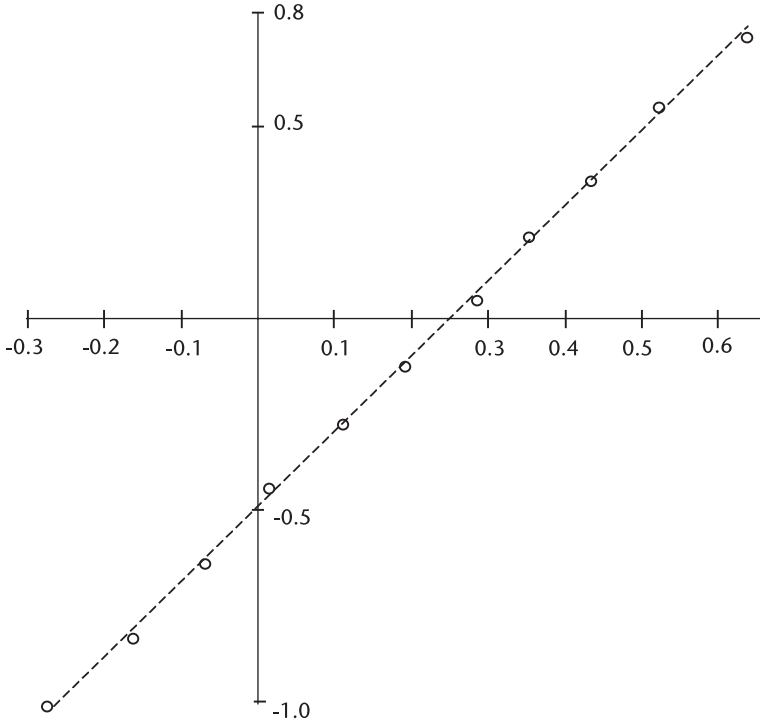


Figure 2.22 Failure rate test on batteries.

$$F_G(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx \tag{2.26}$$

When we specify, for example, $F_G(x) = 0.8$, we wish to know the corresponding value of x through (2.25); that is,

$$0.8 = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} dx$$

An iterative integration technique would give us the value of x (see Chapter 9); however, the Gaussian table and a linear interpolation may be a simpler way to obtain the values of x.

Using a Gaussian table and a straight-line interpolation we have written NORMGRAP.CPP. The result is shown below. The x's are the ordinate scale, labeled as $F_G(x)$. The abscissa is in linear scale.

$F_G(x)$	x	
0.99	2.3264	
0.98	2.0539	
0.95	1.6450	
0.90	1.2816	
0.80	0.8417	
0.70	0.5244	
0.60	0.2534	
0.60	0.0000	(symmetry)
0.40	-0.2534	
0.30	-0.5244	
0.20	-0.8417	
0.10	-1.2816	
0.05	-1.6450	
0.02	-2.0539	
0.01	-2.3264	

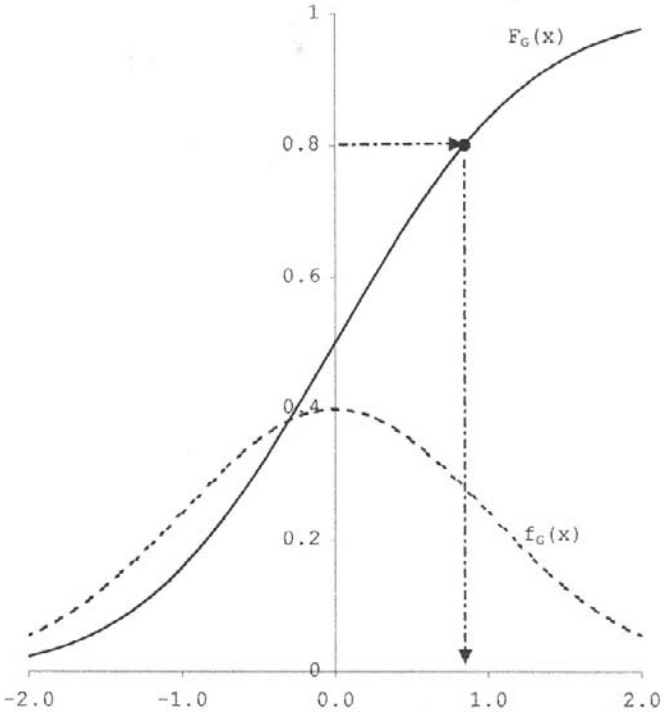
Exponential probability paper is easier to construct since the probability distribution function is integrable analytically.

$$f_E(x) = \frac{1}{\sigma^2} \exp\left\{\frac{x}{\sigma^2}\right\}$$

$$F_E(x) = 1 \quad \exp\{\frac{x}{\sigma^2}\}$$

$$x = \quad \ln [1 \quad F_E(x)] \qquad (\sigma^2=1)$$

The x is ordinate, labeled as $F_E(x)$, and the abscissa is in linear scale. We have not drawn the graph.



Weibull probability paper is a blank sheet with linear-linear scale as shown in Figure 2.21 and 2.22.

2.13 Conclusion

We have generated PRNs by the mixed congruential method with contiguity; no missing numbers, no duplicated numbers. The unit uniform random variables are generated from the PRNs.

Gaussian, Rayleigh, Rician, and chi-squared noise sequences are generated from the unit uniform random variables by the direct method. (Noise is engineering terminology; the random variable is probability/statistical terminology.)

We have learned how to derive the output probability density function of a square-law detector. We have learned different names for chi-squared noise. We have learned how to generate exponential noise, lognormal clutter, and Weibull clutter by the inverse transform method. Lognormal distributions have very spiky waveforms.

The Weibull probability density function is a very flexible, adaptable function with variable shape parameter "c." It can be an exponential, Rayleigh, Gaussian look-alike, or somewhere between.

Noise and clutter are generated from unit uniform random variables. We claim that all the noises and clutters we would encounter in communication and radar signal processing are generated in this chapter.

List of Programs

Program	*Features*
(1) PRN37A.CPP	Generates PRN sequence of population 37.
(2) PRN_MISS.CPP	Checks any missing or duplicated numbers in PRN37A.CPP, corrects the PRN sequence and generates a unit uniform random variables.
(3) PRN37B.CPP	Generates another PRN of length 37, corrects the sequence, generates second set of independent unit uniform random variables.
(4) PRN64A.CPP	Generates PRN sequence of length 64, two
(5) PRN64B.CPP	independent unit uniform random variables for later use.
(6) PRN64B.CPP	independent unit uniform random variables for later use.
(7) PRN128A.CPP	Generates a pair of PRN sequences, N=128,
(8) PRN128B.CPP	independent and identically distributed.
(9) WGN64.CPP	Generates white Gaussian noise,
(10) WGN128.CPP	N=64 and N=128.
(11) RAYLEIGH.CPP	Generates Rayleigh noise, N=64.
(12) RICE64.CPP	Generates Rician random variables of length 64 with a specified signal-to-noise ratio.
(13) GRAPHRIC.CPP	Generates a Rician probability density function with three different SNRs, 0 dB, 3 dB, 6 dB, and 10 dB.
(14) GRAPHCHI.CPP	Generates a χ^2 probability density function with degree of freedom 2, 4, and 8.
(15) EXP128.CPP	Generates exponential noise from unit uniform random variables by inverse transform.
(16) LOGNORM.CPP	Generates lognormal clutter from WGN128.DAT
(17) GRAPHLOG.CPP	Generates a lognormal clutter probability density function with mean 1, 2 and 3.
(18) WEIBULL.CPP	Generates Weibull clutter from PRN128. DAT with two parameters specified.
(19) GRAPHWBL.CPP	Generates three Weibull pdf's with the shape parameter c=1.0, c=2.0 and c=3.0, while the shape parameter b is held constant.
(20) NORMGRA.CPP	Computes data to construct Gaussian probability paper.
(21) STATIST.H	A header file attached to CPP driver whenever statistics of data such as mean, variance, standard deviation, median, or mode are needed to compute.

References

[1] Abramowitz, M., and I. Stegun, *Handbook of Mathematical Functions with Formula, Graphs and Tables*, National Bureau of Standards, 1965.

[2] Marcum, J. I., *A Statistical Theory of Target Detection by Pulsed Radar,* RM-754, The Rand Corporation, Santa Monica, California, Dec. 1947.

[3] Papoulis, A., *Probability, Random Variables and Stochastic Process*, New York, N.Y.: McGraw-Hill, 1965.

[4] Schleher, D. C., "Radar Detection in Lognormal Clutter," *IEEE Int'l Radar Conf*., 1975.

[5] Sekine, M., *Radar Signal Processing Technique (in Japanese)*, Society of Electronics Information and Communication, Tokyo, Japan, 1991.

[6] Bury, K. V., *Statistical Models in Applied Science*, New York, N.Y.: John Wiley & Sons, 1975.

[7] Ross, S. M., *Introduction to Probability and Statistics for Engineers and Scientists*, New York, N.Y.: John Wiley & Sons, 1987.

CHAPTER 3

Filters, FIR, and IIR

3.1 Introduction

A digital filter can be designed in a variety of ways. A practical digital filter implementation is very dependent on the filter structure whether the filter is nonrecursive or recursive. We define the nonrecursive and recursive structure of a filter at the outset.

The output sequence y(n) of a nonrecursive filter is a function only of the past inputs and the present input x(n).

$$y(n) = \sum_{i=0}^{N} b_i\, x(n-i) \tag{3.1}$$

The transfer function of nonrecursive filter is given by, in the z domain,

$$x(z) \rightarrow \boxed{H(z)} \rightarrow y(z)$$

$$\begin{aligned} H(z) = \frac{y(z)}{x(z)} &= \sum_{i=0}^{N} b_i z^{-i} \\ &= b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_N z^{-N} \end{aligned} \tag{3.2}$$

The structure of a nonrecursive filter of order N is shown in Figure 3.1.

The output sequence y(n) of a recursive filter is a function of the past output, the past inputs and the present input x(n).

$$y(n) = \sum_{i=0}^{N} b_i\, x(n-i) - \sum_{j=0}^{N} a_j\, x(n-j) \tag{3.3}$$

The transfer function of a recursive filter in the z domain is,

$$x(n) \rightarrow \boxed{H(z)} \rightarrow y(n)$$

$$\begin{aligned} H(z) &= \frac{\sum b_i z^{-i}}{\sum a_j z^{-j}} \\ &= \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} \end{aligned} \tag{3.4}$$

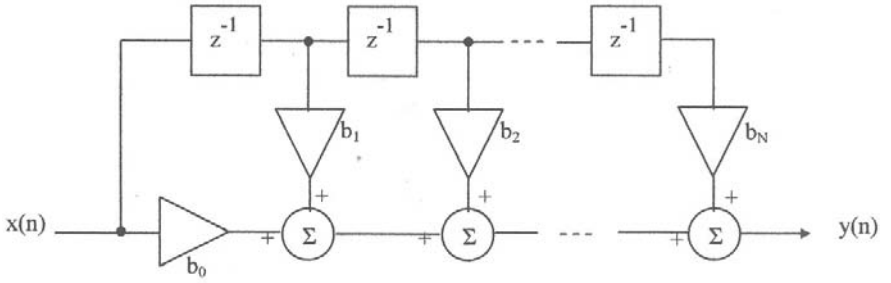


Figure 3.1 Nonrecursive filter of order *N*.

(we usually normalize a_0=1)

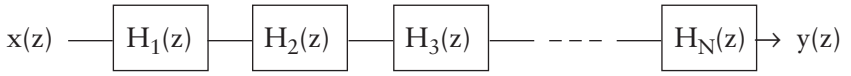
The structure of a recursive filter is shown in Figure 3.2.

The transfer function (3.4) may be implemented as shown in Figure 3.3. We call the structure a canonical form. It has the least number of delay elements but twice the number of summers.

The transfer function can be interpreted as a cascade of several transfer functions where $H_i(z)$ is either a first-order section or a second-order section.

$$H(z) = \frac{\sum b_i z^{-i}}{\sum a_j z^{-j}} = H_1(z)H_2(z)H_3(z)\cdots H_N(z)$$

$$H_{i1}(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}, \quad H_{i2}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$



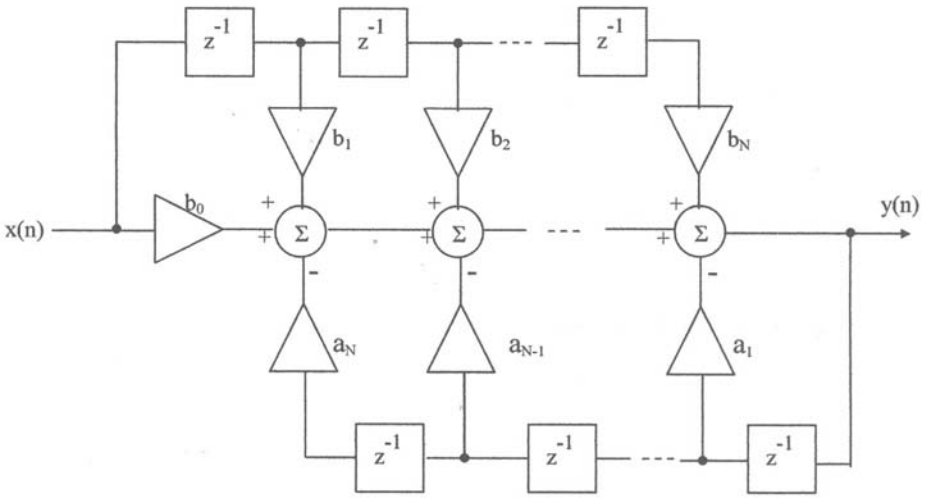


Figure 3.2 Recursive filter of order *N*.

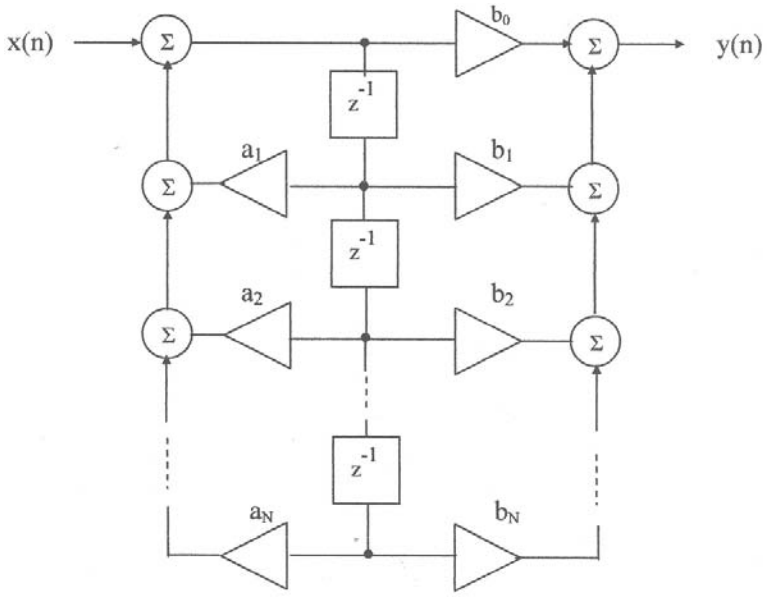


Figure 3.3 Recursive filter, canonical form.

Another variation in implementation is a partial expansion of the transfer function (3.4) (see Figure 3.4). The structure corresponding to the partial expansion is a parallel summing of all $H_i(z)$. $H_i(z)$ is either a first-order section or a second-order section.

$$H(z) = \frac{\sum b_i z^{-i}}{\sum a_j z^{-j}} = H_1(z) + H_2(z) + H_3(z) + \cdots + H_N(z)$$

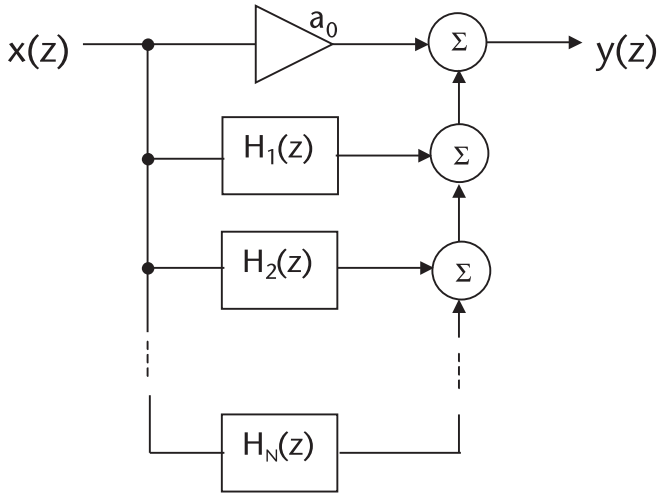


Figure 3.4 Partial fraction expansion.

There are many other structures, and they may be constructed in a countless varieties of ways. The choice among the various structures is dictated by several factors such as the economy of implementation (whether the structures are implemented by hardware or software), the degree of susceptibility to component tolerance, and the round-off errors in software.

3.2 Finite Impulse Response Filter (FIR)

A FIR for short, is usually implemented by a nonrecursive structure shown in Figure 3.1 most of the time. The output y(n) is a function of only the past inputs and the present input x(n).

$$y(n) = \sum_{i=0}^{N} b_i\, x(n-i) \tag{3.5}$$
$$= b_0(n) + b_1x(n-1) + b_2x(n-2) + \cdots + b_Nx(n-N)$$

When the input sequence ceases, the output sequence stops after an N sample delay, and therefore, we use the phrase finite duration. There is no feedback path in the structure, and it is always stable. The input and output phases hold a linear relationship. The phase linearity is an important characteristic of the nonrecursive FIR filter.

How to determine the coefficients $b_0, b_1, b_2, \ldots b_N$? Three techniques appear in the literature [1, 2]. They are listed as follows:

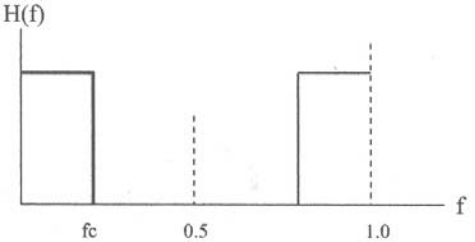
1. Impulse response method (plus window functions);
2. Frequency sampling method;
3. Optimal min-max error approximation method (Remez exchange method).

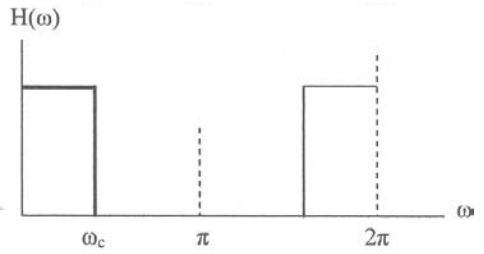
3.2.1 FIR Filters, Lowpass, Highpass, Bandpass, and Bandstop

In this section we study the impulse response plus window function. Even though this method results in a suboptimal design, the coefficients are obtained in a straightforward closed form. Those interested in the other methods should consult the references cited.

Let us study an ideal lowpass filter prototype. The response of a lowpass filter is given by,

$$H(\omega) = \begin{cases} e^{-j\omega(N-1)/2} & 0 \le \omega \le \omega_c \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$





An inverse Fourier transform of H(ω) that corresponds to the impulse response of an ideal lowpass prototype is given by,

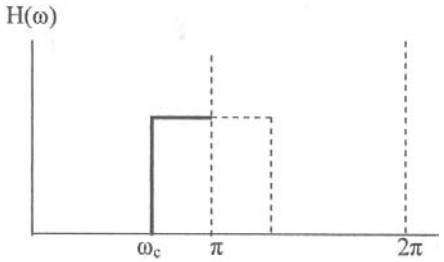
$$h(n) = \frac{1}{2\pi}\int_{-\omega}^{\omega} H(\omega)\, e^{j\omega n}\, d\omega$$

$$= \frac{\sin\left[\omega_c\left(n-\frac{N-1}{2}\right)\right]}{\pi\left(n-\frac{N-1}{2}\right)} \tag{3.7}$$

The impulse response of (3.7) holds true only when N=∞. For a finite number of FIR coefficients we must truncate N to a finite number. We call N the order of the filter. The truncation of the infinite Fourier series results in the Gibb's oscillation, or Gibb's ripple. Various window functions are applied to reduce the amplitude of the Gibb's ripples. The coefficients b_i of an FIR lowpass filter are obtained by convolving the impulse function with a selected window function.

The impulse response of an ideal, prototype, highpass filter is similarly obtained by an inverse Fourier transform.

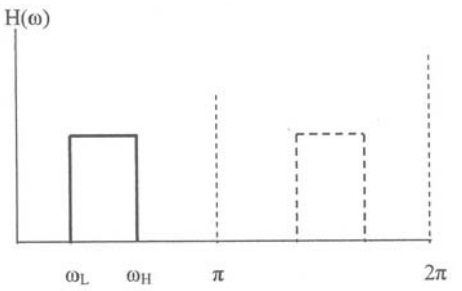
$$H(\omega) = \begin{cases} e^{-j\omega(N-1)/2} & \omega_c \le \omega \le \pi \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$



$$h(n) = \frac{\sin\left[\left(n-\frac{N-1}{2}\right)\right]}{\left(n-\frac{N-1}{2}\right)} - \frac{\sin\left[{}_c\left(n-\frac{N-1}{2}\right)\right]}{\left(n-\frac{N-1}{2}\right)} \tag{3.9}$$

For an ideal bandpass prototype, the frequency response is

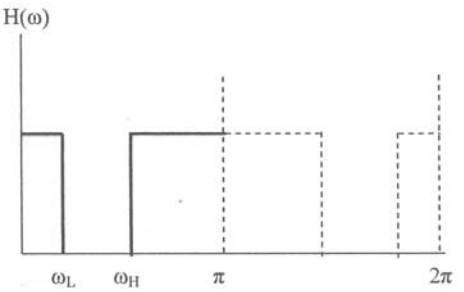
$$H(\omega) = \begin{cases} e^{-j\omega(N-1)/2} & \omega_L \le \omega \le \omega_H \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$



$$h(n) = \frac{\sin\left[\omega_H\left(n - \frac{N-1}{2}\right)\right]}{\pi\left(n - \frac{N-1}{2}\right)} - \frac{\sin\left[\omega_L\left(n - \frac{N-1}{2}\right)\right]}{\pi\left(n - \frac{N-1}{2}\right)} \tag{3.11}$$

For an ideal bandstop prototype, the frequency response is,

$$H(\omega) = \begin{cases} e^{-j\omega(N-1)/2} & 0 \le \omega \le \omega_L \text{ and } \omega_H \le \omega \le \pi \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$



The corresponding impulse function h(n) is

$$h(n) = \frac{\sin\left[\omega_L\left(n - \frac{N-1}{2}\right)\right]}{\pi\left(n - \frac{N-1}{2}\right)} + \frac{\sin\left[\pi\left(n - \frac{N-1}{2}\right)\right]}{\pi\left(n - \frac{N-1}{2}\right)} - \frac{\sin\left[\omega_H\left(n - \frac{N-1}{2}\right)\right]}{\pi\left(n - \frac{N-1}{2}\right)} \tag{3.13}$$

The impulse responses of four prototypes are programmed in IMPULSE. CPP, and the results are shown in Figures 3.5–3.8, with N=21.

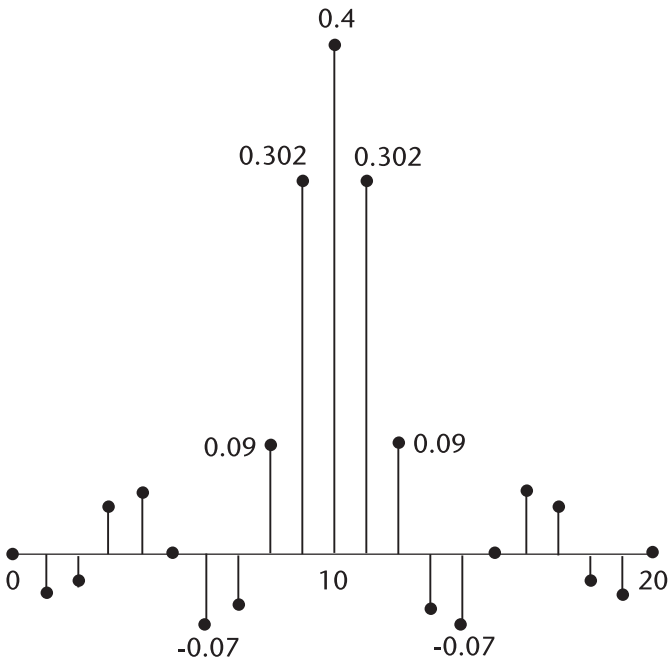


Figure 3.5 Impulse response, lowpass, f_c=0.2.

3.2.2 Window Functions: Rectangle, von Hann, Hamming, and Blackman

The impulse response h(n) will be convolved with window function w(n) to obtain the coefficients of b_i of (3.5). There are many interesting window functions [3]. We study the following four windows.

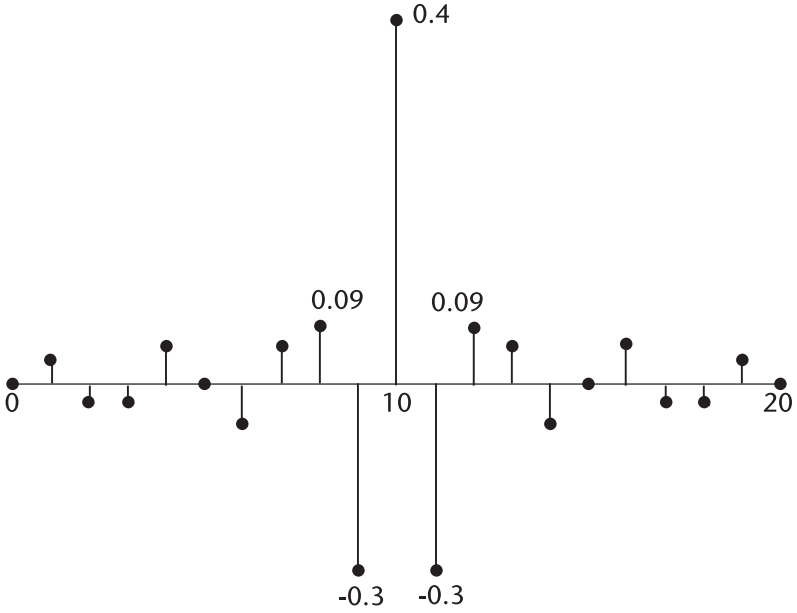


Figure 3.6 Impulse response, highpass, f_c=0.3.

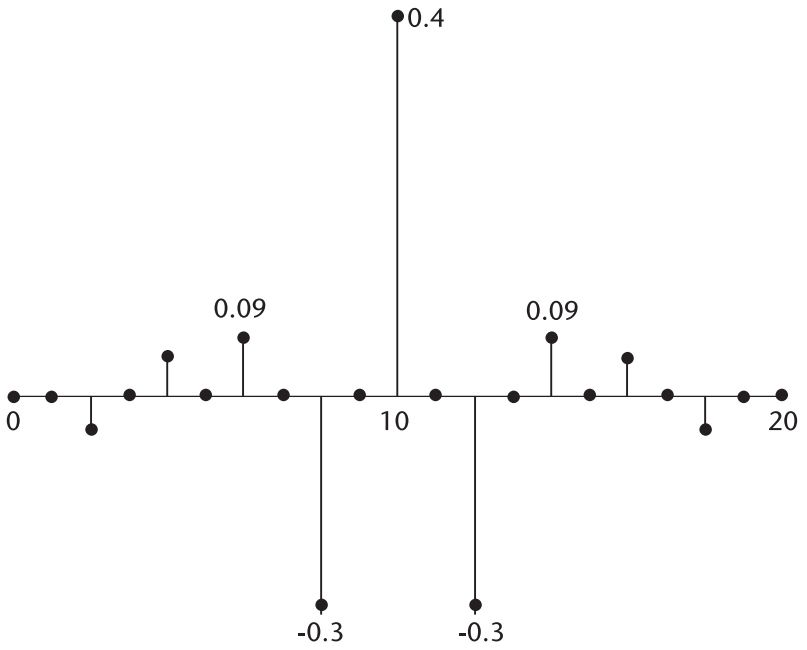


Figure 3.7 Impulse response, bandpass, f_L=0.15, f_H=0.35.

1. Rectangular window;
2. von Hann window;
3. Hamming window;
4. Blackman window.

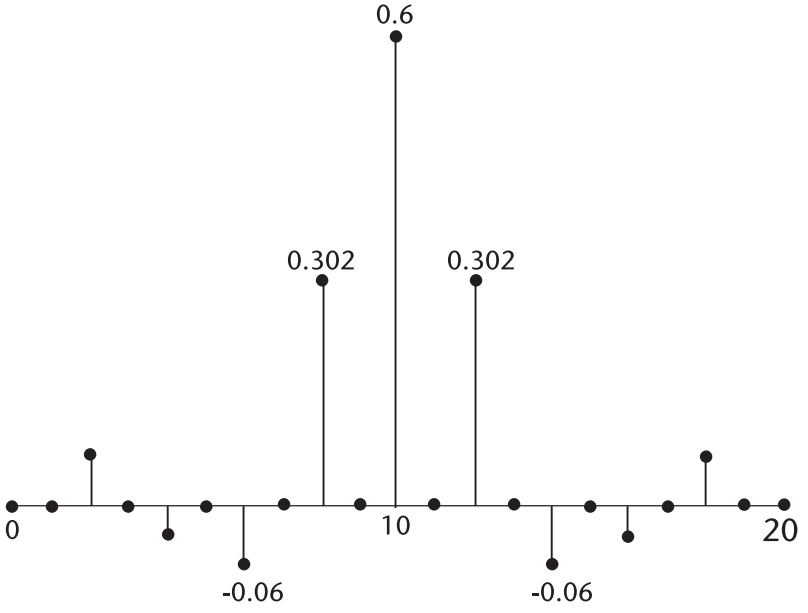


Figure 3.8 Impulse response, bandstop, f_L=0.15, f_H=0.35.

The window functions are generated as follows:

1. Rectangular: $w(n) = 1.0;$
2. von Hann: $w(n) = 0.50 - 0.50 \cos\left(\frac{2n\pi}{N}\right);$
3. Hamming: $w(n) = 0.54 - 0.46 \cos\left(\frac{2n\pi}{N}\right);$
4. Blackman: $w(n) = 0.42 - 0.50 \cos\left(\frac{2n\pi}{N}\right) + 0.08 \cos\left(\frac{4n\pi}{N}\right).$

Rectangle, von Hann, and Hamming windows are special cases of the following function which is a cosine function on a pedestal:

$$w(n) = c - (1-c) \cos\left(\frac{2n\pi}{N-1}\right)$$

where

c = 1.00: rectangle;
c = 0.50: von Hann;
c = 0.54: Hamming.

Window functions are shown in Figure 3.9.

Lowpass filters with rectangle, von Hann, Hamming, and Blackman window are programmed in FIR_LP.CPP, and the frequency responses are shown in Figure 3.10(a–c).

Highpass filters with rectangle, von Hann, Hamming, and Blackman windows are programmed in FIR_HP.CPP, and the frequency responses are shown in Figure 3.11.

The frequency responses of bandpass filters are programmed in FIR_BP. CPP, bandstop filters in FIR.BS.CPP. The results are shown in Figure 3.12 and Figure 3.13. and the stopband attenuation level. We do not have in advance any knowledge of

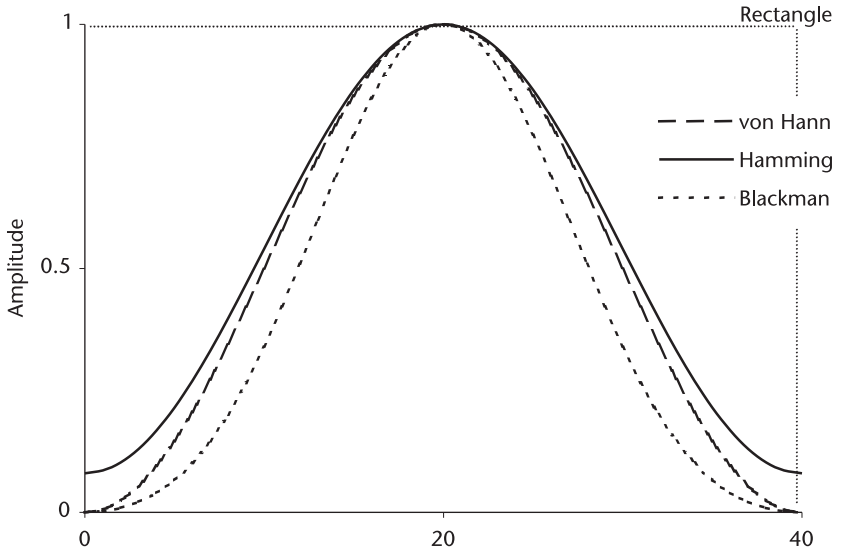


Figure 3.9 Window functions.

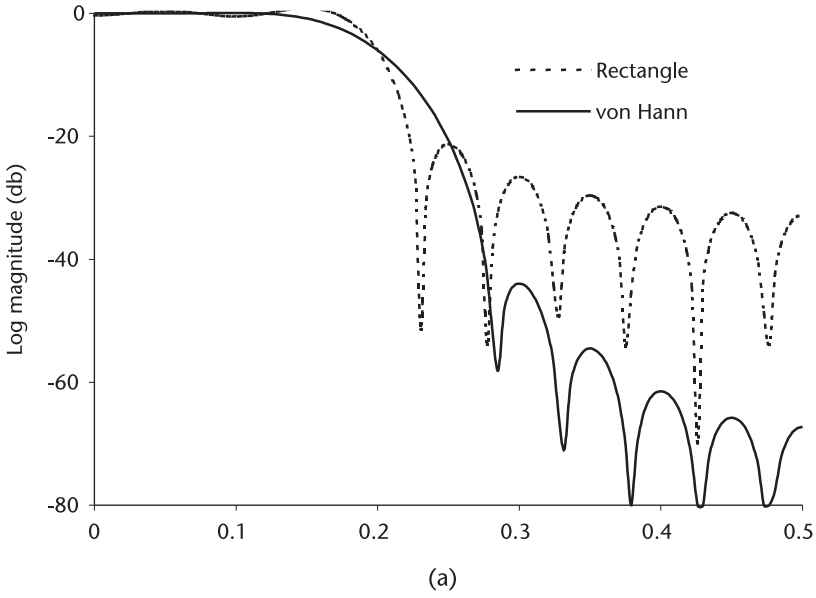


Figure 3.10(a) Lowpass prototype, f_c=0.2, N=21 (von Hann Window).

what the order of filter N would satisfy the specifications. Thus, several trial-error designs are required. Kaiser has proposed his window to remedy this problem [4, 5].

The parameter α controls the height of the pedestal when α is 4.1164.

All the filters we have designed are prototypes. In practical applications, we would convert the normalized frequency to hertz as follows.

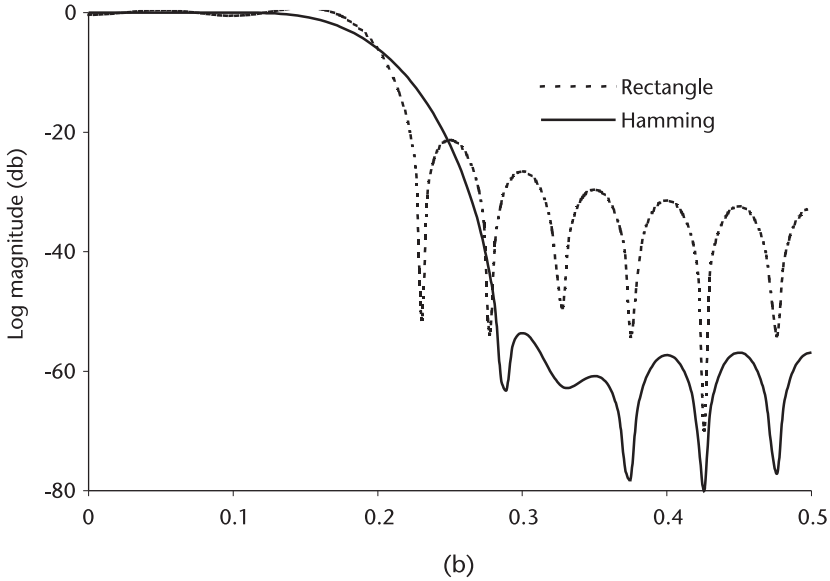


Figure 3.10(b) Lowpass prototype, f_c=0.2, N=21 (Hamming window).

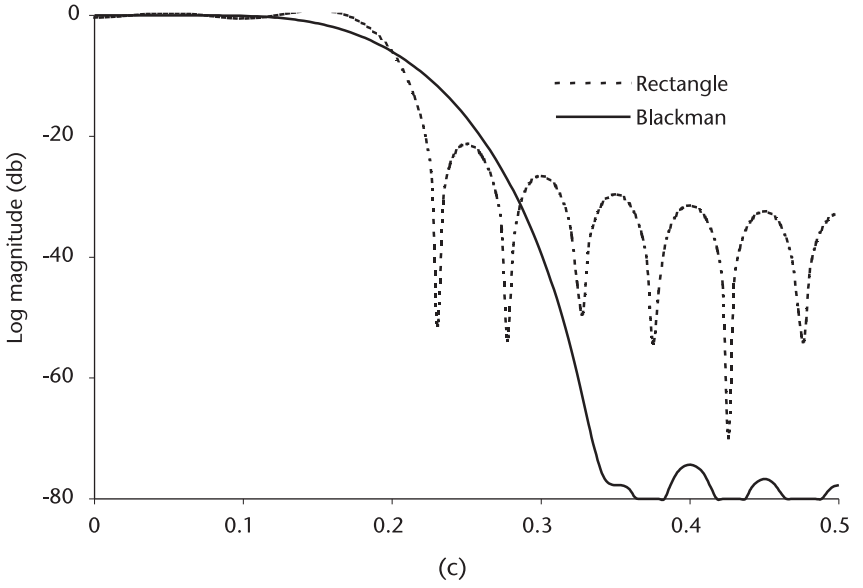


Figure 3.10(c) Lowpass prototype, f_c=0.2, N=21 (Blackman window).

Prototype	*Designed in Hertz*
$f_L = 0.15$	$f_L = 15$ KHz
$f_H = 0.35$	$f_H = 35$ KHz
$\omega_L = 2\pi f_L = 0.3\pi$	$\omega_L = 2\pi f_L T = 0.3\pi$
$\omega_H = 2\pi f_H = 0.7\pi$	$\omega_H = 2\pi f_H T = 0.7\pi$

Note: T = sampling interval of A/D converter; $\frac{1}{100\ \text{KHz}} = 1.0\times10^{-5}$ second.

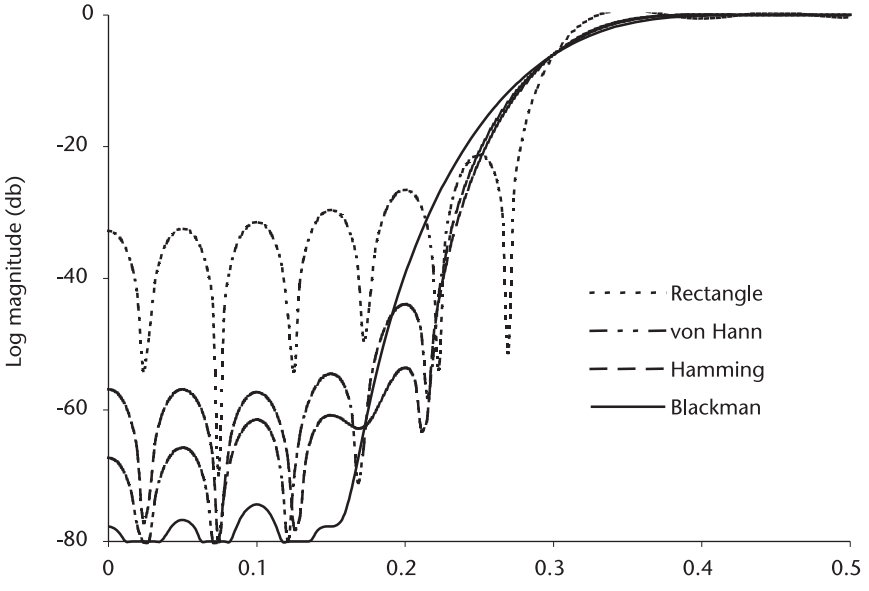


Figure 3.11 Highpass prototype, f_c=0.3, N=21.

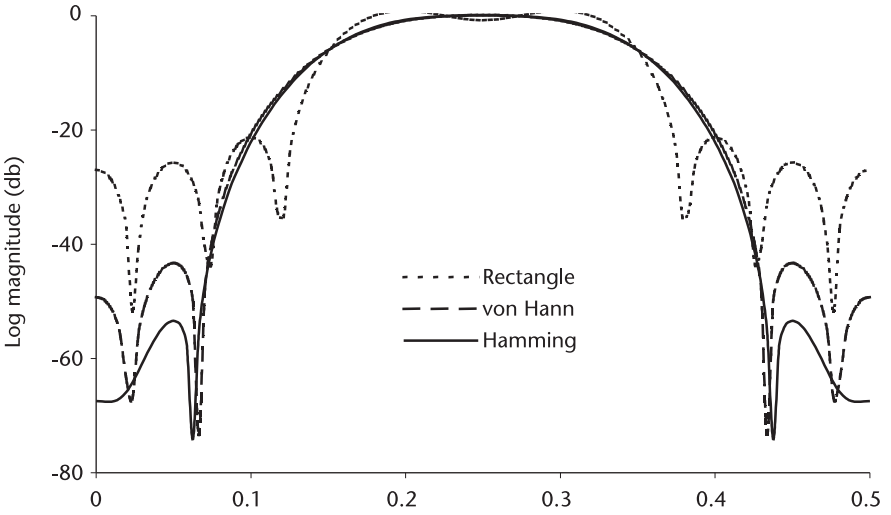


Figure 3.12 Bandpass prototype, f_L=0.15, f_H=0.35, N=21.

As demonstrated, FIR filters are simple to design. First compute the impulse response of lowpass, highpass, bandpass, or bandstop; select a window function; and convolve them. A latent shortcoming of FIR filter design is that we do not have design guidance as to what order of the filter N would meet design specifications such as the magnitude of passband ripples and the transition width.

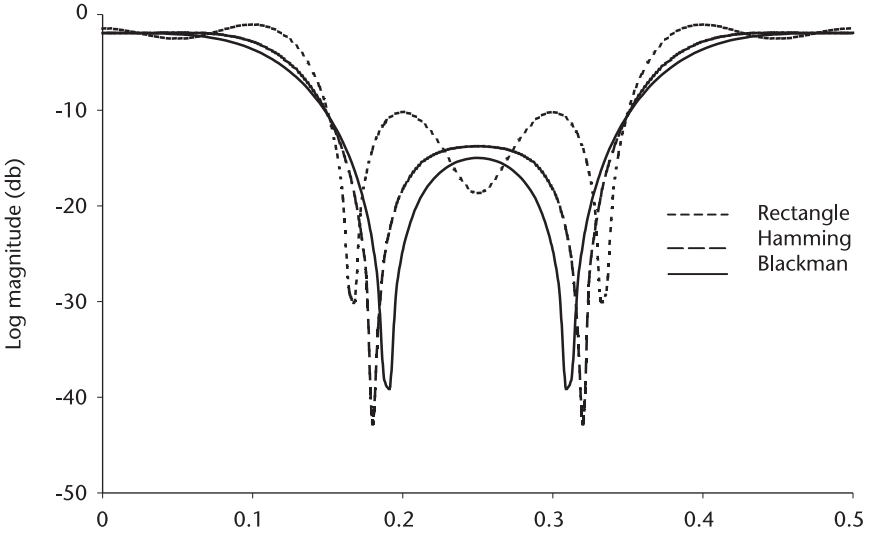


Figure 3.13 Bandstop prototype, f_L=0.15, f_H=0.35, N=21.

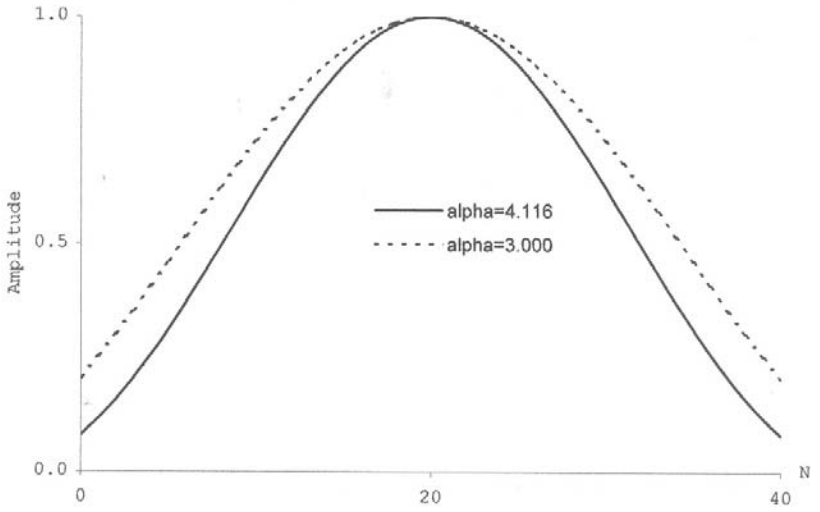
3.2.3 Kaiser Filter: Lowpass, Highpass, Bandpass, and Bandstop

The Kaiser filter (Kaiser window) is a very flexible design based on the modified Bessel function of first kind, zero order.

$$w(n) = \frac{I_o\left[\alpha\sqrt{1-\left(1-\frac{2n}{N-1}\right)^2}\right]}{I_o(\alpha)} = \frac{I_o(\beta)}{I_o(\alpha)}$$

$$I_o(x) = 1 + \sum_{i=1}^{\infty} \frac{\left[\left(\frac{x}{2}\right)^i\right]^2}{i!}$$

The pedestal height is equal to that of Hamming. When α is very large the pedestal height approaches zero. Kaiser window is shown below.



The attractive property of Kaiser filter is that the order of filter N can be determined by a single parameter α. (See two filters in Figure 3.14). All filters have specifications on the passband ripple A_p, the transition width, and the stopband attenuation level A_s. The lowpass filter has two corner frequencies, f_p and f_s. The bandpass filter has four corner frequencies, f_{s1}, f_{p1}, f_{p2}, and f_{s2}. Kaiser has empirically derived the following formula between the stopband attenuation and the parameter α,

$$\alpha = \begin{cases} 0.0 & A_s \le 21 \\ 0.5842(A_s - 21)^{0.4} + 0.07886(A_s - 21) & 21 < A_s \le 50 \\ 0.1102(A_s - 8.7) & A_s > 50 \end{cases}$$

(A_s in dB)

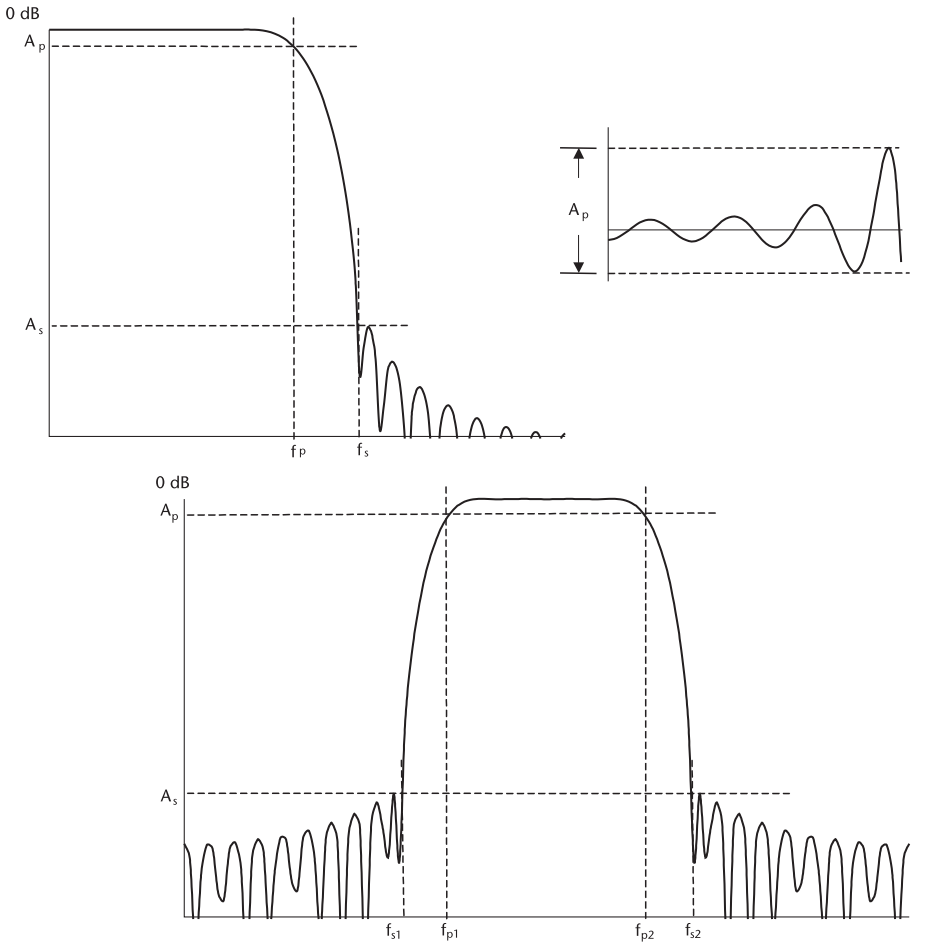


Figure 3.14 Specifications of filter design.

and an auxiliary parameter D to compute the order of filter N.

$$D = \begin{cases} 0.9222 & A_s \leq 21 \\ \dfrac{A_s - 7.95}{14.36} & A_s > 21 \end{cases} \quad (A_s \text{ is in decibel})$$

The order of filter N is given by,

$$N \geq 1 + \frac{D}{\text{transition width}}$$

The transition width shown in Figure 3.14, is defined by,

- $f_s - f_p$: lowpass;
- $f_p - f_s$: highpass;

- min $[(f_{p1} - f_{s1}), (f_{s2} - f_{p2})]$: bandpass;
- min $[(f_{s1} - f_{p1}), (f_{p2} - f_{s2})]$: bandstop.

The Kaiser filter design steps are reiterated as follows:

1. Compute the sidelobe level (decibels):

$$\Delta_1 = 10^{-0.05As} \qquad (A_s \text{ and } A_p \text{ are in dB})$$
$$\Delta_2 = \frac{10^{0.05Ap} - 1}{10^{0.05Ap} + 1}$$

$$\Delta = \min(\Delta_1, \Delta_2)$$
$$\text{sidelobe (dB)} = -20 \log(\Delta)$$

2. Compute α:

$$\alpha = \begin{cases} 0.0 & \text{SLL} \leq 21 \\ 0.5842(\text{SLL} - 21)^{0.4} + 0.07886(\text{SLL} - 21) & 21 < \text{SLL} \leq 50 \\ 0.1102(\text{SLL} - 8.7) & \text{SLL} > 50 \end{cases}$$

3. Compute D:

$$\text{D} = \begin{cases} 0.9222 & \text{SLL} \leq 21 \\ \dfrac{\text{SLL} - 7.95}{14.36} & \text{SLL} > 21 \end{cases}$$

4. Compute N:

$$\text{N} = 1 + \frac{\text{D}}{\Delta\text{F}} \qquad \Delta\text{F: minimum transition width}$$

Numerical Example: Kaiser Filter Design

A lowpass filter with the following specifications will be designed: passband ripple A_p=0.1dB, stopband attenuation A_s=40.0 dB, f_p=0.2, f_s=0.3.

1. $\Delta_1 = 0.01$

   $\Delta_2 = 0.0058$

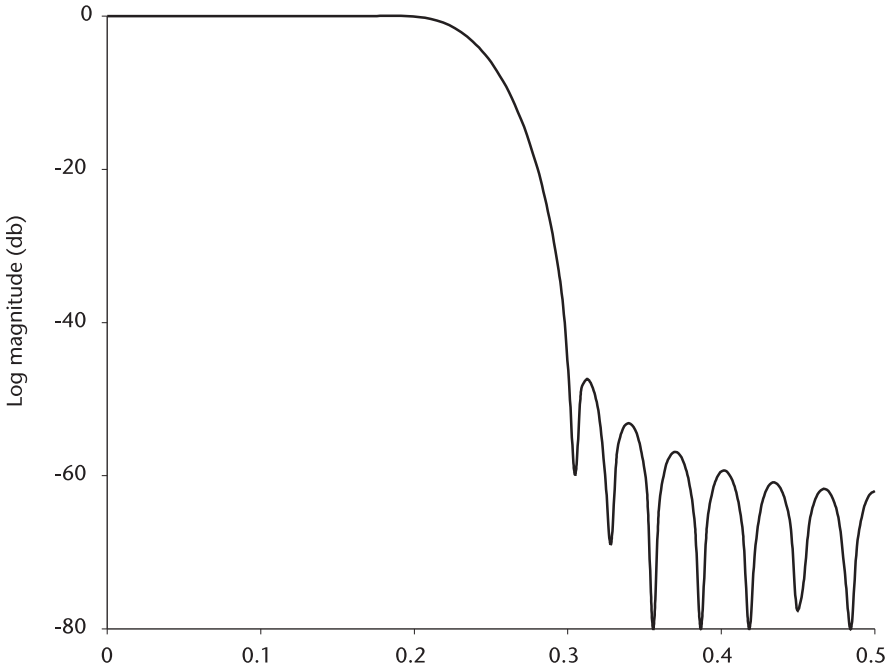


Figure 3.15 Kaiser lowpass, N=27, A_p=0.1dB, A_s=40.0 dB, f_p=0.2, f_s=0.3.

$$\Delta = \min(0.01, 0.0058) = 0.0058$$

$$\text{SLL} = -20 \log(\Delta) = 44.73 \text{ dB}$$

2. $\alpha = 0.5842\ (\text{SLL} - 21)^{0.4} + 0.07886\ (\text{SLL} - 21) = 3.945$
3. $D = \dfrac{\text{SLL} - 7.95}{14.36} = 2.56$
4. $N = 1 + \dfrac{D}{\Delta F} = 26.6 \rightarrow 27$

A Kaiser lowpass filter with the above specifications is programmed in KAISER.CPP, and the frequency response is shown in Figure 3.15. The reader may enjoy designing a highpass, bandpass, or bandstop filter with different specifications.

3.3 Infinite Impulse Response Filter (IIR)

The output of a recursive filter is a function of both past inputs and the past outputs.

$$y(n) = \sum_{i=0}^{N} b_i\ x(n-1) - \sum_{j=0}^{N} a_j\ x(n-j) \tag{3.14}$$

The structure of a recursive filter is shown in Figure 3.2. It has positive feed-forward as well as negative feedback paths. The transfer function of a recursive filter in the z-domain is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum b_i z^{-i}}{\sum a_j z^{-j}}$$
$$= \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_N z^{-N}}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}} \qquad (3.15)$$
$$(a_0 = 1)$$

Since the present output y(n) is a function of the past output y(n-i), the output duration is infinite at least theoretically, even though the output levels diminish to a negligible level very quickly. Thus, the name infinite duration impulse response filter (IIR).

The IIR filter is not always stable as the FIR filter is. The poles of H(z) in (3.15) must reside within a unit circle of the z-plane for stability. In other words, the coefficients a_j must be such that all poles p_j are within a unit circle when the denominator of H(z) is factored as,

$$X(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N} = (z - p_1)(z - p_2)\ldots(z - p_N)$$

The constraining conditions did not exist for FIR filter (feed-forward path only structures). The problem of designing an IIR filter is then to find a set of coefficients b_i and a_j of (3.15) that meet frequency response specifications such as passband ripple, narrow transition width, and stopband attenuation level with the lowest possible order N, and most of all it must be stable.

When the design domain is in z-plane, the resultant filter is a digital filter whereas in the s-domain design they are analog filters. This does not mean that all IIR digital filters are domain-transformed analog filters. Since there is a large body of theory and design techniques available for analog filter design, the domain-transform approach has become a popular method.

An IIR digital filter can be designed (determination of the coefficients a_j and b_i) by,

(1) Mapping of differentials method (backward or forward);
(2) Impulse=invariant transform method;
(3) Matched z-transform method;
(4) Bilinear transform method.

Voluminous publications appeared in the past on the first three methods. In the following sections we study exclusively the bilinear transform method, a transformation of s-domain design to z-domain design.

3.3.1 Bilinear Transform

Why the bilinear transform? The bilinear transform enables us to tap into the large body of already proven analog filter design techniques. Some analog designs have

relatively simple closed-form design formulas. A digital filter design based on an analog filter is therefore relatively simple to implement.

IIR filter designs based on the mapping of differentials (forward or backward), the impulse-invariant transform method, or the matched z-transform method have certain critical flaws. The first method fails to preserve a one-to-one relationship, and a stable analog filter may end up as an unstable digital filter. The second method results in a digital filter that is an aliased version of analog filter unless the frequency response is band-limited absolutely. The absolute band-limitedness precludes a large family of filter design. The third method is not free from defects. When an analog filter has zeroes with center frequencies greater than half the sampling frequency, z-transformed poles will be greatly aliased.

All in all, the bilinear transform is the best method for designing IIR digital filters from proven analog filters. The bilinear transform avoids the aliasing problem encountered in the other methods, but there is a small price to pay, really a small inconvenience, of introducing a frequency compression (distortion or warping) between the analog and digital frequency.

The bilinear transform is a conformal mapping from the s-plane to the z-plane. It is defined by,

$$s \rightarrow \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{3.16}$$

or

$$z \rightarrow \frac{2}{T}\left(\frac{2/T + s}{2/T - s}\right) \tag{3.17}$$

The one-to-one reversible relationship of the bilinear transform makes IIR digital filter design a simple algebraic operation in designing a stable digital filter from a stable analog filter. In order to appreciate the distortion or warping in frequency we substitute $s=j\Omega$ and $z=j\omega T$ in (3.16).

$$\begin{aligned} j\Omega &\rightarrow \frac{2}{T}\frac{1 - e^{-j\omega T}}{1 + e^{-j\omega T}} \\ &= \frac{2}{T}\frac{e^{j\omega T/2} - e^{-j\omega T/2}}{e^{j\omega T/2} + e^{-j\omega T/2}} \\ &= j\frac{2}{T}\tan\left(\frac{\omega T}{2}\right) \end{aligned} \tag{3.18}$$

Equation (3.18) is graphically shown in Figure 3.16. We note that the analog frequency Ω is compressed (distorted or warped) in the digital frequency ωT.

Equally spaced analog frequencies $\Omega_1, \Omega_2\ \Omega_3, \ldots$ are compressed to or stretched to $\omega_1 T, \omega_2 T, \omega_3 T, \ldots$ in the digital frequencies.

To design a digital filter with specified corner frequencies at $\omega_1 T, \omega_2 T, \omega_3 T, \ldots$, all we have to do is to decompress (predistort or prewarp) the analog frequencies

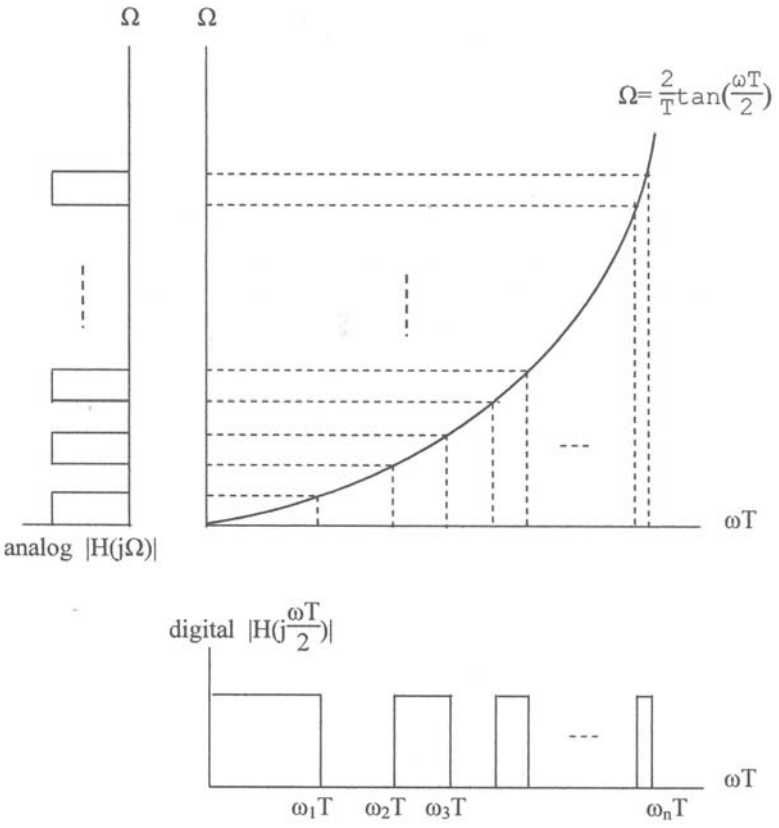


Figure 3.16 Frequency warping between analog and digital frequency due to bilinear transform.

$\Omega_1, \Omega_2 \Omega_3, \ldots$. With this in mind, all the stable analog filter transfer functions H(s) can be transformed to the corresponding stable digital IIR filter transfer functions H(z) by a simple substitution of (3.16).

$$H(z) = H(s)\Big|_{s} \tag{3.19}$$

$$s = \frac{2}{T}\Big/\frac{8-1}{8+1}$$

The mapping strategy of (3.16) or (3.17) is shown in Figure 3.17. From the equations and figure, we see that when s=0, z=1. When σ=0 and Ω=±j∞, z=–1. The positive imaginary axis in the s-plane is mapped into the upper half circle in the z-plane, and the negative imaginary axis in the s-plane is mapped into a lower circle in the z-plane. All the complex roots located in the left half of the s-plane would be mapped inside the unit circle in the z-plane.

3.3.2 Review of Analog Filters

Since the z-plane IIR digital filter designs are derived from the s-plane analog filter designs, we review analog filter design briefly. It is not a thorough, in-depth study but just adequate enough to understand analog filter design, so that we can

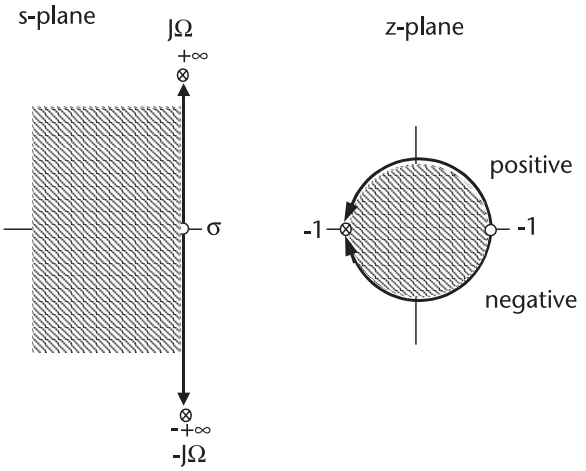


Figure 3.17 Mapping of s-plane(analog) to z-plane(digital) by bilinear transform.

transform analog design techniques to digital design. We shall cover Butterworth, Chebyshev, Inverse Chebyshev, and elliptical filters.

Butterworth Lowpass

The Butterworth lowpass filter is characterized by the property that the frequency response is maximally flat at the origin and monotonically decreasing in the stop-band. The transition width is moderately wide.

The frequency response of Butterworth lowpass prototype is given by,

$$|H(s)|^2 = |H(j\,\Omega)|^2 = \frac{1}{1 + (\Omega/\Omega_c)^{2N}} \tag{3.20}$$

N is the order of Butterworth filter. (See Figure 3.18.)

Equation 3.20 can be rewritten as

$$H(s)H(-s) = \frac{1}{1 - (-s^2)^N} \tag{3.21}$$

The poles of (3.21) are located equally spaced on a unit circle on the s-plane. Figure 3.19 shows the locations of poles on left-half of the s-plane, N=7 assumed. When we take these poles on the left-half s-plane,

$$H(s) = \frac{C_o}{(s - p_1)(s - p_2)\ldots(s - p_N)} \tag{3.22}$$

where $p_1, p_2, p_3, \ldots p_N = \exp\{j\pi\,[1/2+(2k-1)/2N]\}$, k=1, 2, 3, . . . N. and, the constant C_o makes H(s)=1 when s=0.

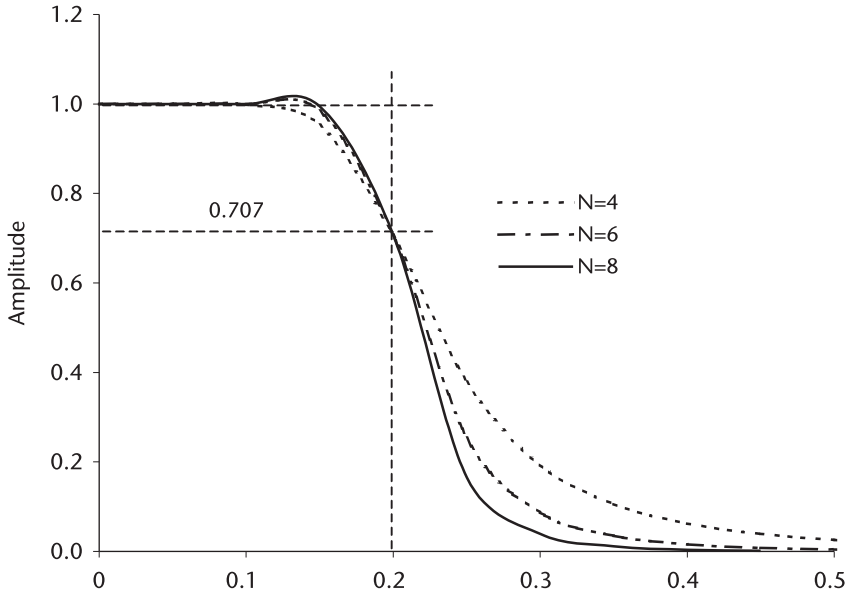


Figure 3.18 Butterworth lowpass filter, analog.

From (3.20) and (3.22) we see that:

1. The Butterworth lowpass filter is an all-pole design;
2. At $\Omega=\Omega_c$, the frequency response is down 3 dB;
3. The frequency response is monotonic, and the transition width would become narrower as the order of filter N increases.

Butterworth filter design is generally determined by specifying the attenuation level A at Ω_c. The order of filter N is given by,

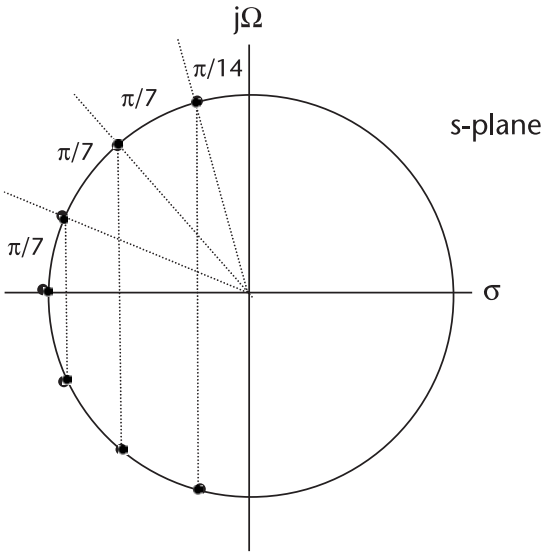


Figure 3.19 Pole locations of butterworth lowpass, N = 7.

$$N = \frac{\log_{10}(A^2 - 1)}{2\ \log_{10}(\Omega_c)}$$

For example, A=1/100 (-40.0 dB), at Ω_c=2.0 in a prototype lowpass. The required order of the filter N is,

$$N = \frac{\log_{10}(100^2 - 1)}{2\ \log_{10}(2)} = 6.64 \rightarrow 7$$

The pole locations of the Butterworth lowpass in (3.22) are listed in Table 3.1 together with the factored form of Butterworth polynomials. Later we show that the factored forms are domain transformed to the z-plane to design IIR digital lowpass filters. A simple conversion technique from the prototype lowpass to highpass, bandpass, or bandstops filter will be shown at the end of the review of analog filters.

The factors of Butterworth polynomials are listed as follows.

N	H(s)
2	$(s^2+1.4142s+1)$
3	$(s+1)(s^2+s+1)$
4	$(s^2+0.7654s+1)(s^2+1.8478s+1)$
5	$(s+1)(s^2+0.6180s+1)(s^2+1.6180s+1)$
6	$(s^2+0.5176s+1)(s^2+1.4142s+1)(s^2+1.9319s+1)$
7	$(s+1)(s^2+0.4450s+1)(s^2+1.2470s+1)(s^2+1.8019s+1)$

Chebyshev Lowpass

The relatively wide transition width of Butterworth lowpass can be made narrower by the Chebyshev lowpass. The Chebyshev filter (many designers call it type 1 Chebyshev) is an all-pole design that exhibits equal ripple in the passband and monotonic in the stopband.

Table 3.1 Complex Roots of Butterworth Polynomials [6, 7]

N=2	*N=3*	*N=4*	*N=5*	*N=6*	*N=7*
-0.707107	-1.000000	-0.382683	-1.000000	-0.258819	-1.000000
±j0.707107		±j0.923879		±j0.965926	
	-0.500000	-0.923879	-0.309017	-0.707107	-0.222521
	±j0.866025	±j0.382683	±j0.951056	±j0.707107	±j0.974928
			-0.809017	-0.965926	-0.623489
			±j0.587785	±j0.258819	±j0.781832
					-0.900969
					±j0.433884

$$|H(\Omega)|^2 = \frac{1}{1 + \varepsilon^2\, T_N^2(\Omega/\Omega_p)}$$

where ε is a parameter that controls the magnitude of ripple, and $T_N(\cdot)$ is an Nth order Chebyshev polynomial defined by,

$$T_N(x) = \begin{cases} \cos[N \cos^{-1}(x)] & \Omega \le \Omega_p \\ \cosh[N \cosh^{-1}(x)] & \Omega > \Omega_p \end{cases}$$

Chebyshev polynomials have a recurrence formula:

N	$T_N(x)$
1	$\cos\,[\cos^{-1}(x)] = x$
2	$\cos\,[2\,\cosh^{-1}(2x)] = 2x^2 - 1$
3	$\cos\,[3\,\cosh^{-1}(3x)] = 4x^3 - 3x$
.	.
.	.
.	.
$N+1$	$\cos\,[(N+1)\,\cosh^{-1}((N+1)x)] = 2x\,T_N(x) - T_{N-1}(x)$

The frequency response of the Chebyshev lowpass when N is odd is shown in Figure 3.20. When the order N is even the black dot moves to the $1/(1+\varepsilon^2)$ level.

The poles are located on an ellipse in the left-half of s-plane. The major and minor axis of the ellipse are given by,

$$\text{major axis:} \qquad \frac{1}{2}(\gamma + \gamma^{-1})$$

$$\text{minor axis:} \qquad \frac{1}{2}(\gamma - \gamma^{-1})$$

$$\text{where} \quad \gamma = \left[\frac{1+\sqrt{1+\varepsilon^2}}{\varepsilon}\right]^{1/N}$$

The locations of the Nth order Chebyshev lowpass are given by,

$$\sigma_k = \sinh\varphi \sin\left[\frac{(2k-1)\pi}{2N}\right] \qquad k = 1,2,3,\ldots N$$

$$\Omega_k = \cosh\varphi \cos\left[\frac{(2k-1)\pi}{2N}\right]$$

where

$\sinh \varphi =$ minor axis;
$\cosh \varphi =$ major axis.

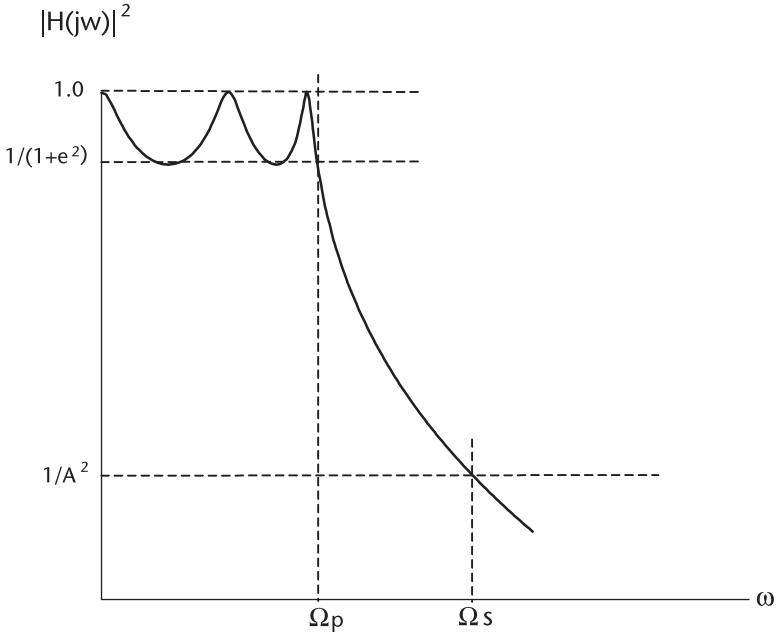


Figure 3.20 Chebyshev lowpass, (type 1).

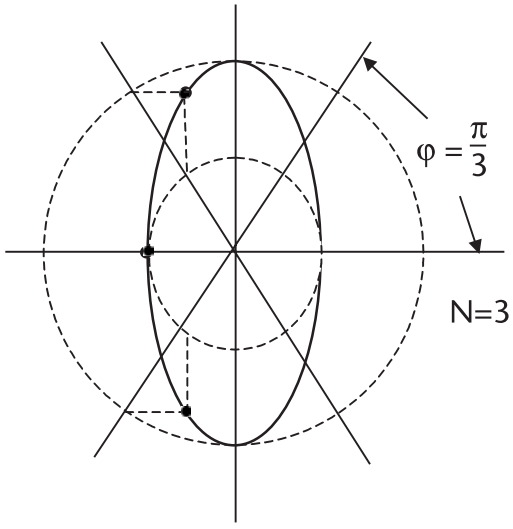


Figure 3.21 Pole locations of Chebyshev on ellipse.

Table 3.2(a) Complex Roots of Chebyshev Polynomials (0.5-dB ripple, ε^2=0.1220184)

N=2	*N=3*	*N=4*	*N=5*	*N=6*	*N=7*
-0.712812 ±j1.004043	-0.626457	-0.175353 ±j1.016253	-0.362319	-0.077650 ±j1.008461	-0.256170
	-0.313228 ±j1.021928	-0.423339 ±j0.420946	-0.111963 ±j1.011557	-0.212144 ±j0.738245	-0.057003 ±j1.006409
			-0.293123 ±j0.625177	-0.289794 ±j0.270216	-0.159719 ±j0.807077
					-0.230801 ±j0.447894

Factors of Chebyshev Polynomials (0.5-dB ripple, ε^2=0.1220184)

N	*H(s)*
2	$(s^2+1.4256s+1.5162)$
3	$(s+0.6265)(s^2+0.6265s+1.1424)$
4	$(s^2+0.3507s+1.0635)(s^2+0.8467s+0.3564)$
5	$(s+0.3623)(s^2+0.2239s+1.0358)(s^2+0.5362s+0.4768)$
6	$(s^2+1.1553s+1.0230)(s^2+0.4243s+0.5900)(s^2+0.5796s+0.1570)$
7	$(s+0.2562)(s^2+0.1140s+1.0161)$ $(s^2+0.3194s+0.6769)(s^2+0.4616s+0.4314)$

Table 3.2(b) Complex Roots of Chebyshev Polynomials (1.0-dB ripple, ε^2=0.2589254)

N=2	*N=3*	*N=4*	*N=5*	*N=6*	*N=7*
-0.548867 ±j0.895129	-0.494171	-0.139536 ±j0.983379	-0.289493	-0.062181 ±j0.943412	-0.205416
	-0.247085 ±j0.965998	-0.336869 ±j0.407329	-0.089158 ±j0.990107	-0.169882 ±j0.727228	-0.045709 ±j0.995284
			-0.234205 ±j0.611919	-0.232063 ±j0.266184	-0.128074 ±j0.798156
					-0.185072 ±j0.442943

Factors of Chebyshev Polynomials (1.0-dB ripple, ε^2=0.2589254)

N	*H(s)*
2	$(s^2+1.0977s+1.1085)$
3	$(s+0.4942)(s^2+0.4942s+0.9942)$
4	$(s^2+0.2791s+0.9865)(s^2+0.6737s+0.2794)$
5	$(s+0.2895)(s^2+0.1789s+0.9883)(s^2+0.4684s+0.4293)$
6	$(s^2+0.1244s+0.8939)(s^2+0.3398s+0.5577)(s^2+0.4641s+0.1247)$
7	$(s+0.2054)(s^2+0.0914s+0.9926)$ $(s^2+0.2561s+0.6536)(s^2+0.3701s+0.2305)$

Table 3.2 lists the pole locations for ε^2=0.1220184 (0.5-dB ripple), and ε^2=0.2589254(1.0-dB ripple), N=2 to N=7. More extensive tables are available elsewhere. Later we show the factored forms are domain transformed to the z-plane to design IIR lowpass Chebyshev filters.

In order to design Chebyshev lowpass filters the following specifications are required:

1. Magnitude of ripple ε^2 up to Ω_p;
2. Attenuation A at Ω_a;
3. The order of filter N.

When we specify two design specifications the third will be uniquely determined. The order N to achieve ε^2 at Ω_p=1.0, A at Ω_a is given by,

$$N = \frac{\log(D + \sqrt{D^2 - 1})}{\log(\Omega_a/\Omega_p + \sqrt{(\Omega_a/\Omega_p)^2 - 1})}$$

where $D = \dfrac{\sqrt{A^2 - 1}}{\varepsilon}$

Please note the following example. (See Figure 3.20.)

1. Passband ripple=3.0 dB, up to Ω_p=1.0;
2. Attenuation level =40.0 dB at Ω_a=2.0.

then

$$20 \log_{10} \frac{1}{\sqrt{1 + \varepsilon^2}} = -3.0(\text{dB}), \quad \varepsilon^2 = 0.9953$$

$$20 \log_{10} \frac{1}{\sqrt{A}} = -40.0\,(\text{dB}), \qquad A = 10{,}000$$

$$D = \frac{\sqrt{A^2 - 1}}{\varepsilon} = 10{,}047$$

$$N \approx 7.52$$

Thus, N = 8 will meet the design goal.

The locations of complex roots of Chebyshev polynomials and the factored forms are listed in Table 3.2, up to N=7. The extended tables are found in the references.

Inverse Chebyshev Lowpass (Figure 3.22)

The inverse Chebyshev lowpass filter exhibits a monotonic response in the passband and ripples in the stopband. The inverse Chebyshev is sometimes called Chebyshev Type 2. The transfer function is given by,

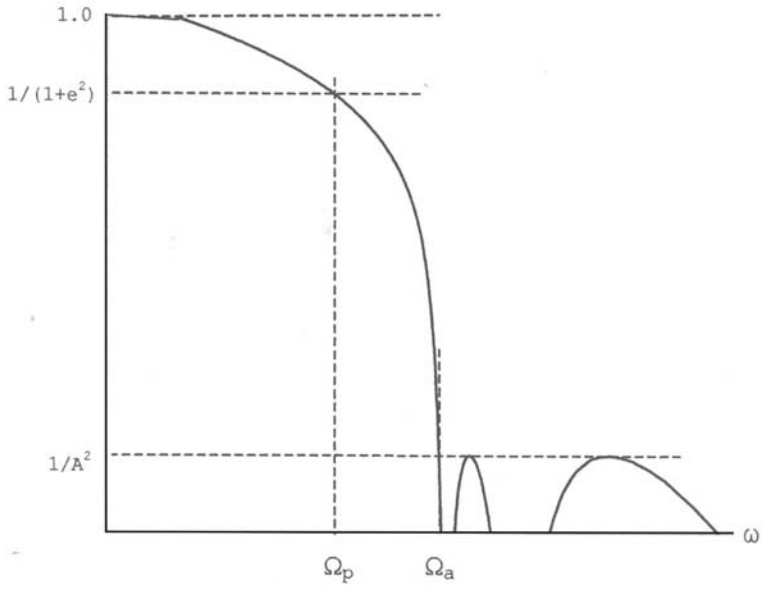


Figure 3.22 Inverse Chebyshev, lowpass.

$$|H(\Omega)|^2 = \frac{T_N^2(\Omega_a/\Omega)}{T_N^2(\Omega_a/\Omega) + \varepsilon^2\ T_N^2(\Omega_a/\Omega_p)}$$

The transfer function has both zeroes and poles in the left half of the s-plane. The zeroes are purely imaginary, and are located at

$$\text{zero}_k = \frac{j\Omega}{\cos\left[\frac{(2k-1)\pi}{2N}\right]} \quad k = 1, 2, 3, \ldots N$$

The locations of poles are given by

$$\sigma_k = \frac{\Omega_k a_k}{a_k^2 + b_k^2}, \quad \Omega_k = \frac{\Omega_k b_k}{a_k^2 + b_k^2}$$

where

$$a_k = -\sinh\varphi\ \sin\left[\frac{(2k\text{-}1)\pi}{2N}\right] \qquad (k=1, 2, 3, \ldots N);$$

$$b_k = \quad \cosh\varphi \cos\left[\frac{(2k\text{-}1)\pi}{2N}\right];$$

$$\sinh\varphi = 12(\gamma - \gamma^{-1});$$

$$\cosh\varphi = 12(\gamma + \gamma^{-1});$$

$$\gamma = \left[A+\sqrt{A^2\text{-}1}\ \right]^{1/N}.$$

We note that a zero is located at infinity on the imaginary axis when N is odd, and the poles are not located on an ellipse as in the Chebyshev Type 1. A table similar to Table 3.2(a,b) can be generated. The order of filter N of inverse Chebyshev is identical to Type 1:

$$N = \frac{\log(D + \sqrt{D^2 - 1})}{\log(\Omega_a/\Omega_p + \sqrt{(\Omega_a/\Omega_p)^2 - 1})}$$

where

$$D = \frac{\sqrt{A^2 - 1}}{\varepsilon}$$

Elliptic lowpass (Figure 3.23)
The last analog filter we review is the elliptic filter (sometimes called the Cauer filter). An elliptic filter has ripples both in the passband and stopband. The transfer function is given by

$$|H(\Omega)|^2 = \frac{1}{1+\varepsilon^2 R_N^2(\Omega k)}$$

where $R_N(\cdot)$ is the Jacobian elliptical function. The elliptical filter has the narrowest transition width among all filters, given an identical frequency response specification. In this sense, we say that the elliptical filter is an optimal filter; however, computations of zeroes and poles locations are much more complicated.

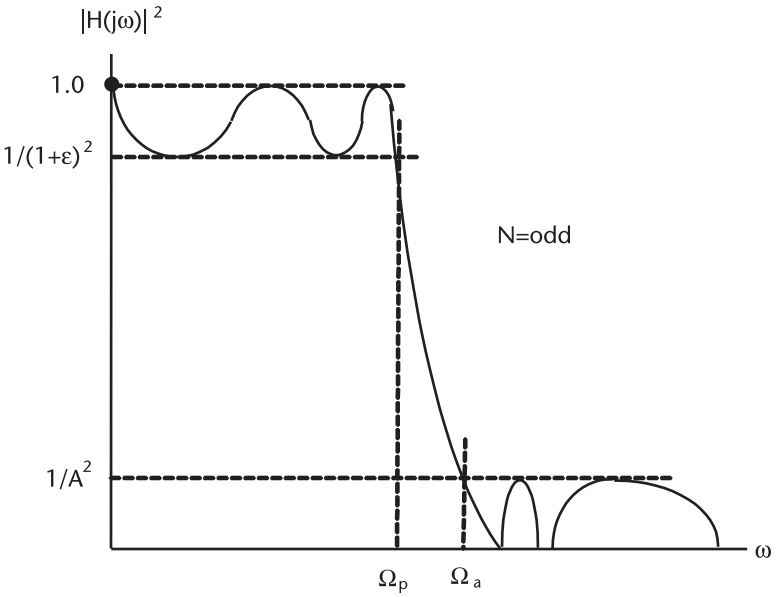


Figure 3.23 Elliptic lowpass.

The theory of elliptical filter design may be mastered from the potential theory analogy, or from Chebyshev rational functions [7–9]. We omit the derivation for the locations of zeroes and poles, for a satisfactory exposition takes more than dozen scores of pages; the subject deserves a separate textbook. Interested readers should consult the references mentioned.

Why an elliptic filter? In the microwave region where waveguides are used, the optimality (the lowest order N among the competing designs) translates to a shorter length of heavy guides and a smaller number of tuning elements. The controlling issues are the weight and the economics of production.

Instead of deriving the locations of zeroes and poles we suggest the use of design tables already published [10,11]. The handbooks have compiled hundreds of design data covering almost all conceivable elliptical filters. Su [10] has even included an interpolation technique in case his table does not exactly match one's specification. His handbook presents the results in factored forms that are much more convenient in transforming to IIR digital filters. For example,

$$H(s) = h_e \frac{(s^2 + \omega_1^2)(s^2 + \omega_2^2)}{(s^2 + a_1 s + b_1)(s^2 + a_2 s + b_2)} \quad \text{(fourth order, N even)}$$

$$H(s) = h_o \frac{(s^2 + \omega_1^2)(s^2 + \omega_2^2)}{(s + a_o)(s^2 + a_1 s + b_1)(s^2 + a_2 s + b_2)} \quad \text{(fifth order, N odd)}$$

h_e and h_o are the constants to make H(s)=1 when all s' are zero.

At the outset we will compute the order N for a speedy search for the best design data in his thick volume.

$$N = \frac{K(k)}{K(k_1)} \frac{K(\sqrt{1 - k_1^2})}{K(\sqrt{1 - k^2})}$$

$$k = \frac{\Omega_p}{\Omega_a}, \quad k_1 = \frac{\varepsilon}{\sqrt{A^2 - 1}}$$

$$K(k) = \int_0^{\pi/2} \frac{d\varphi}{[1 - k^2 \sin^2 \varphi]^{1/2}} \quad \text{(the complete elliptical integral)}$$

As an example, consider the following: We plan to design an elliptic lowpass that meets the following specifications. See Figure 3.23.

$1/(1+\varepsilon^2)$ = 0.5 dB;
$1/A^2 \geq 50.0$ dB;
Ω_p = 1.0 rad/sec;
$\Omega_a \leq 1.5$ rad/sec.

then,

$$k = \frac{1}{1.5}, \quad k_1 = \frac{0.349311}{\sqrt{1 \times 10^5 - 1}} = 0.001105$$

$$N = \frac{K(0.6667)}{K(0.001105)} \frac{K(0.9999)}{K(0.7453)} \approx 4.29$$

For computation of K(·), see K(m)_JAC.CPP and K(m)_ELP. CPP in Chapter 12.

N = 5 should satisfy our design specifications. We find candidates design data fairly quickly in Su [10] (Table 2.52 and Table 3.66).

$$H_1(s) = \frac{0.01918}{(s + 0.4254)} \frac{(s^2 + 2.4255)}{(s^2 + 0.1635s + 1.0319)} \frac{(s^2 + 5.4376)}{(s^2 + 0.5702s + 0.5761)}$$

$$H_2(s) = \frac{0.02029}{(s + 0.4279)} \frac{(s^2 + 2.3747)}{(s^2 + 0.1619s + 1.0319)} \frac{(s^2 + 5.3018)}{(s^2 + 0.5695s + 0.5789)}$$

The first transfer function gives A=50.61 dB at Ω_a=1.5 rad/sec; the second transfer function gives A=50.0 dB at Ω_a=1.4847 rad/sec. The factored form would be prewarped and bilinear-transformed to obtain the coefficients b_i and a_j of the IIR filter.

The following tabulation shows the order of filter N required to meet the specifications of the above example.

- Elliptic: $N = \frac{K(k)}{K(k_1)} \frac{K(\sqrt{1 - k_1^2})}{K(\sqrt{1 - k^2})} = 4.29, \quad N = 5;$
- Chebyshev: $N = \frac{\log(D + \sqrt{D^2 - 1})}{\log(\Omega_a/\Omega_p + \sqrt{(\Omega_a/\Omega_p)^2 - 1})} = 10.9, \quad N = 11;$
- Butterworth: $N = \frac{\log(A^2 - 1)}{2\ \log(\Omega_c)} = 28.39, \quad N = 29.$

Frequency Band Transformation

So far we have reviewed the design procedures for analog lowpass filters in prototype. The lowpass prototype would be transformed to other lowpasses with different cutoff frequencies, to a highpass, to a bandpass, or to a bandstop filter. The following substitutions would accomplish the transformations.

- Lowpass-to-lowpass: $s \rightarrow \frac{s}{\Omega n_L}$ (n_L: new low cutoff frequency);
- Lowpass-to-highpass: $s \rightarrow \frac{\Omega n_H}{s}$ (n_H: new high corner frequency);
- Lowpass-to-bandpass: $s \rightarrow \frac{s^2 + \Omega_L \Omega_H}{s\Omega_H - \Omega_L}$ (Ω_L, Ω_H: new low, high frequency);
- Lowpass-to-bandstop: $s \rightarrow \frac{s\Omega_H - \Omega_L}{s^2 + \Omega_L \Omega_H}$.

New transfer function H(s) will first be frequency-prewarped and bilinear-transformed to obtain the coefficients b_i and a_j for IIR digital filters. A few demonstrations will be given in the next sections.

3.3.3 IIR Filter, Butterworth Lowpass

A design procedure for an IIR Butterworth lowpass prototype, N=2 will be shown. We have chosen deliberately a low order to learn the prewarp operation and bilinear transform. The steps in this process are listed as follows.

1. Selected analog transfer function H(s) of Butterworth, N=2.

$$H(s) = \frac{1}{1 + \sqrt{2}s + s^2} = \frac{B_0 + B_1 s + B_2 s^2}{A_0 + A_1 s + A_2 s^2}$$

2. Specify the cutoff frequency.

$$f_c = 0.2 \text{ Hz}, \quad \omega_c T = 2\pi f_c T = 0.4\pi \text{ rad/sec}$$

3. Prewarp the analog frequency.

$$H(s) = \frac{1}{1 + \sqrt{2}(s/\Omega_c) + (s/\Omega_c)^2} = \frac{\Omega_c^2}{\Omega_c^2 + \sqrt{2}\Omega_c s + s^2}$$

Substitute $\Omega_c = 2\tan(\frac{\omega T}{2})$

$$H(s) = \frac{B_{0PW}}{A_{0PW} + A_{1PW}s + A_{2PW}s^2}$$

4. Bilinear transform the prewarped H(s) by substituting $s = 2\left[\frac{z-1}{z+1}\right]$

$$H(s) = \frac{B_{0PW}}{A_{0PW} + A_{1PW}s + A_{2PW}s^2} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

(normalize the coefficients so that a_0=1)

The steps described above have been programmed in IIR_B_LP.CPP; B stands for Butterworth, LP for lowpass. The result is shown in Figure 3.24. The slope of the frequency response is not very impressive, N being only 2. The program IIR_B_LP.CPP can be extended to accommodate a higher order. Regardless of the order, all Butterworth lowpasses are characterized by a 3-dB drop at the corner frequency and a monotonic response throughout the range. We note that the phase and time delay are nonlinear. The nonlinearity in phase and time delay would become a critical issue if any subsequent signal processing requires linearity. More about that later.

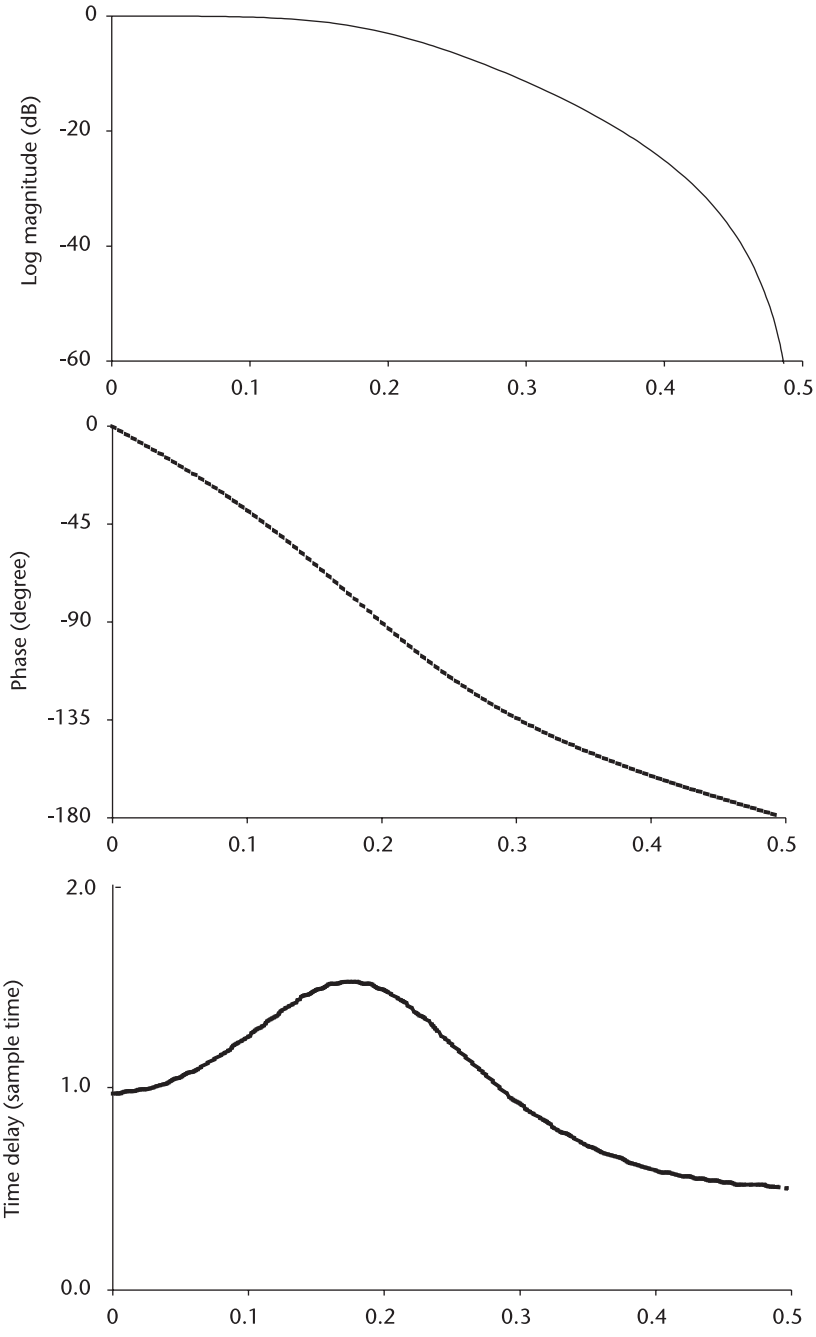


Figure 3.24 Butterworth lowpass, N = 2, $\omega_p = 0.2$.

3.3.4 IIR Filter, Chebyshev Lowpass

In this demonstration program the prewarping and bilinear transform operations are executed in two separate header files; PRE_WARP.H and BILINEAR.H since these two operations are repetitively needed in an IIR filter designs.

We take up the following analog transfer function, N=5, 1.0-dB ripple, Table 3.2(b).

$$H(s)\frac{B(s)}{A(s)} = \frac{1}{(s+0.2895)}\frac{1}{(s^2+0.1789s+0.9883)}\frac{1}{(s^2+0.4684s+0.4293)}$$

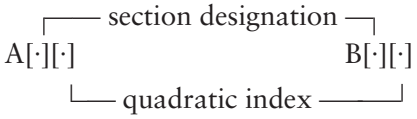
The denominator A(s) has one simple factor and two quadratic factors.

$$\text{Section '0'} = \frac{1}{(s+0.2895)} \qquad \rightarrow \frac{B[0][0]+B[0][1]+B[0][2]}{A[0][0]+A[0][1]+A[0][2]}$$

$$\text{Section '1'} = \frac{1}{(s^2+0.1789s+0.9883)} \rightarrow \frac{B[1][0]+B[1][1]+B[1][2]}{A[1][0]+A[1][1]+A[1][2]}$$

$$\text{Section '2'} = \frac{1}{(s^2+0.4684s+0.4293)} \rightarrow \frac{B[2][0]+B[2][1]+B[2][2]}{A[2][0]+A[2][1]+A[2][2]}$$

We have used the following convention for the index of coefficients:



We treat section '0' as if it were a standard quadratic factor even through some coefficients are zero. We execute the prewarping and bilinear-transform for each section. The program is written in IIR_C_LP.CPP, the result is shown in Figure 3.25. The frequency response is steeper and the phase and time delay are nonlinear. Since the coding for the log magnitude is long, a third header file FREQRESP.H will be created for the next demonstration.

3.3.5 IIR Filter, Elliptic Lowpass

The analog filter transfer function for this demonstration is an elliptic lowpass, N=5, passband ripple=0.5 dB. Su [10] (Table 2.52).

$$H(s) = \frac{0.01918}{(s+0.4254)}\frac{(s^2+2.4255)}{(s^2+0.1635s+1.0319)}\frac{(s^2+5.4376)}{(s^2+0.5702s+0.5761)}$$

The transfer function would meet the following specifications:

ripple ≤ 0.5 dB;
$A_a \geq 50.0$ dB;
$\Omega_p = 2.0$ rad/sec;
$\Omega_a \leq 3.0$ rad/sec.

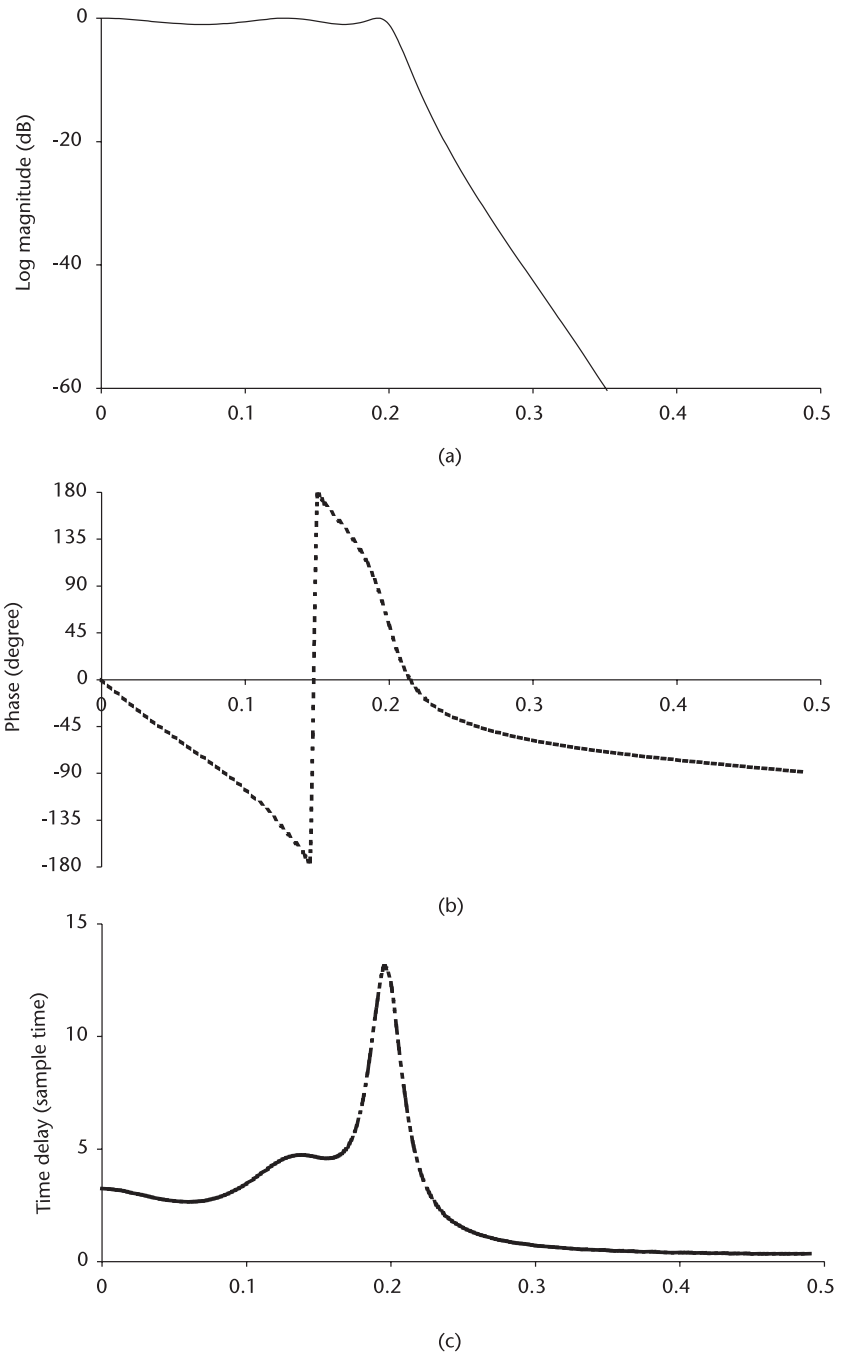


Figure 3.25 Chebyshev lowpass, N = 5, $\omega_c = 0.2$.

The analog transfer function is frequency-prewarped, followed by bilinear transform to obtain the coefficients a_j and b_i. The program is written in IIR_E_LP.CPP. This program has a third header file, FREQRESP.H in order to shorten the driver coding CPP. The result is shown in Figure 3.26. The attenuation slope is steepest and the phase and time-delay are nonlinear.

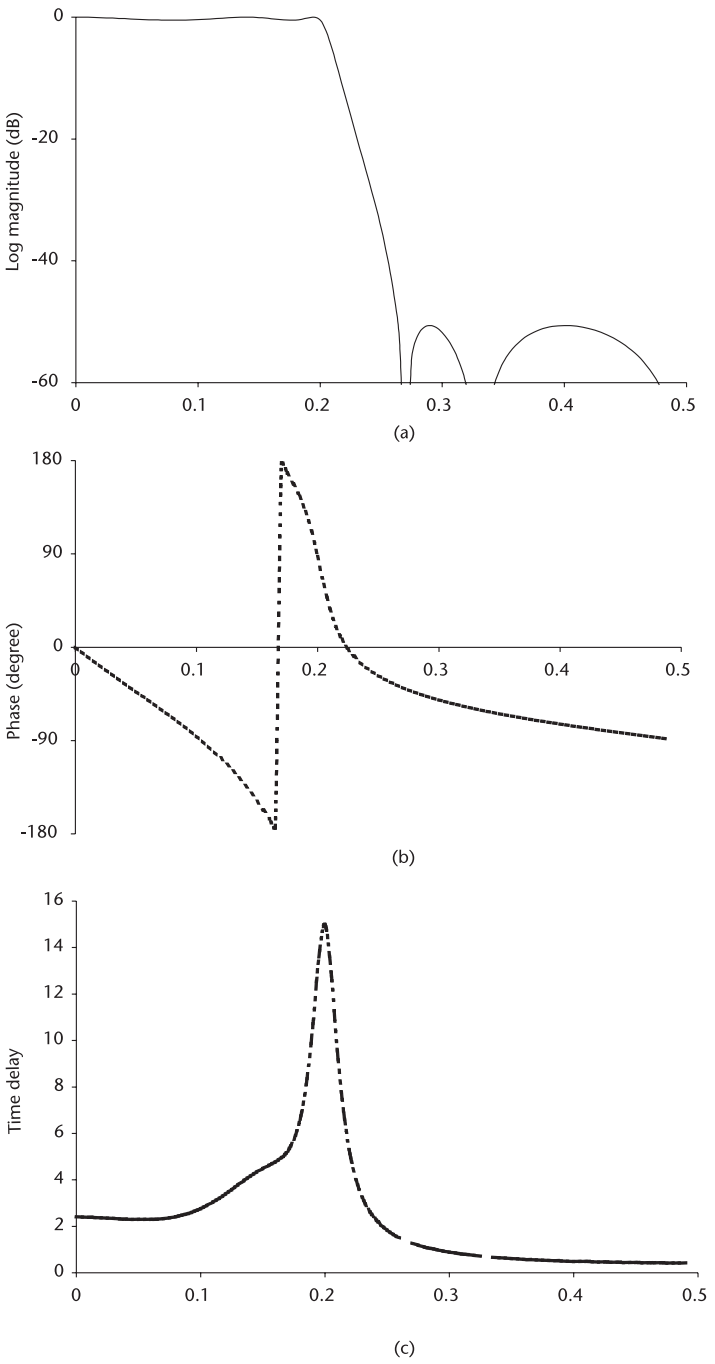


Figure 3.26 Elliptic lowpass, $N = 5$, ripple = 0.5dB.

3.3.6 IIR Filter, Elliptic Bandpass

We demonstrate how to design a bandpass filter from a lowpass prototype. We have listed the frequency band transformation previously; it is repeated below.

$$\text{lowpass-to-bandpass: } s \rightarrow \frac{s^2+\Omega_L\Omega_H}{s(\Omega_H-\Omega_L)} \quad \Omega_L, \Omega_H\text{: new low, high frequency}$$

The elliptic lowpass filter, N=5, ripple = 0.5 dB that we designed in the previous section will be transformed to a bandpass filter. The procedure is quite simple and mechanical:

1. The lowpass transfer function $H_{LP}(s)$ will be transformed to a bandpass transfer function $H_{BP}(s)$ by the transformation listed above.

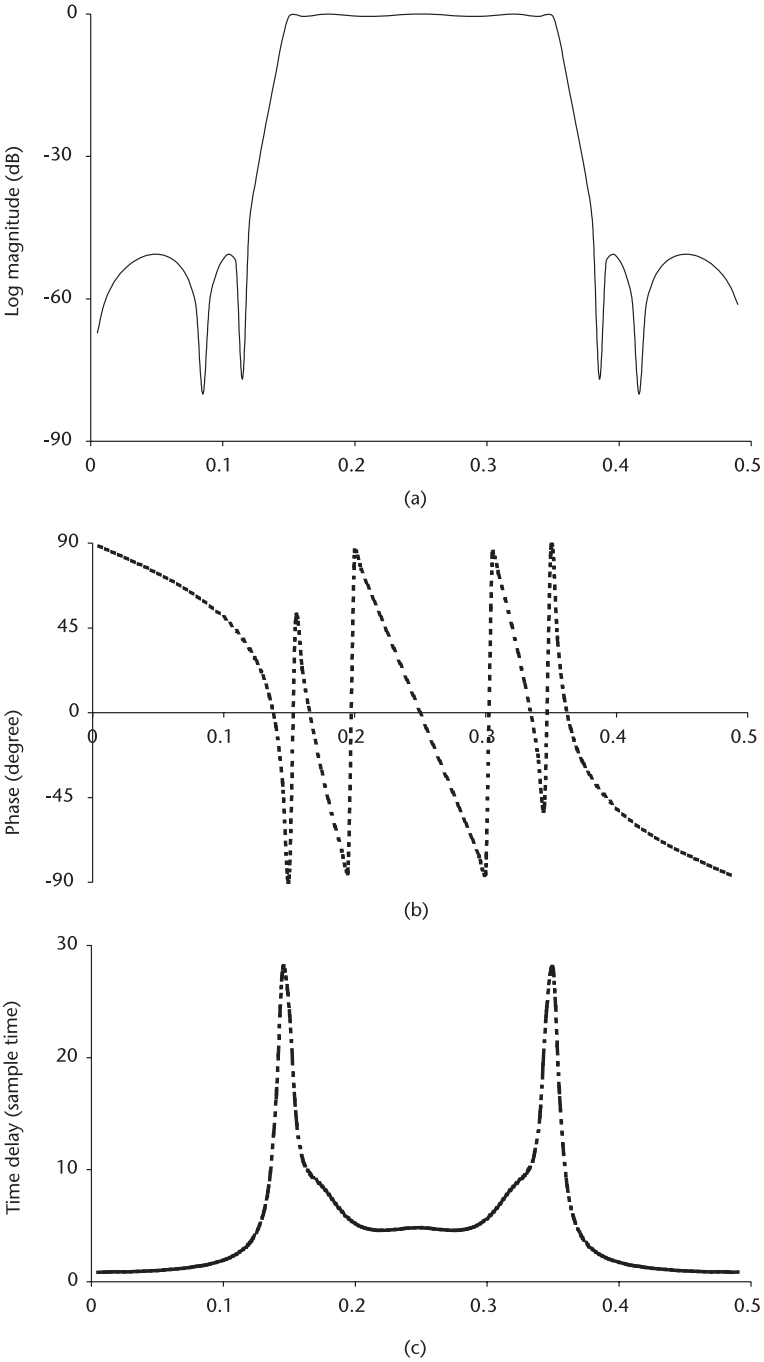


Figure 3.27 Elliptic bandpass, $N = 5$, $\omega_{lo} = 0.15$, $\omega_{hi} = 0.35$.

2. A frequency warping is followed by bilinear transform to obtain the coefficients a_j and b_i.
3. Compute the frequency response, phase, and time delay of the IIR bandpass.

A demonstration program is written in IIR_E_BP.CPP, and the result is shown in Figure 3.27. We note the phase response is nonlinear, as is the time-delay response.

3.3.7 Issue of Nonlinearity

We have observed that all IIR filters have a nonlinear response in phase and time delay. If any signal processing following the IIR filter requires the linearity we must have an equalizer immediately after the IIR filter. The equalizer must be an all-pass network; that is, the magnitude response must be a constant (flat) in the passband, and the time delay must be the inverse of the IIR filter that precedes it. See Figure 3.28.

A design procedure for an equalizer is reported in [12–14]. It is an interesting subject in the filter design regime. A question arises as to whether there is a way to design an IIR filter that meets both specified frequency response and phase linearity. There appear to be at least three methods to accomplish the task:

1. Direct synthesis method of Pade approximation;
2. Minimum-mean-square-error method;
3. Minimum p-error method.

The first method involves Pade's procedures to approximate the magnitude and phase response. The second method is called the steepest gradient algorithm, and the gradient of the function to be minimized has to be known. The design involves 4S+1 simultaneous equations to be solved where S is the number of sections of the factored quadratics. The third method is an extension of the second method in that a higher-order error criteria must be met. The "square" in the second method is raised to the pth power (p>2) for the third method. 4S+1 nonlinear equations must be solved through iterative procedures.

We omit the analysis and discussion on these design methods. Interested readers should consult with the references cited to explore the methods. The necessary mathematics are given by the authors. The reasons for the omission are discussed in the next section.

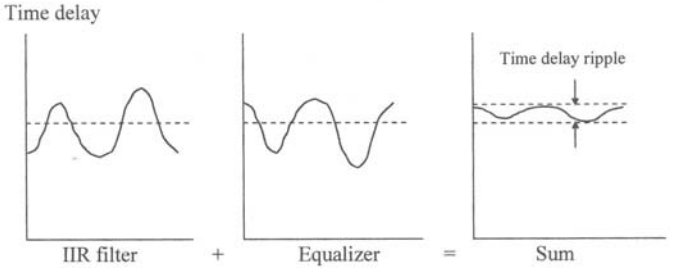


Figure 3.28 Role of time-delay equalizer.

3.3.8 Comparison Between FIR and IIR Filters

When a filter design assignment is given, the first question should be whether there is any follow-on signal processing after the filter, and if so, does it require phase linearity. When there is no requirement for phase linearity, the obvious choice is an IIR filter, preferably an elliptic filter.

On the other hand, if the follow-on signal processing demands phase linearity we must choose a FIR filter or IIR filter plus an all-pass equalizer. When the transition width is relatively narrow, (i.e., $N_{FIR} \geq 21$, or IIR elliptic $N_{IIR} \geq 5$) the order of an equalizer N_{EQ} required would be twice that of N_{IIR} if the delay ripple must be less than 1%. The complexity of an equalizer may be evaluated by the number of multiplications in the equalizer.

The number of multiplications for a FIR is given by (N+1)/2, and the corresponding number for an IIR filter is (4N+3)/2. If we assume for a moment that the order of an equalizer is twice that of the IIR filter (1% delay ripple), the number of multiplications for an equalizer is (4N+3). Then, for N_{FIR}=21, N_{IIR}=5 and N_{EQ}=10, eleven multiplications are required for a FIR, 38 multipli-cations for an IIR elliptic plus an equalizer, almost four times more multiplications. Suffice it to say that for most cases the FIR filter is the preferred one, the exception being when N_{IIR} is less than 3 and the delay ripple is relaxed to 10%.

3.3.9 Quantized Noise and the Dynamic Range of A/D Converter

Throughout this chapter the various coefficients are declared "float." The float declaration has a word length of four bytes, six significant digits after the decimal point. The smallest number ε (epsilon) summed to unity that is recognized greater than unity is 1.192092×10^{-7}.

Often in implementing a filter design we wish to reduce the word length to three bytes or two bytes for speed. A reduced word length will degrade the filter performance. The expected filter performance cannot be realized with quantized coefficients in reduced word length, and the extent of degradation can never be predicted, so several trial-error programs would be required. The coefficients of a lowpass FIR filter with a Hamming window is quantized to 8 bits, and the frequency response is shown in Figure 3.29 for comparison.

The quantization of an analog signal by an A/D converter introduces quantization noise. We take an example of the A/D converter shown in Figure 3.30. The discussion on the quantization noise is based on expedient assumptions about the probability and statistics of sampled noise; that is, we assume the following:

1. The sequence of error e[n] is the sample sequence of stationary random process;
2. The sequence of error e[n] is uncorrelated from one sample to another, a white noise process;
3. The sequence of error e[n] is uncorrelated with the unquantized analog signal S(t).

There are cases where the assumptions made cannot be justified; however, in many practical situations when the signal fluctuates rapidly, as in speech, music,

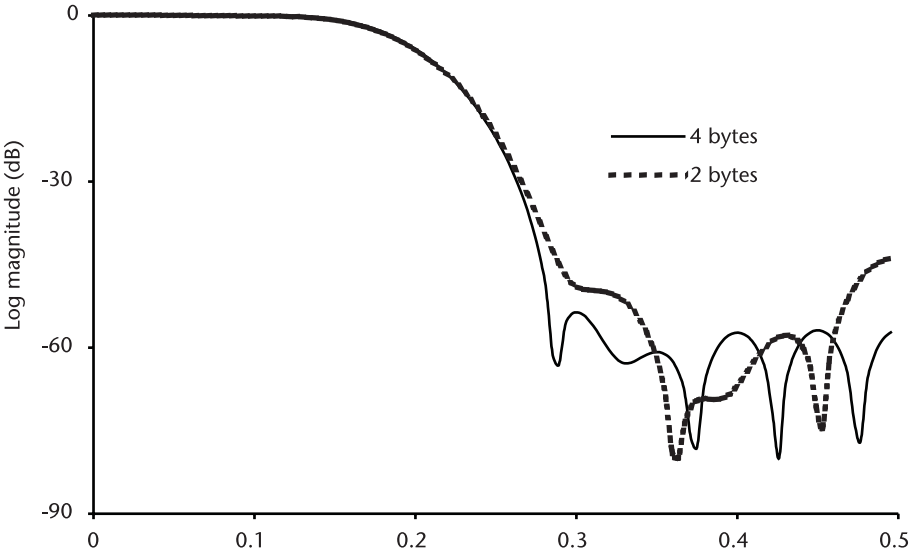


Figure 3.29 Performance degradation due to reduced word-length quantization of coefficients.

or radar return, the assumptions are realistic and acceptable. Figure 3.31 shows a small segment of a music note, a sum of four trigonometric functions with different periods and amplitudes.

Assuming the quantized errors e[n] are uniformly distributed in the range

$$\frac{-\Delta}{2} \le e[n] \le \frac{\Delta}{2}$$

the probability density function of e[n] can be visualized.

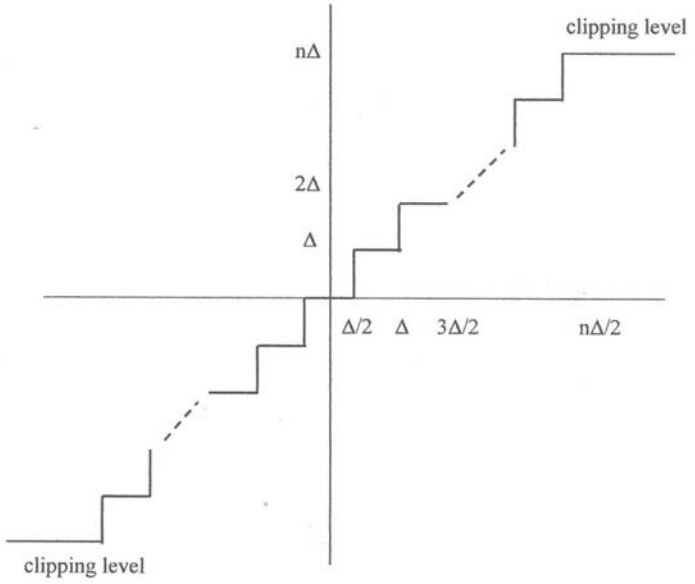


Figure 3.30 Quantization by A/D converter.

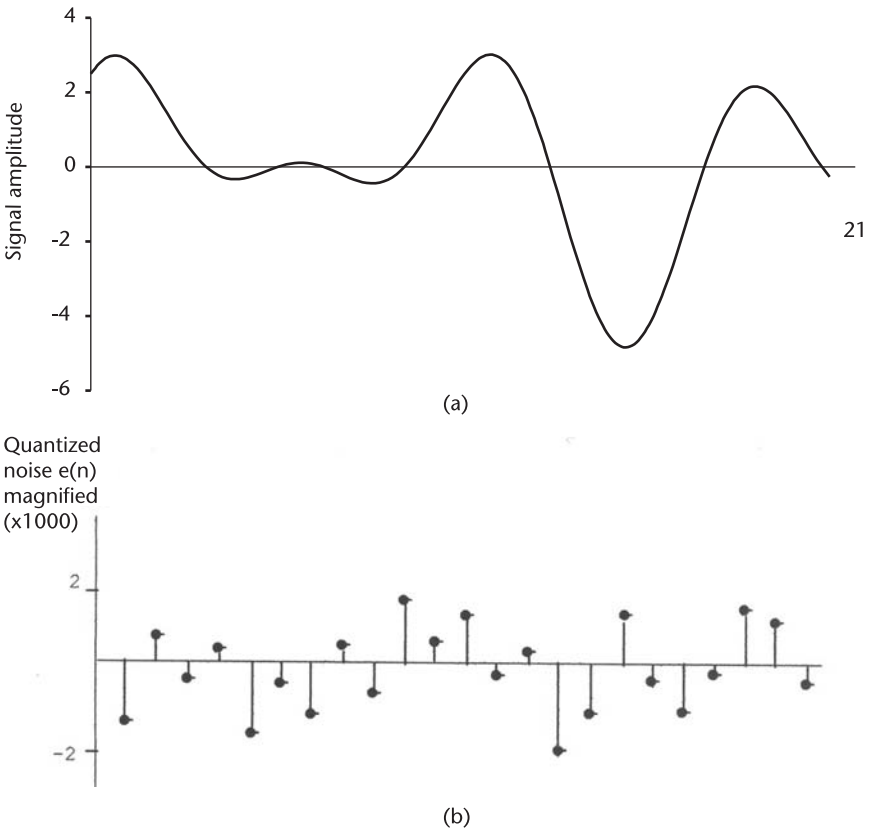
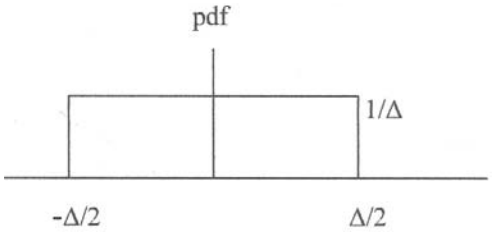


Figure 3.31 Analog signal s(t) and quantized error e[n].



The mean and variance of quantized error e[n] are,

$$\text{mean} = 0.0$$

$$\sigma_n^2 = \frac{\Delta^2}{12} \Big/ \frac{1}{12}(2^{-2b})$$

In other words the quantization noise power is inversely proportional to the square of the number of quantization bits; the larger the number of bits the smaller the noise power. In signal processing we are concerned with signal-to-noise ratio in power, that is, how clean the quantized signal is relative to how dirty the quantized noise.

$$\frac{\sigma_s^2}{\sigma_n^2} = \frac{\sigma_s^2}{\frac{1}{12}(2^{-2b})} = (12)2^{2b}\sigma_s^2$$

When the ratio is expressed as a logarithm, SNR in power is

$$\mathrm{SNR}_{\mathrm{power}} = 10\ \log_{10}\left(\frac{\sigma_s^2}{\sigma_n^2}\right) = 10.79 + 6.02\mathrm{b} + 10\ \log_{10}(\sigma_s^2)$$

How to determine the variance of signal σ_s^2 ? Examine Figure 3.30 where the quantization is shown together with the clipping levels. To eliminate (or to avoid) the clipping distortion we must reduce the maximum amplitude of signal s(t) or increase the dynamic range of A/D converter.

Many experiments have proven that the present amplitude of a signal is completely independent of the past history of magnitudes except one or two immediate past ones. The magnitude fluctuation is a Gaussian-Markov process. The probability density function of a Gussian Markov process is characterized by an exponential or Gaussian look-alike as shown in Figure 3.32. The probability of signal amplitude is peaked at zero and rapidly falls off as the signal amplitude increases.

If we set $\mathrm{A}= \sigma_s^2/4$, the probability of clipping is very, very low. Then we rewrite the SNR as

$$\begin{aligned}\mathrm{SNR}_{\mathrm{power}} &= 10\ \log_{10}\left(\frac{\mathrm{A}^2\sigma_s^2}{\sigma_n^2}\right)\\ &= 10.79 + 6.02\mathrm{b} + 10\ \log_{10}(\sigma_s^2) + 20\ \log_{10}(\mathrm{A})\\ &= 6.02\mathrm{b} - 1.25\ \mathrm{dB} \qquad (\sigma_s^2 = 1, \mathrm{A} = 1/4)\end{aligned}$$

The last expression indicates that the SNR in power increases by 6.02 dB with each bit added to A/D converter. Thus, SNR>=60 dB requires 11 bits, SNR>=80 dB requires 12 bits.

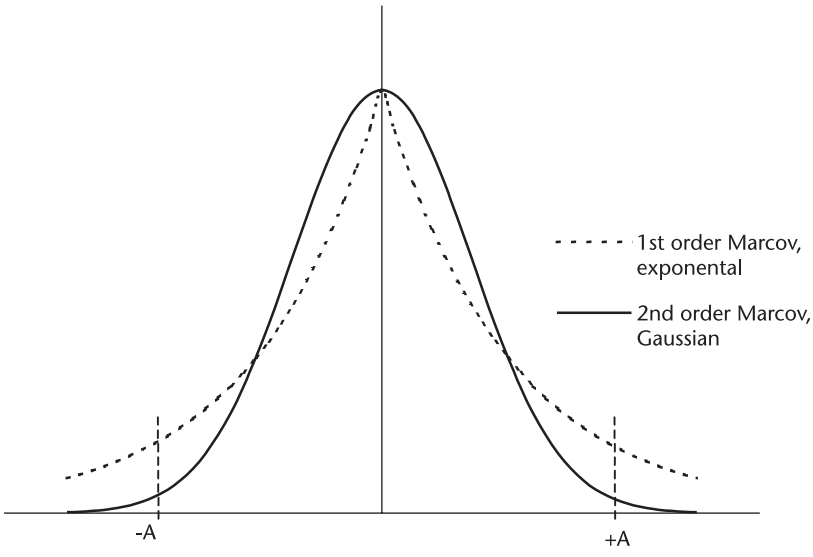


Figure 3.32 Gauss-Markov process.

List of Programs

Program	*Features*
(1) IMPULSE.CPP	FIR impulse response of lowpass, highpass, bandpass, and bandstop
(2) FIR_LP.CPP	FIR lowpass prototype and choice of window function
(3) FIR_HP.CPP	FIR highpass prototype and choice of window function
(4) FIR_BP.CPP	FIR bandpass prototype and choice of window function
(5) FIR_BS.CPP	FIR bandstop prototype and choice of window function
(6) KAISER.CPP	Kaiser filter design, lowpass, highpass, bandpass and bandstop by switch command
(7) IIR_B_LP.CPP	IIR Butterworth lowpass
(8) IIR_C_LP.CPP	IIR Chebyshev lowpass
(9) IIR_E_LP.CPP	IIR elliptic lowpass
(10) IIR_E_BP.CPP	IIR elliptic bandpass
(11) PRE_WARP.H	Analog frequency prewarping operations
(12) BILINEAR.H	Bilinear transform from analog to digital
(13) FILT_SEL.H	Computes Kaiser α and the order N, and select lowpass, highpass, bandpass, or bandstop
(14) FREQRESP.H	Frequency response of filters, log magnitude
(15) FFTDIF.H	Fast Fourier transform, decimation-in-frequency (replica in Chapter 4)

References

[1] Rabiner, L.R., and B. Gold, *Theory and Application of Digital Signal Processing*, Englewood Cliffs, N.J.: Prentice-Hall, 1975.

[2] Oppenheim, A.V., and R.W. Schafer, *Digital Signal Processing*, Englewood Cliffs, N.J.: Prentice-Hall, 1975.

[3] Harris, F. J., "On the Use of Windows for Harmonic Analysis with Discrete Fourier Transform," *Proc. IEEE*, Vol.16, Jan. 1978, pp.51–83.

[4] Kaiser, J. F., "Nonrecursive Digital Filter Design using I_0-sinh Window Function," *Proc. IEEE, Intern'l Synposium*, Circuit Theory, 1974.

[5] Hamming, R.W., *Digital Filters*, Englewood Cliffs, N.J.: Prentice-Hall, 1977.

[6] Weinburg, L., *Network Analysis and Synthesis,* New York, N.Y.: McGraw-Hill, 1962.

[7] Guillemin, E. A., *Synthesis of Passive Network*, New York, N.Y.: John Wiley & Sons, 1957.

[8] Storeer, J. E., *Passive Network Synthesis,* New York, N.Y.: McGraw-Hill, 1957.

[9] Antoniou, A., *Digital Filters Analysis and Design*, New York, N.Y.: McGraw-Hill, 1979.

[10] Su, K. L., *Handbook of Table for Elliptic Function Filter,* Norwell, MA: Kluwer Academic Publishers, 1990.

[11] Zverevm, A. I., *Handbook of Filter Synthesis*, New York, N.Y.: John Wiley & Sons, 1967.

[12] Thiren J. P., "Equal-Ripple Delay Recursive Digital Filters," *IEEE Trans. Circuit Theory*, Vol. CT-18, Nov. 1971.

[13] Fetlweis, A., "A Simple Design of Maximally Flat Delay Digital Filters," *IEEE Trans. Audio & Electroacoustics*, AU-20, Jan. 1972.

[14] Deczky, A. G., "Synthesis of Recrusive Digital Filters Using Maximum p-Error Criterion," *IEEE Trans. Audio & Electroacoustics*, AU-20, Oct. 1972.

CHAPTER 4

Fast Fourier Transform (FFT) and IFFT

4.1 Introduction

The fast Fourier transform, forward and inverse, has found many applications in signal processing. Although the theory of fast Fourier transforms is well-known, numerous commercially available software packages have caused some confusion for beginners; some of them are written in radix 2, 4, or 8; in mixed radix 8 (4x2); decimation-in-time; or decimation-in-frequency scheme. Some codings are dedicated to real input/output only, others for complex input/output only, or sinusoidal input/output only, for example.

This chapter describes the basic building blocks of FFT and IFFT in radix 2 exclusively. The decimation-in-time and decimation-in-frequency algorithms will be explained in detail. All other algorithms readers may encounter are the variants of radix 2 of FFT and IFFT in order to shorten the computation time (to lessen the computation load) or minimize the memory size for certain specific applications.

The Fourier transform of a finite sequence is defined as

$$X(k) = \sum_{n=0}^{N-1} x[n]\, e^{-j(2\pi/N)kn} \quad \text{(indexed in n)} \tag{4.1}$$

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{+j(2\pi/N)nk} \quad \text{(indexed in k)} \tag{4.2}$$

where

X[k]: Frequency sampled data;
x[n]: Time sampled data;
N: Total number of samples;
n: Time index, n=0, 1, 2, 3, ... , N-1;
k: Freq index, k=0, 1, 2, 3, ... , N-1.

Using a shorthand notation,

$$e^{-j(2\pi/N)} = W_N$$

Equation (4.1) and (4.2) can be written as,

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk} \quad \text{(index in n)} \tag{4.3}$$

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-nk} \quad \text{(index in k)} \tag{4.4}$$

When we examine (4.3) and (4.4), there is quite a large number of duplicated multiplications, and this adds operations. The FFT is an algorithm that eliminates the duplications by recognizing which indices "n" and "k" are repeated by what sequences. There are a few excellent textbooks that describe a rather simple concept but intricate steps to eliminate these duplicated operations, Oppenheim and Schafer[1], Rabiner and Gold[2], and Brigham[3].

4.2 Fast Fourier Transform, Decimation-in-Time and Decimation-in-Frequency

The signal flow diagram shown in Figure 4.1 describes how (4.3) is implemented without any duplication; eight pieces of time-sampled data, x[n], are to be transformed to eight pieces of frequency data, X[k], without any duplication. We note the eight pieces of time-sampled data are split into even and odd sequences, and they are not in the natural increasing order. This scrambled input sequence is to eliminate the duplications. The scrambled input sequence is called "bit-reversed," which will be discussed shortly. The basic operation in Figure 4.2 is called the "butterfly" operation.

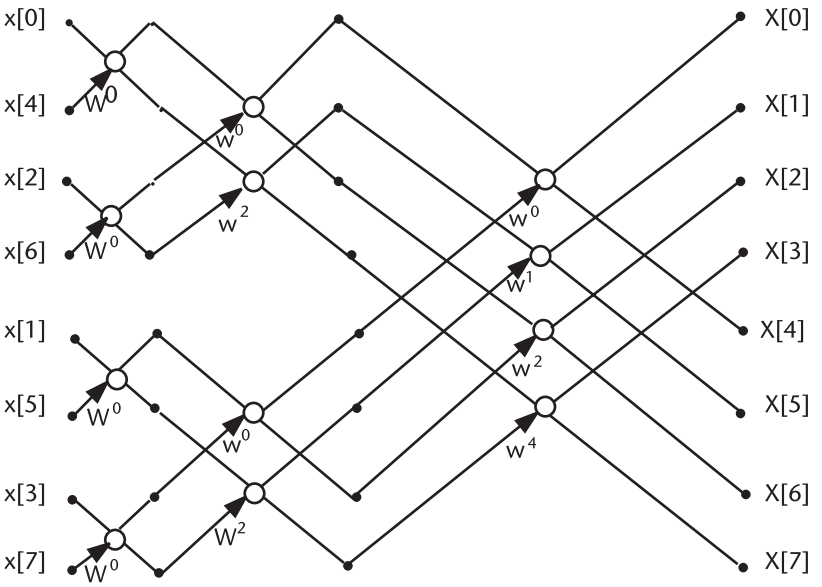


Figure 4.1 FFT, decimation-in-time, $N = 8$.

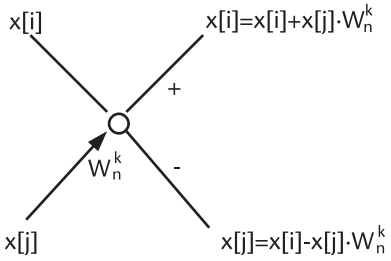


Figure 4.2 Butterfly operation, decimation-in-time.

The butterfly operation for decimation-in-time is coded as,

```
for(i=0; i<N; i++)
 {
  T=x[j]*W          //T: temporary memory
    x[j]=x[i]-T
         x[i]=x[i]+T
 }
```

The results, x[i] and x[j] on the right in Figure 4.2, will replace the input data. We call this an "in place" operation; that is, we do not need separate memories to store the results as we progress from left to right in Figure 4.1.

If we prefer the time-sampled input data sequence to be in natural increaseing order, the butterfly diagram shown in Figure 4.3 results. We note that the output frequency sequence is all scrambled. This is an FFT in decimation-in-frequency. The basic butterfly operation in decimation-in-frequency is shown in Figure 4.4, and coded as follows.

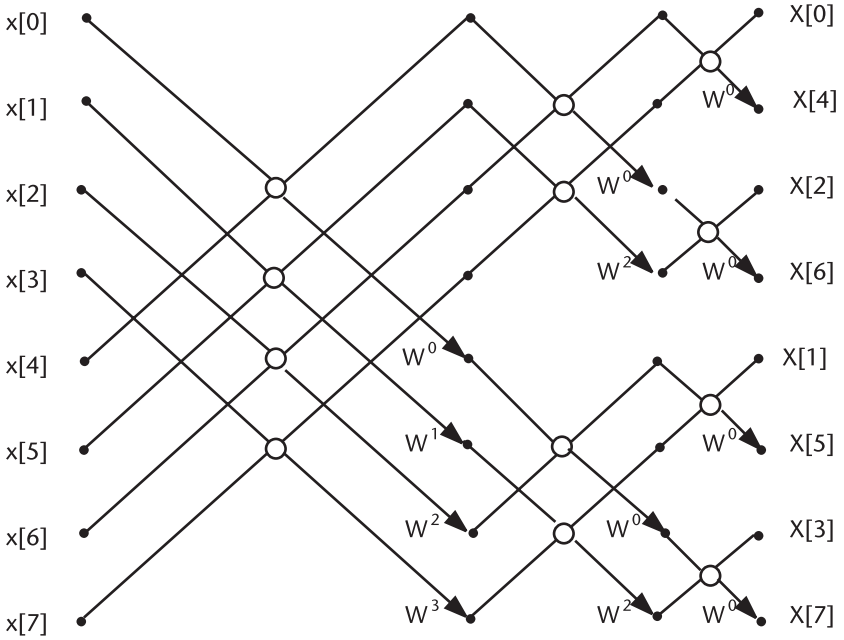


Figure 4.3 FFT, decimation-in-frequency, $N = 8$.

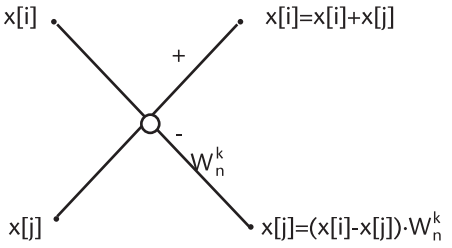


Figure 4.4 Butterfly operation, decimation-in-frequency.

```
for(i=0,i<N,i+)
  for(i=0, i<N; i++)
   {
     T=x[i]-x[j];            //T: temporary memory
            x[i]=x[j]+x[i]
            x[j]=T*W
   }
```

When we examine Figure 4.1, FFT with decimation-in-time, the output frequency spectral terms are in natural increasing order but the input time data must

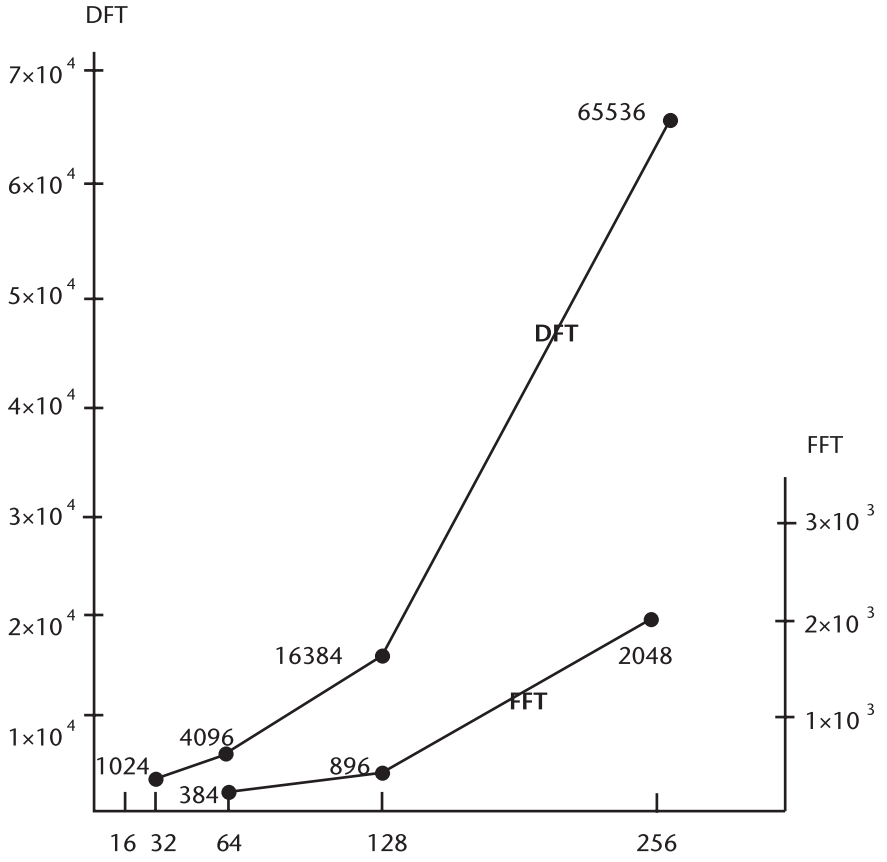


Figure 4.5 Comparison between DFT and FFT.

be bit-reversed. On the other hand, for FFT by decimation-in-frequency, in Figure 4.3, the input time data are in natural increasing order but the output frequency spectral terms are in bit-reversed. This is the consequence of eliminating the duplicated operations.

How many duplicated multiplication/add operations are eliminated by decimation-in-time or decimation-in-frequency is computed in DFTvsFFT. CPP, and the result is shown in Figure 4.5. When the input data is moderately large the saving in computation is substantial. This is the power of the fast Fourier transform algorithm. The saving enables us to perform the Fourier transform in real time, which was heretofore impractical.

To have the input data in the natural increasing order and the output data in increasing natural order, we need the bit-reversal operation. The principle idea of the bit-reversal is shown below.

Input time data	*Binary*	*Binary reversed*	*Output freq data*
x[0]	0 0 0	0 0 0	X[0]
x[1]	0 0 1	1 0 0	X[4]
x[2]	0 1 0	0 1 0	X[2]
x[3]	0 1 1	1 1 0	X[6]
x[4]	1 0 0	0 0 1	X[1]
x[5]	1 0 1	1 0 1	X[5]
x[6]	1 1 0	0 1 1	X[3]
x[7]	1 1 1	1 1 1	X[7]

The index of time data, first column, in decimal is converted to binary, second column. The binary bits are reversed in order in the third column. The reversed binary is converted back to decimal in the fourth column, the output of frequency spectra sequence. An example of bit-reversal for N = 16 is shown.

Index of input data	*Index of bit-reversed data*
0	0
1	8
2	4
3	12
4	2
5	10
6	6
7	14
8	1
9	9
10	5
11	13
12	3
13	11
14	7
15	15

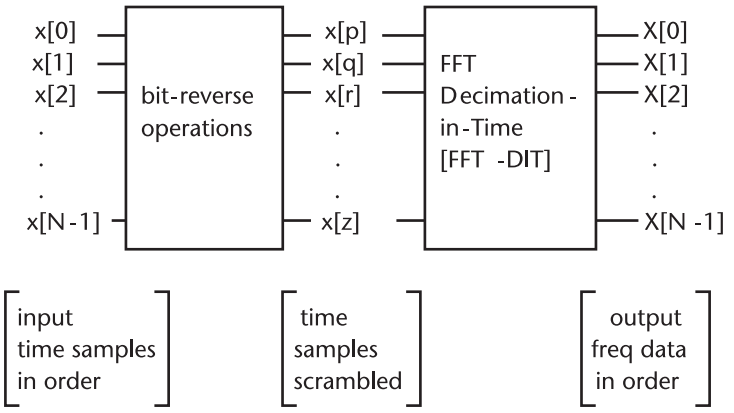


Figure 4.6 FFT, decimation-in-time.

The bit-reversal algorithm is programmed in BIT_REV.CPP. The number of input data N must be a power of 2 (radix-2 requirement). Figures 4.6 and 4.7 show decimation-in-time and the decimation-in-frequency with bit-reversal operation in proper sequence.

Nonexecutable fragmented FFT_DIT code is shown below. The fragment executes the butterfly flow diagram of Figure 4.1.

```
for(L=0; L<m; L++)                    //m: number of butterfly
                                      //stages from left to right,
   {                                  //radix 2.
      LE=pow(2, L+1);
      LE1=LE/2.0;
      W=complex(1.0, 0.0);

   U=complex(cos(PI/LE1)              //U=e^-j(·) = cos(·)-jsin(·)
           -(sin(PI/LE1));            //PI= π
```

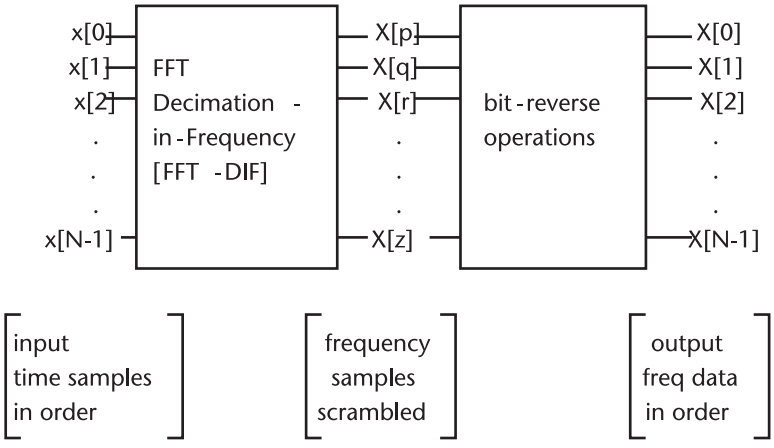


Figure 4.7 FFT, decimation-in-frequency.

```
for(j=0; j<LE1; j++)
 {
   for(i=0; i<N; i++)      //basic butterfly operation
     {
       ip=i+LE1;           //index i primed,  i'
       T=x[ip]*W;          //multiply and add

         x[ip]=x[i]-T;
               x[i]=x[i]+T;
     }
   W=W*U;
 }
```

An identical code for the FFT_DIF algorithm can be written down except for a small alteration in the basic butterfly operation as shown in Figure 4.3.

FFT with decimation-in-time and decimation-in-frequency are programmed in FFT_DIT.CPP and FFT_DIF.CPP with a header file that performs the bit-reversal operation in proper order. Both programs require that the number of input data must be a power of 2. In the literature this is called the "radix-2" scheme.

$$N = 2^m, \quad m = 1, 2, 3, \ldots$$

Readers who are interested in the radix-4, radix-8, or mixed radix scheme should consult the references cited. When the number of pieces of input data does not meet the radix-2 requirement we shall add zeros to the input sequence to satisfy the requirement.

4.3 Demonstration of FFT_DIT and FFT_DIF

For a demonstration we choose a trigonometric function as an input.

$$f(t) = \cos(2\pi f_0 t) = \cos\left(\frac{2n\pi}{N}\right)$$

where

N=16;
n=0, 1, 2, 3, . . . 15.

From mathematic textbooks on the Fourier transform, the transform pair is given by

$$f(t) = \cos(2\pi f_0 t) = 1/2(e^{j2\pi f_0 t} + e^{-j2\pi f_0 t})$$

$$F(f) = \int_{-\infty}^{\infty} \cos(2\pi f_0 t)\, e^{-j2\pi ft}\, dt$$

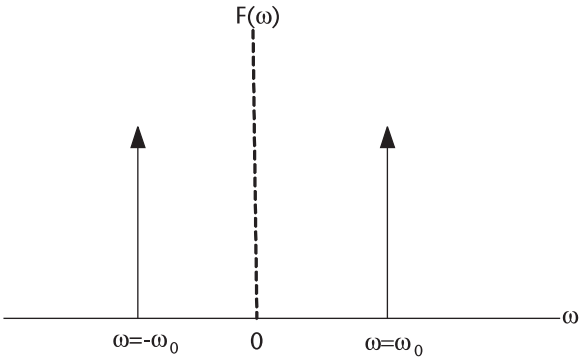


Figure 4.8 Spectrum of cos(ωt).

$$= 1/2 \int_{-\infty}^{\infty} [e^{j2\pi(f_0-f)t} + e^{-j2\pi(f_0-f)t}]\, dt$$

$$= 1/2\ \delta(f_0-f) + 1/2\ \delta(f_0+f)$$

Our demonstration program, FFT_DIT.CPP, produced the frequency data, pictorially presented in Figures 4.8 and 4.9.

When we cut out the right half of the spectrum, index 8 to 16, and slide to the negative frequency axis to make index 16 coincide with index 0, we see the spectra illustrated in Figure 4.10.

The amplitude of 1/2 in the analytic result and the amplitude 8 of the FFTed result is a scaling problem we have ignored in the butterfly operation. Equation (4.2) indicates the scaling by 1/16.

As a second demonstration we choose the sine function. The program is written in FFT_DIF.CPP.

$$f(t) = \sin(2\pi f_0 t) = \sin\left(\frac{2n\pi}{N}\right)$$

where

N=16;
n=0, 1, 2, 3, . . . 15.

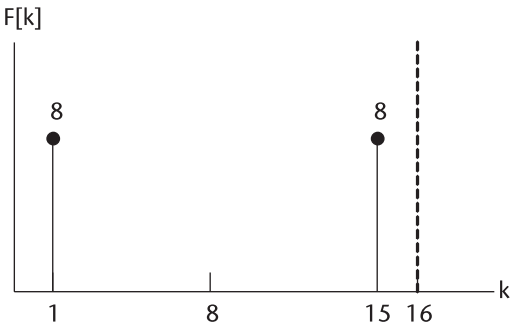


Figure 4.9 Spectra of FFTed cosine function.

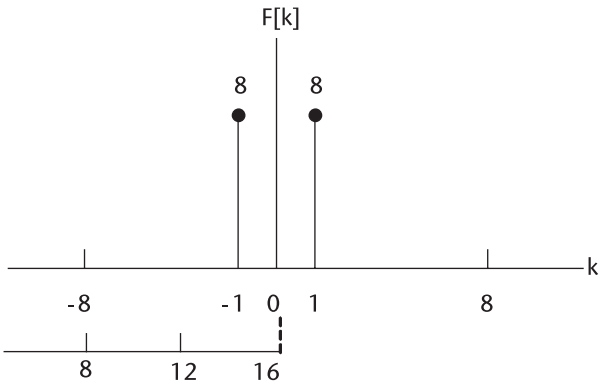


Figure 4.10 Interpretation of Figure 4.9.

The Fourier transform pair is

$$f(t) = \sin(2\pi f_0 t) = \frac{1}{j2}[e^{j2\pi(f_0-f)t} - e^{-j2\pi(f_0-f)t}]$$

$$F(f) = \frac{1}{j2}\int_{-\infty}^{\infty}[e^{j2\pi(f_0-f)t} - e^{-j2\pi(f_0-f)t}]e^{-j2\pi ft}dt$$

$$= \frac{1}{j2}[\delta(f_0+f) - \delta(f_0-f)]$$

See Figure 4.11.

Our demonstration program FFT_DIF.CPP produced the frequency data that is pictorially presented in Figure 4.12.

The right half of the spectrum is shifted to the negative frequency axis as we have done before, shown in Figure 4.13. We should apply the scaling mentioned previously. Note that the vertical axis is imaginary.

From Figures 4.10 and 4.13 we see that cosine and sine functions maintain a quadrature angle relationship between them.

$$e^{jx} = \cos x + j \sin x, \quad e^{j(\pi/2)} = 90°$$

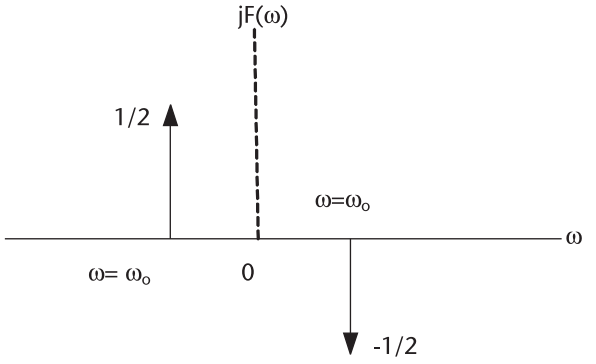


Figure 4.11 Spectra of $\sin(2\pi f_0 t)$.

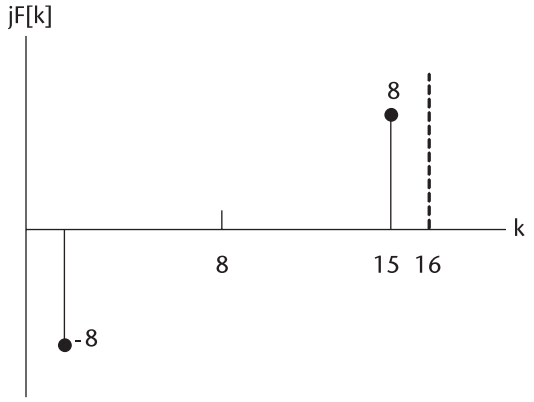


Figure 4.12 Spectra of FFTed sine function.

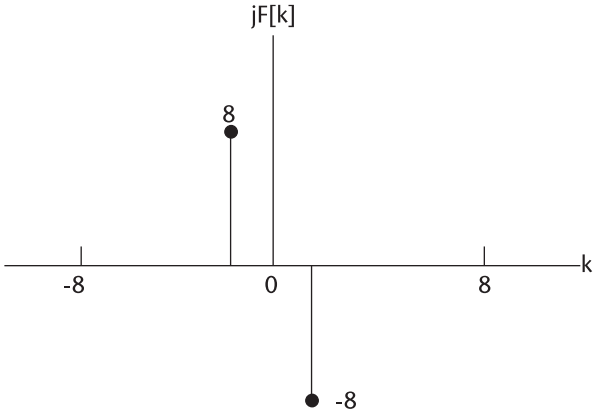


Figure 4.13 Interpretation of Figure 4.12.

Sometimes, the cosine function is called the "in-phase" function, the sine function the "quadrature-phase" function.

4.4 Spectral Leakage and Window Function

We note that in demonstration programs FFT_DIT.CPP and FFT_DIF. CPP that the sampling time interval is an exact submultiple of the duration of the function, and the number of samples N is matched to the duration of the time function. The results of the FFTed spectra agreed with analytic results.

In practical situations we seldom meet the exact multiplicity and the complete duration of time function. More often than not we would never know in advance the exact period nor the duration of the function. That is why we compute the spectrum through FFT to begin with.

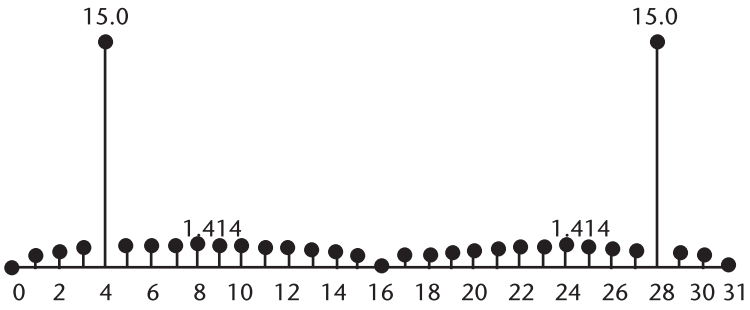


Figure 4.14 Spectrum leakage, LEAKAGE1.CPP.

When the number of samples does not cover the entire time function and the sampling interval is not an exact submultiple of the time function, we observe that the FFTed spectrum spills over to the neighboring bins. When the duration of time function is not an exact integer multiples of sampling interval, a spectrum spillage will result. The former case is programmed in LEAKAGE1.CPP and the latter in LEAKAGE2.CPP. The results of two programs are shown in Figures 4.14 and 4.15.

The spectrum leakage can be controlled but not completely eliminated by applying a window function. We investigate von Hann, Hamming, Blackman, and Kaiser windows in FIR filter design in Chapter 3; however, these windows are not suitable for leakage control. These windows associated with filter designs have nonzero end values. Here we must have zero at the end points.

The input time function used in LEAKAGE1.CPP is multiplied by a Welch window and FFTed in LEAKAGE3.CPP. The result is shown in Figure 4.16. We note a substantial reduction in spectral leakage. The best windows for FIR filters are not necessarily the best for leakage control. For leakage control a window should have zeros at the end points and the largest area under the window envelope. A trapezoid with a rounded shoulder may be a good candidate. We avoid a lengthy discussion on the choice of window; all windows serve the purpose in contrast to no-windowed processing, more or less.

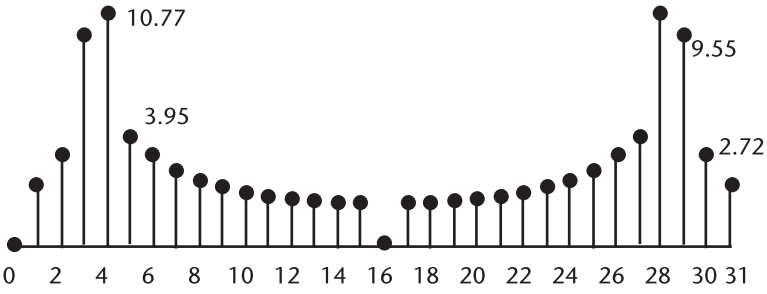


Figure 4.15 Spectrum leakage, LEAKAGE2.CPP.

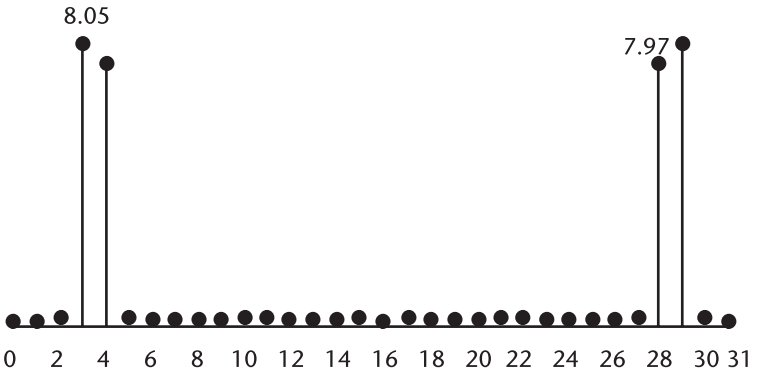


Figure 4.16 Leakage reduction by window, LEAKAGE3.CPP.

4.5 Inverse Fast Fourier Transform, Decimation-in-Time, and Decimation-in-Frequency

The inverse Fourier transform of a finite duration sequence is defined by (4.5), which is a repeat of (4.1).

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j(2\pi/N)\,kn} \qquad \text{(indexed in n)} \tag{4.5}$$

When $e^{-j(2\pi/N)} = W_N$ is substituted, we have

$$X(k) = \sum_{n=0}^{N-1} x[n]\, W_N^{nk} \tag{4.6}$$

The butterfly flow diagrams for decimation-in-time and decimation-in-frequency are identical to Figures 4.1 and 4.3 except the sign of (nk) on W. As in the forward

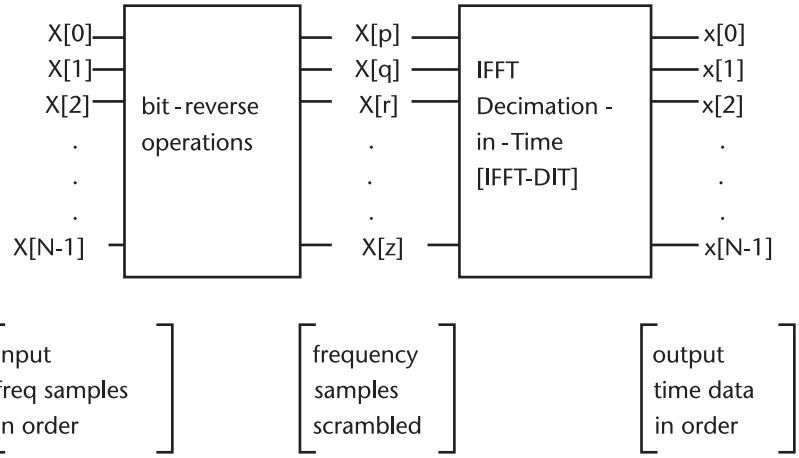


Figure 4.17 IFFT, decimation-in-time.

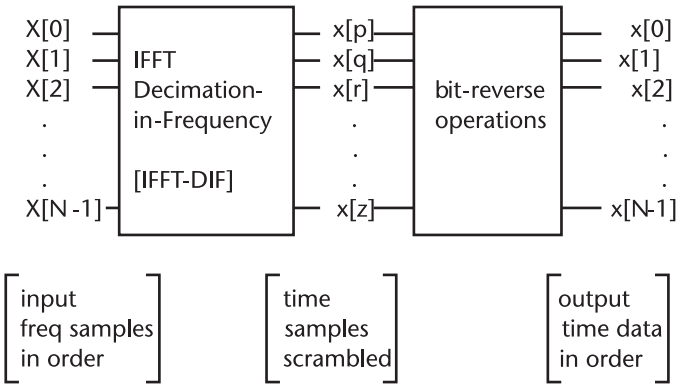


Figure 4.18 IFFT, decimation-in-frequency.

transform, the inverse FFT with decimation-in-time requires a bit-reversal first, followed by butterfly operations. For inverse FFT with decimation-in-frequency the order of bit-reversal and butterfly operations is reversed. We do not draw the butterfly for IFFT; we just change the sign on the power of W and replace the time input data x[n] by frequency data X[k] in Figures 4.1 and 4.3. See Figure 4.17.

Two demonstration programs for the inverse transform are written in IFFT_ DIT. CPP and IFFT_DIF.CPP. The input data are those we have obtained in the forward transform by FFT_DIT.CPP and FFT_DIF.CPP. The correct answer should be the time-sampled sequence we had as an input in the forward transform. See Figure 4.18.

4.6 Applications of FFT and IFFT

A very large number of application reports poured out of academia and industry analogous to opened flood gates, once the power of the fast Fourier transform algorithm was understood. Hundreds, if not thousands, of application notes covering almost all fields of science and engineering have been published.

We quote [4], ". . . the best mouse trap stand aside: if you speed up any non-trivial algorithm by a factor of a million or more, the world will beat the path in finding useful applications for it."

We will classify these application reports under major topics:

1. Spectral computation and filtering in frequency domain;
2. Convolution and deconvolution;
3. Autocorrelation and cross-correlation;
4. Speech analysis, voice synthesis and recognition;
5. Two dimensional FFT_IFFT for image processing;
6. Others.

Filtering in the frequency domain is an obvious application of the FFT-IFFT algorithm (see Figure 4.19). A noise-contaminated signal is forward-transformed, multiplied by a suitable filter function (window), and inverse FFTed to recover the signal. The signal-to-interference + noise ratio will be greatly improved.

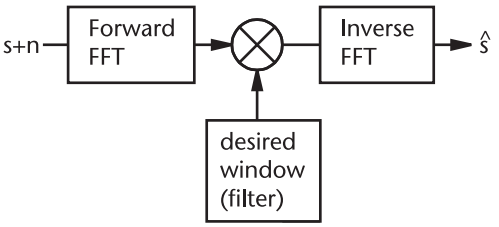


Figure 4.19 Filtering in frequency domain, FFT+Window+IFFT.

We have discussed the transition width in FIR and IIR filter design. Here we can implement the transition width in a single sampling interval, the steepest possible slope.

The convolution of two finite duration signal sequences is given by,

$$\text{conv(g,h)} = \int_{-t}^{t} g(\tau)h(t-\tau)d\tau$$

When we compute the forward Fourier transforms of g(t) and h(t) the convolution is obtained by the product of G(f) and H(f). Thus,

$$\text{conv(g,h)} = \int_{-t}^{t} g(\tau)\ h(t-\tau)d\tau = G(f) \cdot H(f)$$

The correlation of two functions g(t) and h(t) can be obtained similarly as the product of two time functions Fourier transformed: H*(f) is the conjugate of H(f).

$$\text{corr(g,h)} = \int_{-t}^{t} g(\tau + t)\ h(\tau)d\tau = G(f) \cdot H^*(f)$$

When two functions are identical, g(t) = h(t), we obtain the autocorrelation function.

$$\text{auto-corr(g,h)} = \int_{-t}^{t} g(\tau + t)\ g(\tau)\ d\tau = |G(f)|^2$$

Parceval's theorem states that the total power of a signal can be computed in the time domain or in frequency domain:

$$\int_{-t}^{t} |g(t)|^2 dt = \int_{-f}^{f} |G(f)|^2\, df$$

We ask readers to forgive the author for not covering the next two topics: voice synthesis and recognition and image processing, for these topics deserve separate books.

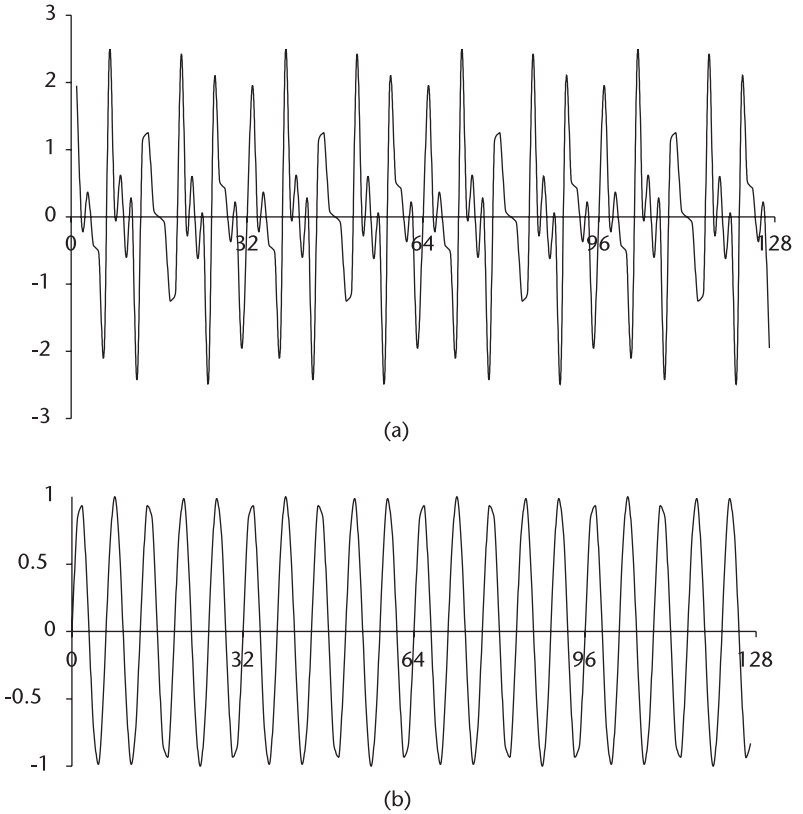


Figure 4.20 (a) Input signal, mixture of 20 Hz, 40 Hz, and 60 Hz. (b) Result of lowpass filter.

4.6.1 Filtering in the Frequency Domain

Filtering in the frequency domain is demonstrated in FFT_FILT.CPP. A sinusoidal signal with a mixture of 20 Hz, 40 Hz and 60 Hz is the input to the FFT-filter-IFFT

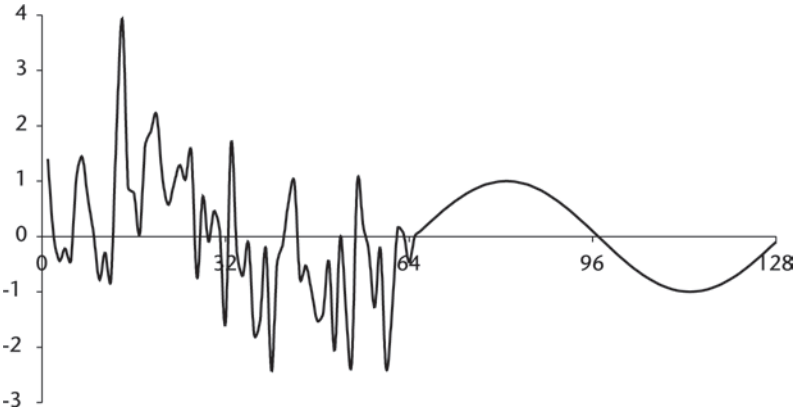


Figure 4.21(a) Input signal: First half is contaminated by noise; second half is noise-free.

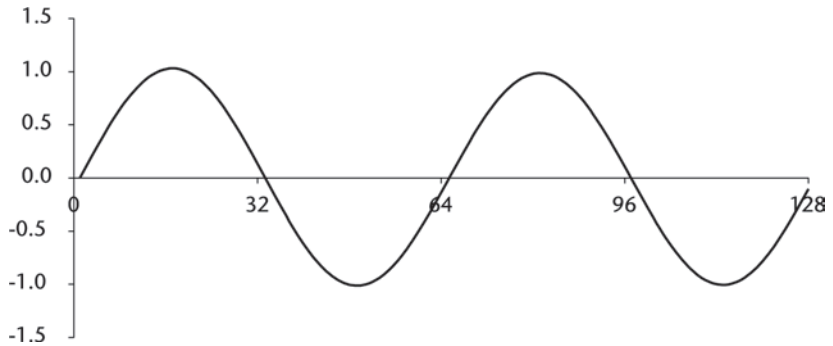


Figure 4.21(b) Recovered signal.

algorithm. The input signal is shown in Figure 4.20(a), and the output of a lowpass filter with a cutoff frequency at 21 Hz is shown in Figure 4.20(b).

The output waveform is not pure 20 Hz. Would you recall the subjects of leakage we have discussed earlier? Readers who are interested should compute the magnitude of the leakage components.

4.6.2 Detection of Signal Buried in Noise

A second demonstration program is written to recover a signal buried under white Gaussian noise, SIGNOISE.CPP. The signal-to-noise ratio of the input signal is –3 dB, shown in Figure 4.21(a) in the first half period.

4.6.3 Interpolation of Data

In some signal processors the input sampling rate may be made different from the output sampling rate for efficiency of processing. A slow sampling rate can be interpolated by the FFT-IFFT algorithm to increase the spectral detail of the output. For example, an interpolation is implemented to prepare the received time data for a subsequent D/A converter to produce pleasant music or a clear voice. A case in point is demonstrated in INTER-PO.CPP. Suppose we have received the time data shown in Figure 4.22(a).

We stretch the input spectrum, Figure 4.22(b), by breaking the spectrum into two at N/2=8 and insert enough zeros to meet the ratix-2 requirement, Figure 4.22(c). The stretched frequency data is IFFT'ed. The output is shown in Figure 4.22(d).

Compare Figure 4.22(a, d). It shows quite an improvement in signal quality.

4.6.4 Pulse Compression

In pulsed radar systems a transmitted power of a megawatt or higher is not rare in order to cover a longer search range. A high-power system is voluminous, heavy, and expensive. Let us examine the following two pulsed radar systems. One has one megawatt output power with 1-ms pulse width, the other a 100-kW and 10-ms pulse width. The energies in both systems are identical. One has the range resolution of 150m, the other 1.5 km.

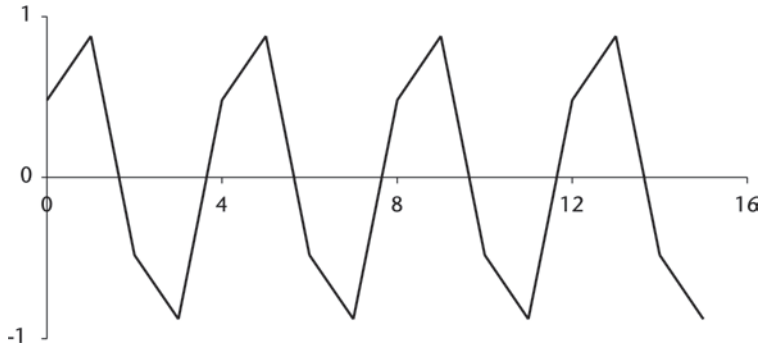


Figure 4.22(a) Time-sampled input data.

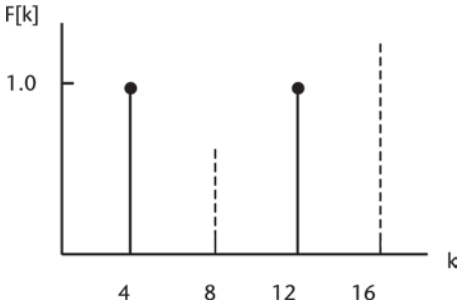


Figure 4.22(b) Spectrum of input data.

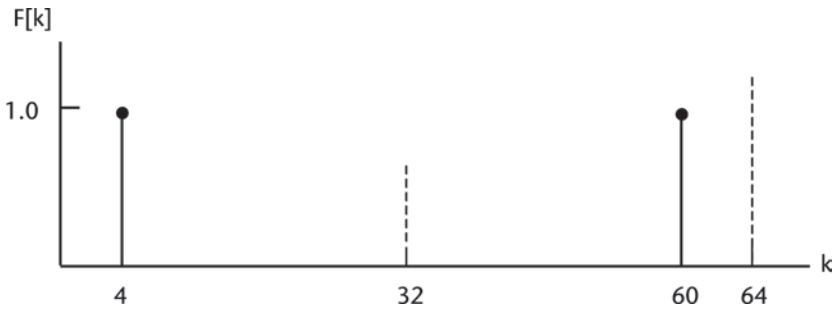


Figure 4.22(c) Stretched spectrum, zero-padded.

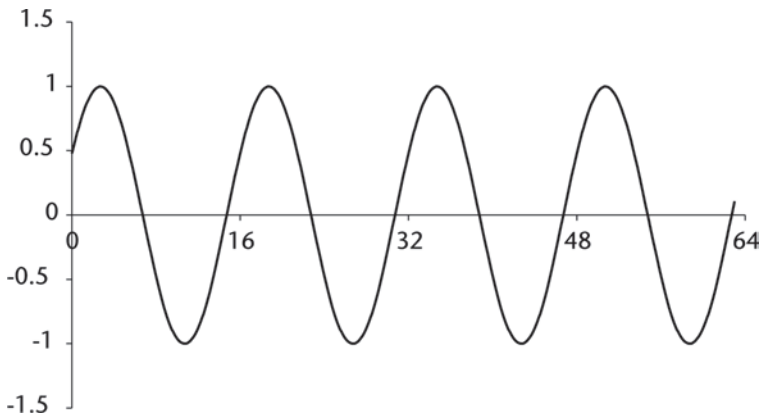
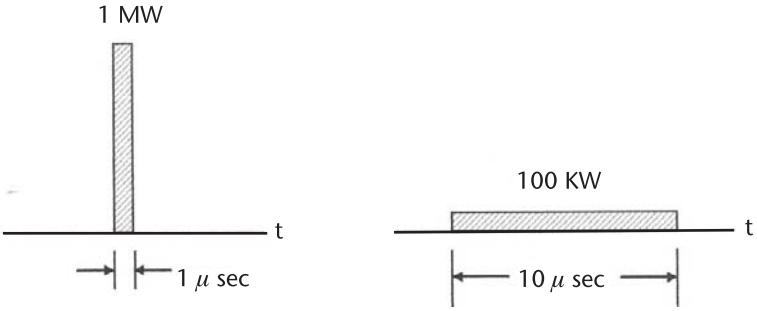


Figure 4.22(d) Interpolated input signal.



The pulse compression scheme is to improve the range resolution of the second radar comparable to the first by modulating the 10-ms waveform. The reflected returned signal will be compressed by an FFT-IFFT algorithm since the fast Fourier transform can be implemented in real time. There are three basic modulation techniques: frequency modulation, phase modulation, and polarization modulation.

We discuss a typical linear frequency modulation. The transmitter frequency is linearly modulated, up-chirp or down-chirp, and upon reception of the returned signal we plan to compress the 10-ms pulse to a narrower pulse width. Figure 4.23 depicts the principle idea involved.

If we somehow delay the lower frequencies of the signal most, in Figure 4.23 by almost 2T, and the higher frequencies by T, and the intermediate frequencies linearly between T and 2T, the returned signal waveforms will be heaped upon one another to produce a peaked pulse of a narrower pulse width comparable to the first radar. Detailed theoretical analysis can be found in [5–7].

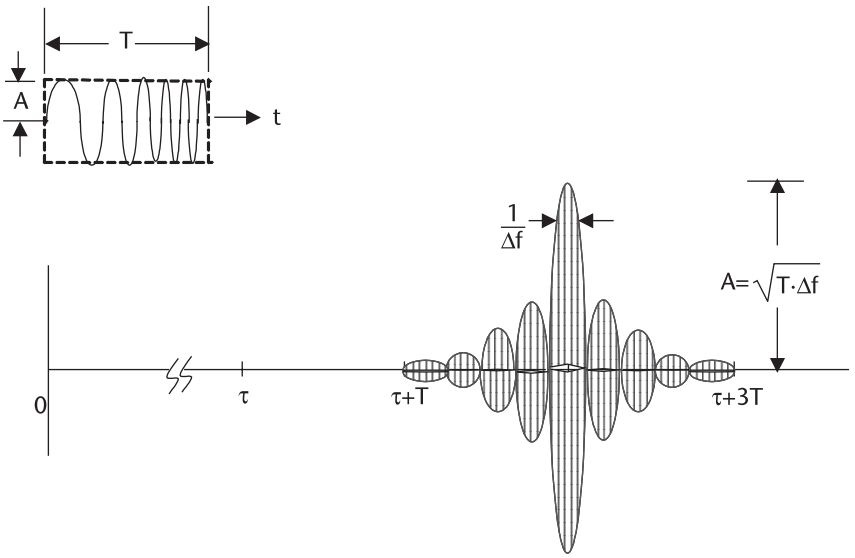


Figure 4.23 Linear FM pulse and compressed pulse.

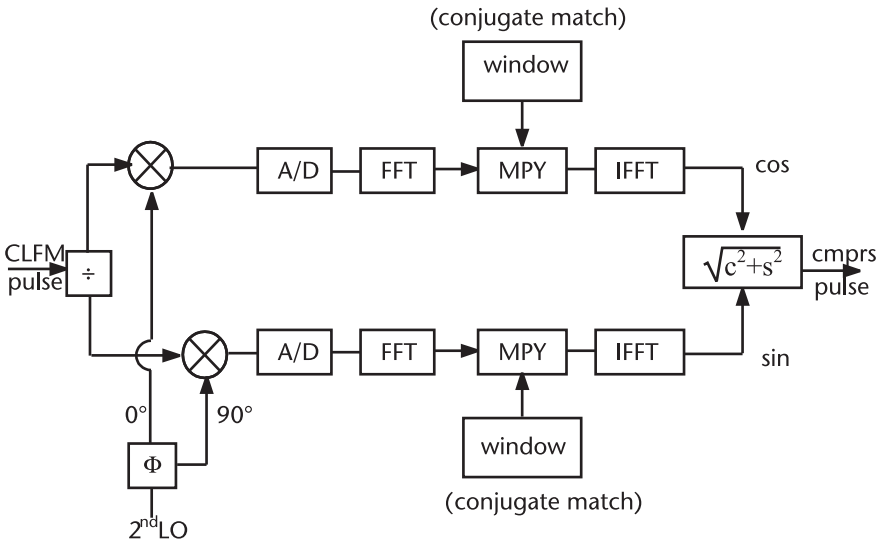


Figure 4.24 Pulse compression, FFT-conjugation-IFFT.

In earlier days, the proportional time delay was implemented by cascading a dozen or so of "bridged-tee" circuits of high-quality inductors and capacitors. There were other implementations using surface-acoustic wave devices.

Once the FFT-IFFT algorithm was understood the implementation took a different form, eliminating the bulky and hard-to-maintain circuits and devices.

Pulse compression through FFT-IFFT can be heuristically described by the following receiver structure.

In this section we study pulse compression using coherent linear frequency modulation (CLFM) through a receiver structure shown in Figure 4.24. For a demonstration the CLFM has a time-bandwidth product of 32, $T\Delta f=32$ and a compression gain of 15.05 dB with a rectangular window. There are many waveforms that have a time-bandwidth product of 32 as shown in Table 4.1.

There are many pulse compression radars with time-bandwidth products of thousands or more already developed and deployed. Certainly we can extend the above table to accommodate a large time-bandwidth product for a higher compression gain. (See Figure 4.25.)

Table 4.1 Waveforms with a Time-Bandwidth Product of 32

T(μ sec)	*Δf(MHz)*	*A/D converter speed (Nyquist rate)*	*N, number of time samples*
6.4	5.0	10.0	64
8.0	4.0	8.0	64
12.8	2.5	5.0	64
16.0	2.0	4.0	64
32.0	1.0	2.0	64

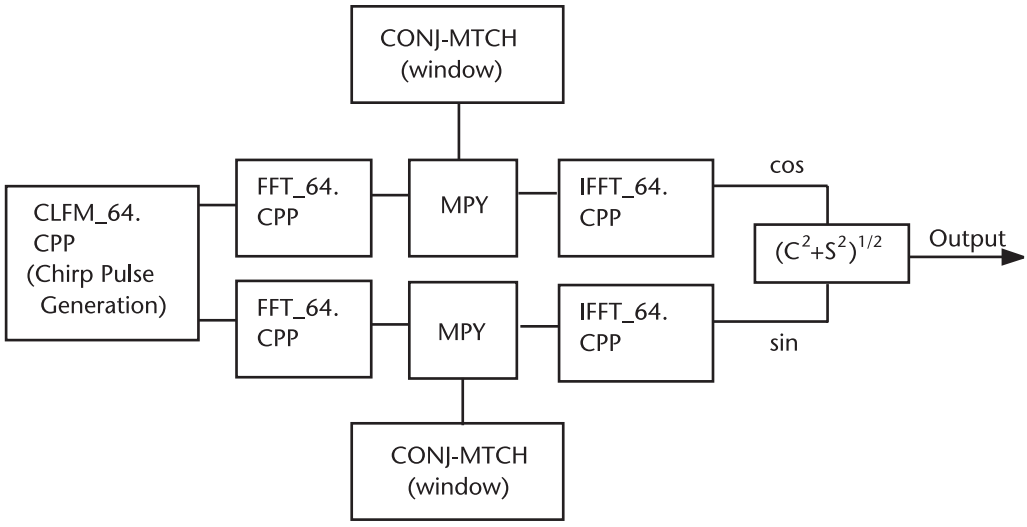


Figure 4.25 CLFM compression, TΔf=32, *N* = 64.

Our modularized demonstration programs follow Figure 4.25. The intermediate results are stored in DAT files for examination.

1. CLFM_64.CPP: Generation of CLFM, up-chirp signal; in-phase and quadrature-phase, N=64, T=12.8 μsec, Δf=2.5 MHz.
2. FFT_64.CPP: Two signals, in-phase and quadrature-phase are FFTed by decimation-in-frequency. The spectral magnitude is shown in Figure 4.27.
3. CONJ_MATCH: Conjugate-matched window. The envelope of the window is either rectangle or Hamming. Conjugate

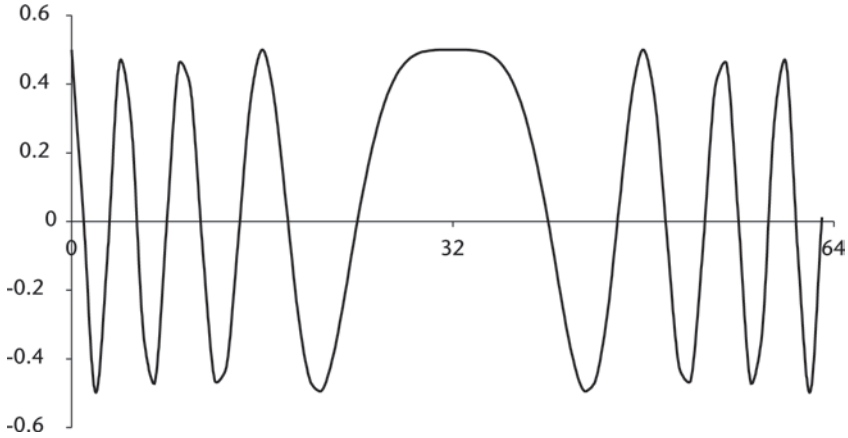


Figure 4.26(a) In-phase waveform.

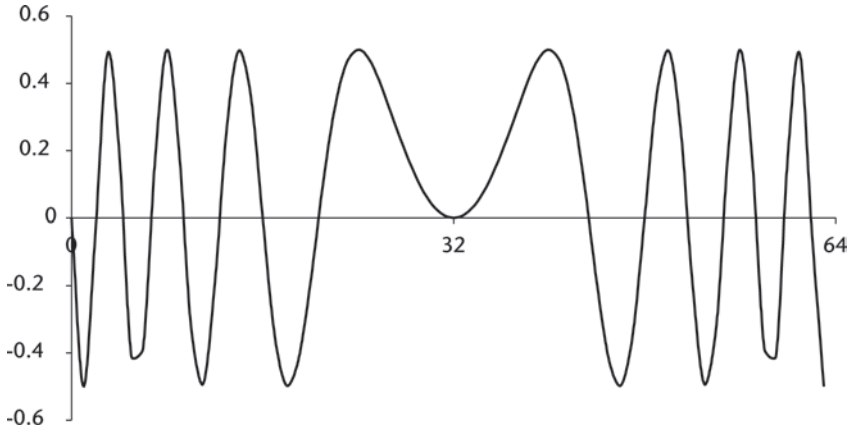


Figure 4.26(b) Quadrature-phase waveform.

matching is synonymous to the auto-correlation previously mentioned:

$G(f)\ G^*(f) = |\ G(f)\ |^2$

When G(f) is a complex, G(f) = a +jb

$G(f)\ G^*(f) = (a+jb)(a-jb) = a^2+b^2,$ real

4. IFFT_64.CPP: The conjugately matched signals, real, are the input to the IFFT module, decimation-in-frequency. Two outputs are squared, summed, and square-rooted. The compressed pulse is stored in IFFT_64.DAT, shown in Figure 4.28.

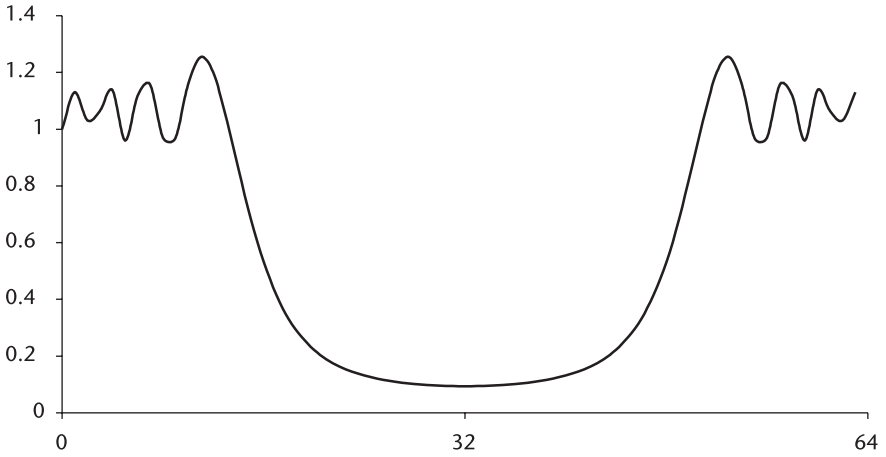


Figure 4.27 Spectrum magnitude.

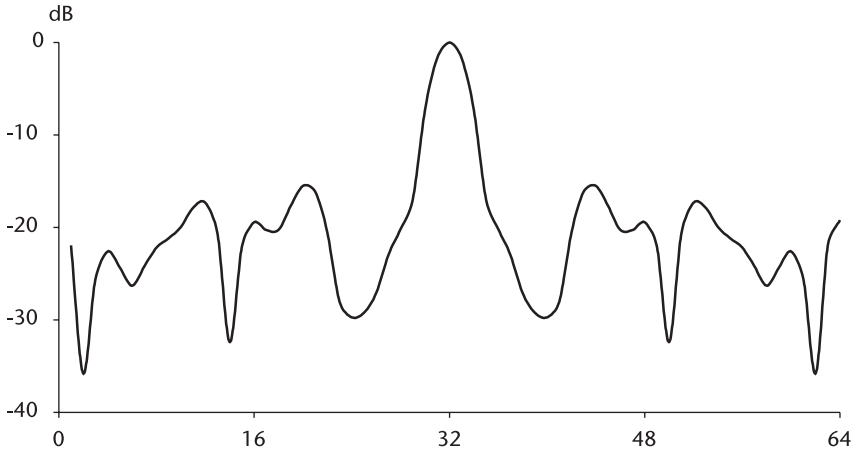


Figure 4.28 Compressed pulse.

The result indicates that the compression gain is 14.21 dB instead of the 15.05 dB theoretically predicted. The main reason for the discrepancy is spectrum spill over beyond Δf/2, as shown in Figure 4.29(a, b). We incur an additional loss if we apply a Hamming window to reduce the sidelobe levels.

There is a trade-off between sidelobe reduction and the loss of compresion gain. A Chebyshev window gives the narrowest pulsewidth with constant sidelobe level. The Hamming window gives sidelobes diminishing in amplitude but a slightly wider compressed pulse. Taylor window [8], (a modified Chebyshev) may be the most popular window. Consult Harris [9] for a variety of windows available.

Our demonstration program is written in three separate programs to show the step-by-step procedure. A second demonstration program, CMPR_128. CPP is written in one continuous operation. The design specifications are listed as follows.

- T 12.8 μsec
- Δf: 5.0 MHz
- TΔf: 64
- A/D converter speed
- (Nyquist rate) 10.0 MHz
- ts: sampling interval 100.0 nsec
- N: 128

The compressed data is stored in CMPR_128.DAT. The result shows a compression gain of 17.17 dB with a rectangle window instead of the 18.06 dB theoretical and 11.98 dB with a Hamming window.

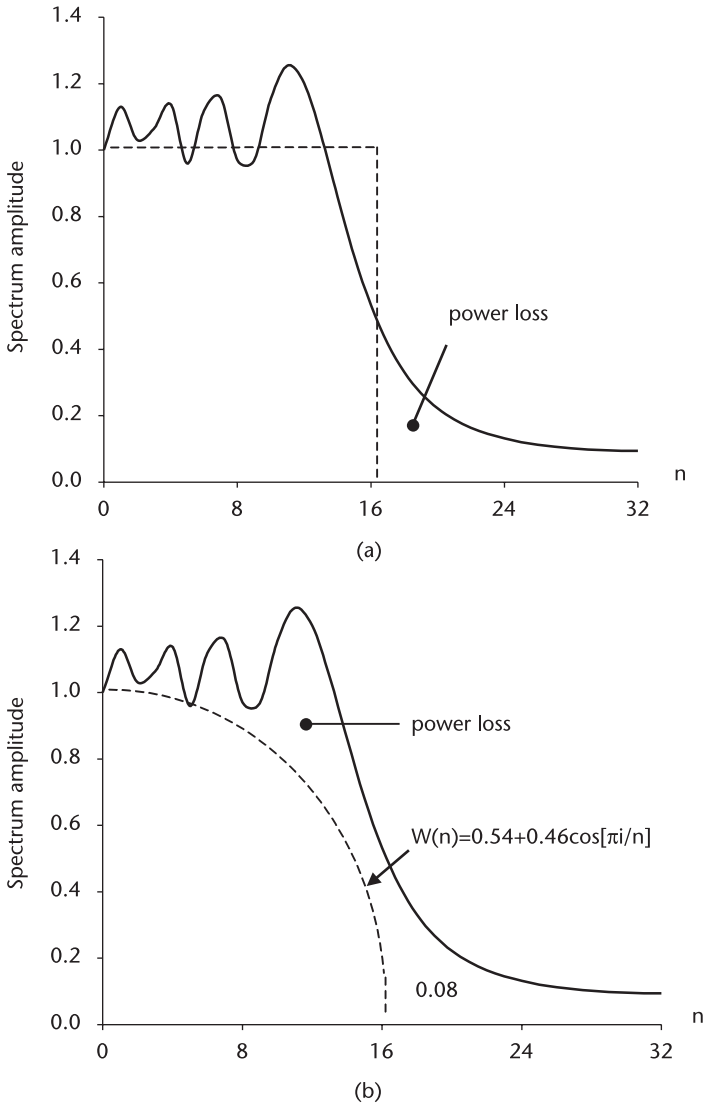


Figure 4.29 (a) Power loss, rectangular window. (b) Power loss, hamming window.

We should remind ourselves that if we limit the word length of the A/D converter to less than 12 bits (let's say 8 bits) we should expect the gain and sidelobe levels to deteriorate.

In earlier days we had to design a rack of dedicated PCB boards for pulse compression. Nowadays the system engineers routinely integrate modified PC in their system.

Phase modulation instead of frequency modulation may be entertained if the propagation medium would support the coherency of phase modulation. The ocean may not maintain the phase coherency. Frequency-hopping modulation (Chapter 5) should be given serious consideration in a nonuniform saline medium.

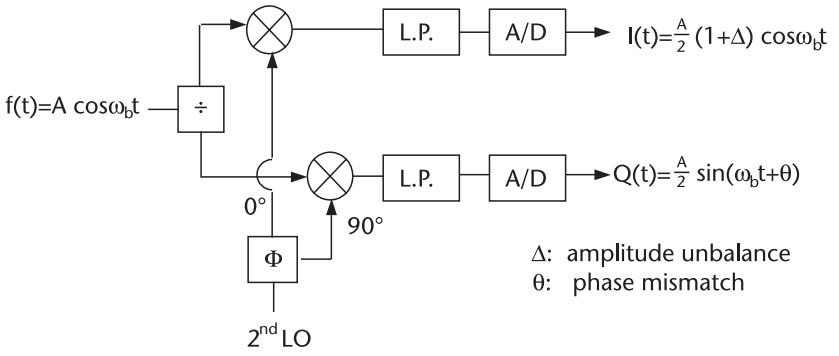


Figure 4.30 Representation of amplitude unbalance and phase mismatch between I and Q channels.

4.6.5 Amplitude Unbalance and Phase Mismatch

Let us examine Figure 4.30 where an in-phase and quadrature channel of a coherent receiver is shown. A coherent receiver consists of a power divider, a phase splitter, two frequency down-converters, two lowpass filters, and two A/D converters. This hardware will invariably produce an amplitude unbalance and phase mismatch between the two channels.

An amplitude unbalance of 0.5–2.0 dB, and phase mismatch of anywhere between 2.0–10.0 degrees over the operating environment of temperature, humidity, and the component fatigue are expected.

The amplitude unbalance and phase mismatch are lumped in I channel and Q channel, respectively.

Let us analyze the effects of the unbalance and mismatch on subsequent signal processing such as pulse compression or Doppler detection. The unbalance and mismatch are lumped in the I and Q channels as,

$$\begin{aligned} f(t) &= I(t) + jQ(t) \\ &= A/2(1+\Delta)\cos\omega_b t + jA/2\sin(\omega_b t + \theta) \\ &= A/2\cos\omega_b t + \Delta A/2\cos\omega_b t + jA/2\,(\sin\omega_b t \cdot \cos\theta + \cos\omega_b t \cdot \sin\theta) \end{aligned}$$

For a very small θ, let $\cos\theta \approx 1$, $\sin\theta \approx \theta$. Then,

$$\begin{aligned} f(t) &= A/2\cos\omega_b t + \Delta A/2\cos\omega_b t + jA/2\sin\omega_b t + jA\theta/2\cos\omega_b t \\ &= A/2\, e^{+j\omega_b t} + \Delta A/4(e^{+j\omega_b t} + e^{-j\omega_b t}) + jA\theta/4(e^{+j\omega_b t} + e^{-j\omega_b t}) \\ &= A/2[(1+\Delta/2) + j\theta/2]\, e^{+j\omega_b t} + A/4(\Delta + j\theta)\, e^{-j\omega_b t} \end{aligned}$$

The final expression shows that the unbalance and mismatch produce an image frequency at $e^{-j\omega_b t}$. The ratio of power at the image frequency to the primary frequency is given by,

$$\begin{aligned} \frac{\text{power at imaginary}}{\text{power at primary}} &= \frac{(A/4)^2(\Delta^2+\theta^2)}{(A/2)^2[(1+\Delta/2)^2+(\theta/2)^2]} \\ &= \frac{\Delta^2+\theta^2}{4[(1+\Delta/2)^2+(\theta/2)^2]} \end{aligned} \tag{4.7}$$

The following table gives the power levels (decibels) at the image frequency with respect to the primary, computed from (4.7).

θ \ Δ	*0.5 dB*	*1.0 dB*	*1.5 dB*
1°	-30.5	-24.7	-21.3
5°	-25.8	-23.0	-20.5
10°	-21.0	-20.0	-18.6

All combinations in the table produce an image power higher than -30 dB except one. Equation (4.7) is programmed in IQ_IMAGE.CPP. If we must have the image power level below –30 dB for any signal processing subsequent to A/D converters, we have a problem unless the amplitude unbalance is less than 0.5 dB and the phase mismatch less than 1°. The conditions are very stringent, and the hardware would be unable to meet them. See Figure 4.31.

In order to correct the amplitude unbalance and phase mismatch, Churchill et al. [10] proposed a corrective operation shown in Figure 4.31.

$$\begin{cases} I' = EI \\ Q' = PI + Q \end{cases}$$

or

$$\begin{bmatrix} I' \\ Q' \end{bmatrix} = \begin{bmatrix} E & 0 \\ P & 1 \end{bmatrix} \begin{bmatrix} I \\ Q \end{bmatrix}$$

if

$$E = \frac{\cos\theta}{1+\Delta}, \quad \text{and} \quad P = \frac{-\sin\theta}{1+\Delta}$$

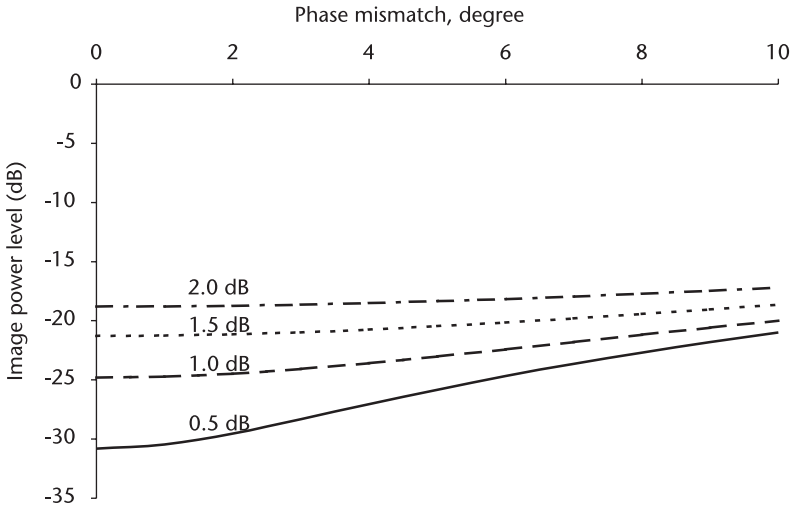


Figure 4.31 Power level at image frequency.

then,

$$I' = A/2 \cos\theta \cdot \cos \omega_b t$$
$$Q' = A/2 \cos\theta \cdot \sin \omega_b t$$

We see that the corrected I' and Q' have equal amplitudes and that the phase mismatch has vanished. The task is to determine E and P when Δ and θ are unknown a priori and time-varying.

We take the fast Fourier transform of the unbalanced and mismatched signals I and Q.

$$f[n] = A/2\,(1+\Delta)\cos[n] + jA/2\sin(n+\theta)$$

$$F[k] = \frac{1}{N}\sum_{n=0}^{N-1} f[n]e^{-j(2\pi/N)nk}$$

It can be shown that the real and imaginary components of F[k] are given by, (See Appendix 4A)

$$F[k]_{real} = \frac{1}{N}\frac{A}{4}(1+\Delta)\sum_{n=0}^{N-1}[e^{j2\pi n(1-k)/N} + e^{-j2\pi n(1+k)/N}]$$

$$F[k]_{imag} = \frac{1}{N}\frac{A}{4}\cos\theta\sum_{n=0}^{N-1}[e^{j2\pi n(1-k)/N} - e^{-j2\pi n(1+k)/N}] + \frac{1}{N}\frac{jA}{4}\sin\theta\sum_{n=0}^{N-1}[e^{j2\pi n(1-k)/N} + e^{-j2\pi n(1+k)/N}]$$

When k=1, the primary frequency, the spectrum is given by

$$F[1] = \frac{1}{N}\frac{A}{2}[(1+\Delta) + \cos\theta + j\sin\theta]$$

When k=N–1, the image frequency, the spectrum is given by

$$F[N-1] = \frac{1}{N}\frac{A}{2}[(1+\Delta) - \cos\theta + j\sin\theta]$$

We take the following ratio:

$$\frac{F[N-1]}{1/2\{F^*[1] + F[N-1]\}} = 1 - \frac{\cos\theta}{1+\Delta} + j\frac{\sin\theta}{1+\Delta}$$

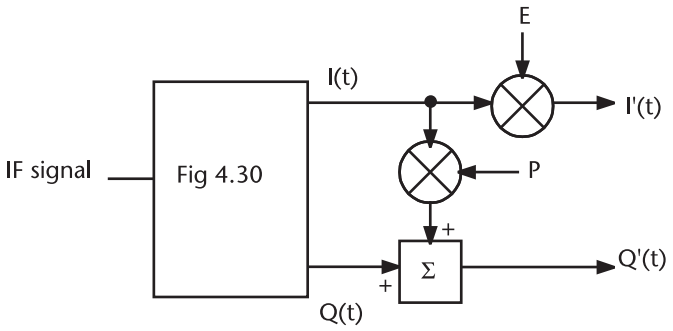


Figure 4.32 Correction factor E and P.

We note that the real term contains the correction factor E. The imaginary term is negative of correction factor P. We modify Figure 4.32 accordingly as shown in Figure 4.33.

The final corrective network is shown in Figure 4.34. The FFT module is activated periodically, preferably at the end of range bins having a specified test frequency injected at the IF port. In order to improve the accuracy of E and P, we should average several E's and P's. A demonstration program is written in IQ_ERROR.CPP without averaging. The program assumed the following parameters:

- Δ: 1.0 dB unbalance;
- θ: 5.0 degrees mismatch;
- ts: 0.25 μsec, sampling interval;
- N: 32;
- Test frequency: 62.5 KHz = MHz/32;
- F[1]: Primary frequency;
- F[63]: Image frequency.

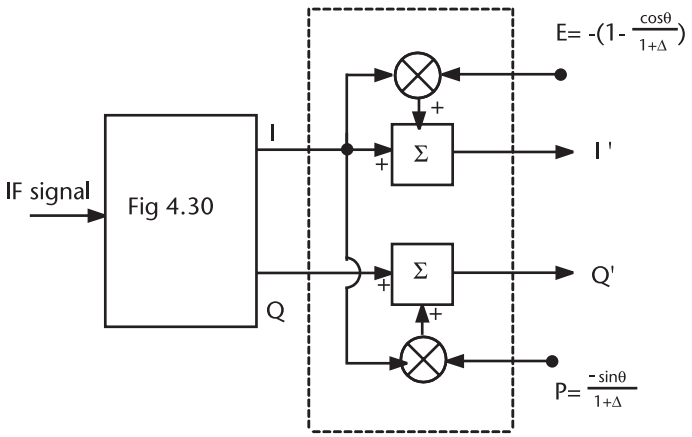


Figure 4.33 Correction factors found by FFT.

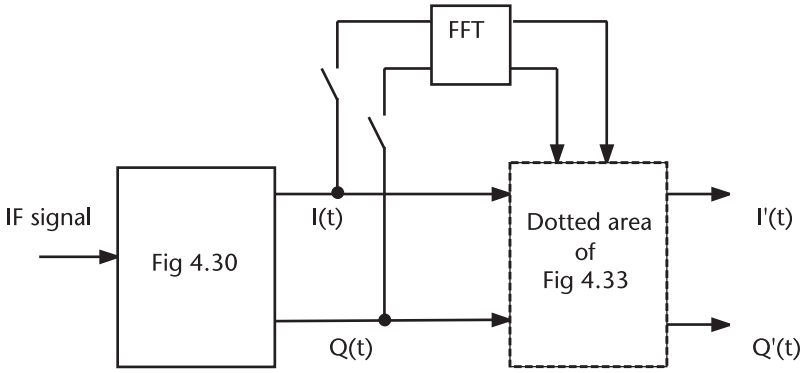


Figure 4.34 Unbalance and mismatch corrections through FFT.

The correction factors E and P turn out to be –0.1121 and –0.0776, respectively.

We have demonstrated that a long pulse with a lower transmitter power radar could replace a high power with short pulse provided that the long pulse is linearly frequency-modulated and that the returned signal is compressed by the FFT-IFFT operation in a coherent IQ receiver. This amazing achievement has a hidden shortcoming: so-called range-Doppler coupling. The phenomenon of the coupling will be analyzed in Chapter 5. The ambiguity principle states that if we improve the range resolution, the velocity measurement (Doppler frequency) would deteriorate, and vice versa. We shall study the ambiguity function in Chapter 5.

Appendix 4A

The Fourier transform pair of amplitude unbalance and phase mismatch are given by

$$f[n] = A/2\ (1+\Delta)\cos[n] + jA/2\sin(n+\theta)$$

$$F[k] = \frac{1}{N}\sum_{n=0}^{N-1} f[n]e^{-j(2\pi/N)nk}$$

Substitute exponential expressions for the cosine and sine functions

$$\cos[n] = 1/2(e^{j(2\pi/N)nk} + e^{-j(2\pi/N)nk})$$
$$\sin[n] = 1/j2(e^{j(2\pi/N)nk} - e^{-j(2\pi/N)nk})$$

Then, the real and imaginary F[k] are given by

$$F[k]_{real} = \frac{1}{N}\sum_{n=0}^{N-1}\frac{A}{2}(1+\Delta)\frac{1}{2}(e^{j(2\pi/N)nk} + e^{-j(2\pi/N)nk})e^{-j2\pi nk/N}$$
$$= \frac{1}{N}\frac{A}{4}(1+\Delta)\sum_{n=0}^{N-1}[e^{j2\pi n(1-k)/N} + e^{-j2\pi n(1-k)/N}]$$

$$F[k]_{imag} = \frac{1}{N}\frac{jA}{2}\sum_{n=0}^{N-1}\sin\,(n+\theta)e^{-j2\pi nk/N}$$
$$= \frac{1}{N}\frac{jA}{2}\sum_{n=0}^{N-1}\{\sin\,[n]\,\cos\,\theta + j\,\cos\,[n]\,\sin\,\theta\}\,e^{-j2\pi nk/N}$$
$$= \frac{1}{N}\frac{A}{4}\cos\,\theta\sum_{n=0}^{N-1}[e^{j2\pi n(1-k)/N} - e^{-j2\pi n(1+k)/N}]$$
$$+ \frac{j}{N}\frac{A}{4}\sin\,\theta\sum_{n=0}^{N-1}[e^{j2\pi n(1-k)/N} + e^{-j2\pi n(1+k)/N}]$$

When k=1,

$$F[1] = \frac{1}{N}\frac{A}{4}[(1+\Delta) + \cos\,\theta + j\,\sin\,\theta]$$

When k=N-1,

$$F[N-1] = \frac{1}{N}\frac{A}{4}[(1+\Delta) - \cos\,\theta + j\,\sin\,\theta]$$

The ratio:

$$\frac{F[N-1]}{1/2\{F^*[1] + F[N-1]\}} = \frac{\text{numerator}}{\text{denominator}} = 1 - \frac{\cos\,\theta}{1+\Delta} + j\frac{\sin\,\theta}{1+\Delta}$$

where:

Numerator = $(1+\Delta) - \cos\,\theta + j\,\sin\,\theta$;
Denominator = $\frac{1}{2}\{[(1+\Delta) + \cos\,\theta - j\,\sin\,\theta] + [(1+\Delta) - \cos\,\theta + j\,\sin\,\theta]\}$.

List of Programs

Program	*Features*
(1) DFTvsFFT.CPP	Compares the number of multiplications and adds required in the discrete Fourier transform and fast Fourier transform
(2) BIT_REV.CPP	Bit-reversal operation
(3) FFT_DIT.CPP	Fast Fourier transform with decimation-in-time
(4) FFTDIT.H	Header file where bit-reversal and butterfly operations are executed
(5) FFT_DIF.CPP	Fast Fourier transform with decimation-in-frequency
(6) FFTDIF.H	Header file where the butterfly operation is first and bit-reversal follows
(7) LEAKAGE1.CPP	First example of spectral leakage
(8) LEAKAGE2.CPP	Second example of spectral leakage
(9) LEAKAGE3.CPP	Third example of spectral leakage
(10) IFFT_DIT.CPP	Inverse FFT with decimation-in-time
(11) IFFTDIT.H	Header file where bit-reversal is first, then butterfly follows
(12) IFFT_DIF.CPP	Inverse FFT with decimation-in-frequency
(13) IFFTDIF.H	Header file where butterfly operation is first, and bit-reversal follows
(14) FFT_FILT.CPP	Filtering through FFT+window+IFFT
(15) SIGNOISE.CPP	Signal-to-noise ratio enhancement through FFT+IFFT
(16) INTERPO.CPP	Interpolation of time-sampled data
(17) CLFM_64.CPP	Generation of I and Q channel signals, N=64
(18) FFT_64.CPP	Spectral computation of I and Q signals
(19) IFFT_64.CPP	Pulse compression by IFFT after conjugate window
(20) CMPRS128.CPP	Second demonstration of pulse compression, N=128
(21) IQ_IMAGE.CPP	Computes power levels at the image frequency
(22) IQ_ERROR.CPP	Computes the correction factors for amplitude unbalance and phase mismatch

References

[1] Oppenheim, A.V., and R.W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

[2] Rabiner, L. R., and B. Gold, *Theory and Application of Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1975.

[3] Brigham, E. O., *The Fast Fourier Transform and Its Applications,* Englewood Cliffs, NJ: Prentice-Hall, 1988.

[4] Press, W. H., et al., *Numerical Recipes in C,* second ed., Cambridge, UK: Cambridge Univ. Press, 1995.

[5] Cook, C. E., and M. Bernfeld, *Radar Signal, an Introduction to Theory and Application,* Norwood, MA: Artech House, 1993.

[6] Rihacek, A. W., *Principles of High-Resolution Radar*, Norwood, MA: Artech House, 1996.

[7] Barton, D. K., *Pulse Compression Radar*, Vol. 3, Dedham, MA: Artech House, 1975.

[8a] Taylor, T. T., "Design of Line Source Antenna for Narrow Beamwidth and Low Sidelobes," *IRE Trans*., Vol. AP-3, Jan. 1955.

[8b] Taylor, T. T., "Design of Circular Aperture for Narrow Beamwidth and Low Sidelobes," *IRE Trans*., Vol. AP-8, Jan. 1960.

[9] Harris, F. J., "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proc. of IEEE*, Vol. 66, No.1, Jan. 1978.

[10] Churchill, F. E., G. W. Ogar, and B. J. Thompson, "The Correction of I and Q Error in a Coherent Processor," *IEEE Trans*., Vol. AES-17, No.1, Jan. 1981.

[11] Churchill, F. E., G.W. Ogar, and B. J. Thompson, U.S. Patent #3,050,750, Apr. 1976.

CHAPTER 5

Ambiguity Function

5.1 Introduction

The measurement uncertainties in range and Doppler frequency can be analyzed by the ambiguity function. The function is originally introduced by Woodward [1] from a different perspective; however, the credit is accorded to him. The response function describing the output of a matched filter to a waveform u(t) that is shifted in time delay τ and Doppler frequency f_d is

$$\chi(\tau, f_d) = \frac{1}{2E}\int_{-\infty}^{\infty} u(t)\, u^*(t-\tau)\, e^{+j2\pi f_d t}\, dt \tag{5.1}$$

where

- u(t): The complex envelope of transmitted signal;
- $u^*(t-\tau)$: The complex conjugate of transmitted signal delayed by τ due to target range;
- f_d: The Doppler shifted carrier frequency;
- E: The total energy.

The above expression is sometimes called the "Time-frequency correlation function" from a mathematician's point of view, or the "uncertainty function" from a physicist's point of view, analogous to Heisenburg's uncertainty principle.

The squared magnitude $|\chi(\tau, f_d)|^2$ is called the ambiguity function. In our discussion the ambiguity function is $|\chi(\tau, f_d)|^2$, and when a need arises to clarify we shall note it with absolute symbol $|\cdot|$ or $|\cdot|^2$.

The ambiguity function has an alternative expression in the frequency domain. The ambiguity function has the following three principal properties:

1. $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\chi(\tau, f_d)|^2\, d\tau\, df_d = 1$;
2. $|\chi(\tau, f_d)| \le |\chi(0, 0)| = 1$;
3. $|\chi(\tau, f_d)| = |\chi(-\tau, -f_d)|$.

The property (1) states that the volume under the surface of the ambiguity function in the τ-f_d plane is constant, unity. The property (2) states that the function is maximum at the origin and is smaller in magnitude elsewhere. The property (3) states that the function is symmetrical with respect to $\tau = 0$ and $f_d = 0$.

The ambiguity surface (plot of $|\chi(\tau, f_d)|^2$) is the central part of an analysis of target resolution or separability of targets in the range and Doppler frequency. The width of the main peak of the function at the origin is a measure of resolution.

We shall study the ambiguity function of three basic transmitted signals.

1. Rectangular pulse with a single constant frequency;
2. Rectangular pulse with linear frequency modulation;
3. Rectangular pulse with Costas-coded frequency hopping.

Phase modulations are excluded in our discussion because of their poor tolerance to high-Doppler targets [2]. The binary phase code or the Frank polyphase code has higher sidelobes responses than a linear frequency modulation.

5.2 Rectangular Pulse with a Single Constant Frequency

The ambiguity function of rectangular pulse without any modulation on carrier frequency is obtained by substituting

$$u(t) = \begin{cases} e^{+j2\pi ft} & \tau \le t \le T \\ 0 & \text{otherwise} \end{cases}$$

into (5.1). The response function of a simple rectangular pulse is given by,

$$\chi(\tau, f_d) = \int_{-\infty}^{\infty} u^*(\tau)u(\sigma - \tau)\ e^{+j2\pi f_d \sigma}\ d\sigma$$

$$= \frac{1}{T}\int_{\tau}^{T} e^{-j2\pi f\sigma}\ e^{+j2\pi f(\sigma-\tau)}\ e^{+j2\pi f_d\sigma}\ d\sigma$$

$$= e^{-j2\pi f\tau}\ e^{+j\pi f_d(T+\tau)}\left(1 - \frac{\tau}{T}\right)\frac{\sin\left[\pi f_d T\left(1 - \frac{\tau}{T}\right)\right]}{f_d T\left(1 - \frac{\tau}{T}\right)} \qquad (5.2)$$

The magnitude of the response function of a rectangular pulse with constant frequency is then

$$|\chi(t, f_d)| = \left(1 - \frac{|\tau|}{T}\right)\left|\frac{\sin\left[\pi f_d T\left(1 - \frac{\tau}{T}\right)\right]}{\pi f_d T\left(1 - \frac{\tau}{T}\right)}\right| \qquad (5.3)$$

In order to visualize the ambiguity function we take two principal planes, one with $\tau = 0$, the other $f_d = 0$. Two magnitude expressions are shown in Figure 5.1.

When $\tau = 0$,

$$|\chi(0, f_d)| = \left|\frac{\sin[\pi f_d T]}{\pi f_d T}\right|$$

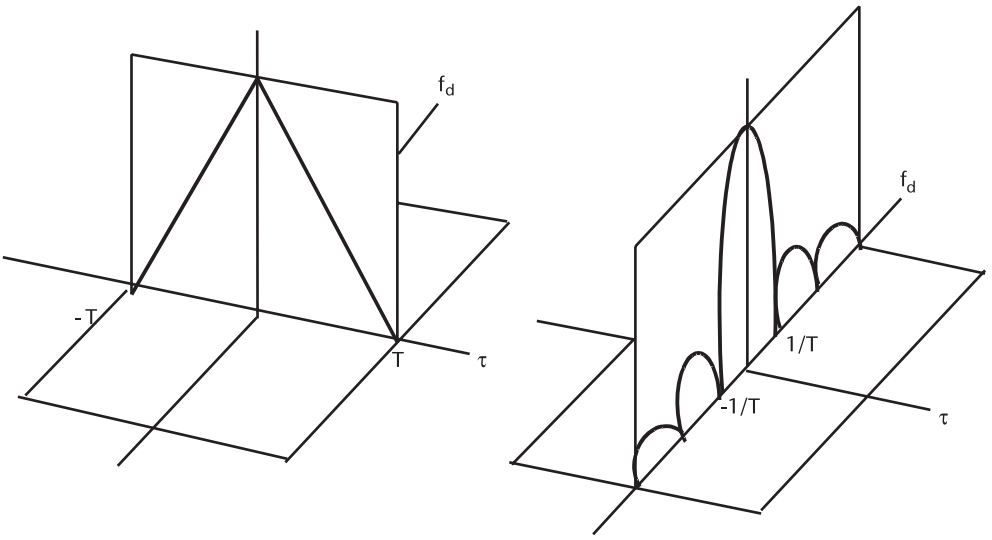


Figure 5.1 Two principal cuts of response function.

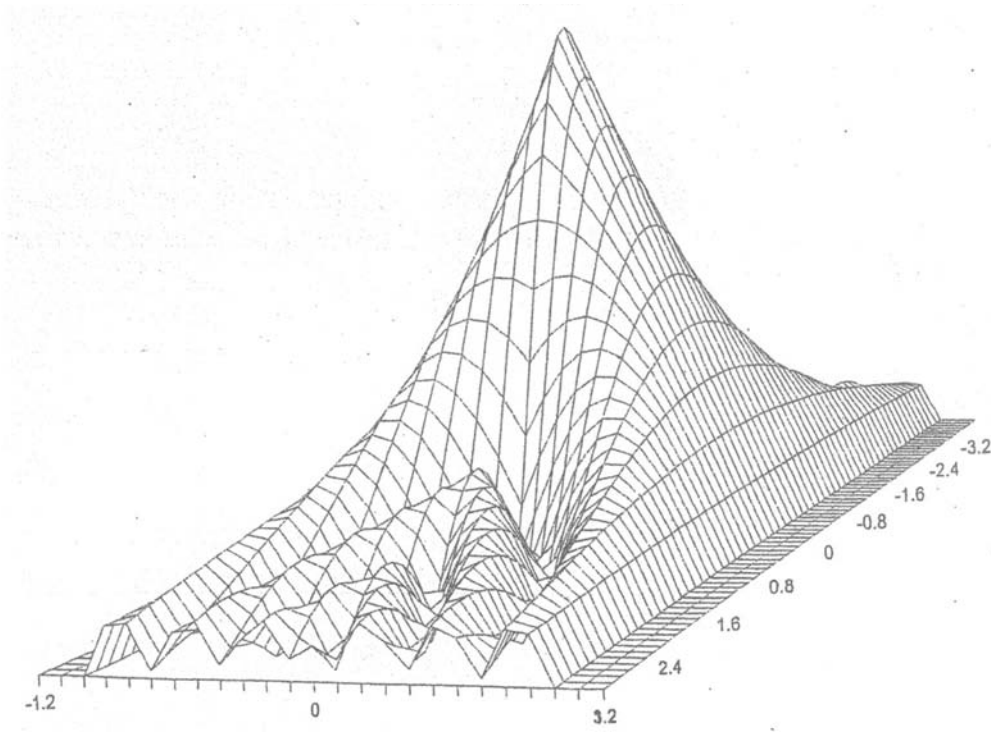


Figure 5.2 The ambiguity function of a rectangular pulse with constant frequency.

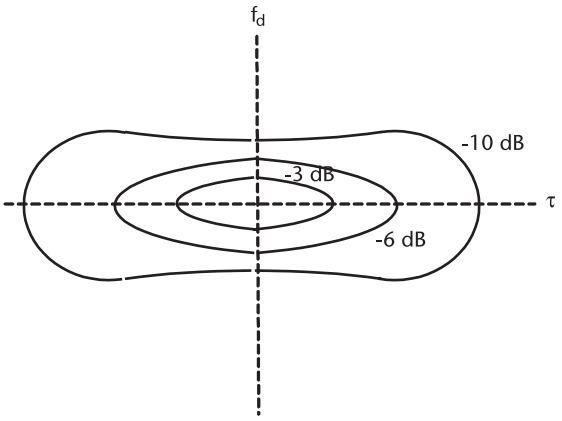


Figure 5.3 Contour map.

When $f_d = 0$,

$$|\chi(\tau, 0)| = 1 - \frac{|\tau|}{T}$$

We note that the longer the pulse, the higher the resolution in Doppler measurement, but the poorer in range resolution. The reverse is true: The shorter the pulse, the higher the resolution in range measurement but the poorer the Doppler resolution.

Eq (5.3) is programmed in AMB_RECT.CPP, and the ambiguity surface in 3D is shown in Figure 5.2. The contour level at –3dB is approximately an ellipse but the contour changes to a dog-biscuit-like shape when the level is –10 dB or lower.

5.3 Linear Frequency Modulation (LFM)

The complex envelope u(t) of a rectangular pulse with carrier frequency that is linearly frequency modulated is given by

$$u(t) = \begin{cases} e^{+j\pi\mu t^2} & \tau \le t \le T \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

The $j\pi\mu t^2$ is the instantaneous phase of the carrier. The carrier frequency is obtained by differentiating the phase

$$f(t) = \frac{d}{dt}\Phi(t) = \frac{1}{2\pi}(2\pi\,\mu t) = \mu t$$

The constant μ is the modulation rate in Hertz/second. We study linear FM for pulse compression in Chapter 4. There we mention the "range-Doppler coupling" phenomenon. We shall also analyze the ambiguity function of the LFM the pulse and the coupling between the range and Doppler frequency.

The response function is obtained by substituting (5.4) into (5.1).

$$\begin{aligned}\chi(\tau, f_d) &= \frac{1}{2E}\int_{-\infty}^{\infty} u^*(t)u(\sigma-\tau)e^{+j2\pi f_d\sigma}\,d\sigma \\ &= \frac{1}{T}\int_{\tau}^{T} e^{-j\pi\mu\sigma^2}\, e^{+j\pi\mu(\sigma-\tau)^2}\, e^{+j2\pi f_d\sigma}\,d\sigma \\ &= \frac{1}{T}\, e^{+j\pi\mu\tau^2}\int_{\tau}^{T} e^{-j2\pi(\mu\sigma-f_d)}\,d\sigma \\ &= \left(1-\frac{\tau}{T}\right)\frac{\sin\left[\pi T\left(1-\frac{\tau}{T}\right)(\mu\tau-f_d)\right]}{\pi T\left(1-\frac{\tau}{T}\right)(\mu\tau-f_d)}\end{aligned} \tag{5.5}$$

The magnitude of the response function is given by

$$|\chi(\tau, f_d)| = \left(1-\frac{|\tau|}{T}\right)\left|\frac{\sin\left[\pi T\left(1-\frac{\tau}{T}\right)(\mu\tau-f_d)\right]}{\pi T\left(1-\frac{\tau}{T}\right)(\mu\tau-f_d)}\right| \tag{5.6}$$

Equation (5.6) is identical to (5.3) except that the Doppler frequency f_d in (5.3) is replaced by $\mu\tau$-f_d in (5.6). This means that the ellipse of the contour map of rectangular pulse with no modulation is sheared off the principal axis. The center line of the sheared ridge is $f_d = \mu\tau$. This shearing causes coupling between the range and Doppler measurements.

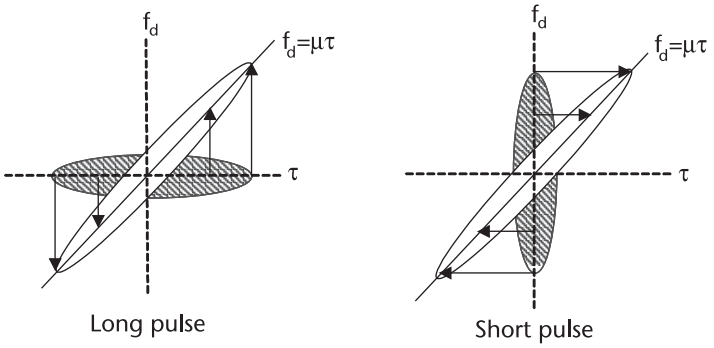


Figure 5.4 Long and short pulse sheared off from the principal axis.

A straight line $f_d = \mu\tau$ on τ-f_d plane is shown in Figure 5.4 for long pulses and short pulses. From (5.6) we obtain two principle cuts, $\tau = 0$ and $f_d = 0$ of a CLFM pulse.

$$|\chi(0, f_d)| = \left|\frac{\sin(\pi f_d T)}{\pi f_d T}\right| \tag{5.7a}$$

$$|\chi(\tau, 0)| = \left|\left(1 - \frac{|\tau|}{T}\right)\right| \left|\frac{\sin\left[\pi\mu\tau T\left(1 - \frac{\tau}{T}\right)\right]}{\pi\mu\tau T\left(1 - \frac{\tau}{T}\right)}\right| \tag{5.7b}$$

Equation (5.6) is programmed in AMB_CLFM.CPP, and the result is shown in Figure 5.6. Resolution is somewhat poor due to the slower A-D converter speed assumed.

An ideal ambiguity function (surface plot in 3D) is one that has a central peak at the origin of the τ-f_d plane with a circular or nearly circular contour level and rapidly, uniformly decreasing surface as depicted in Figure 5.7.

What transmitter waveform would produce a thumbtack-like ambiguity surface? A large number of research papers have reported on the subject.

We shall cover one of the waveforms that produces a thumbtack-like ambiguity surface on the τ-f_d plane in the next section.

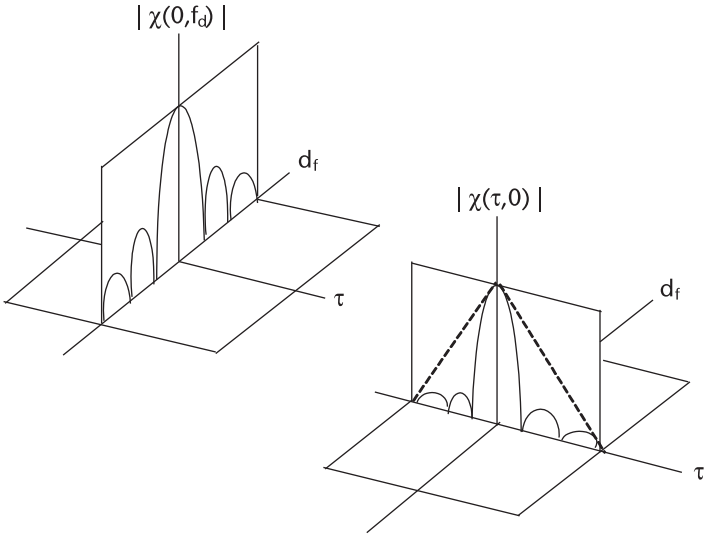


Figure 5.5 Two principal cuts of CLFM.

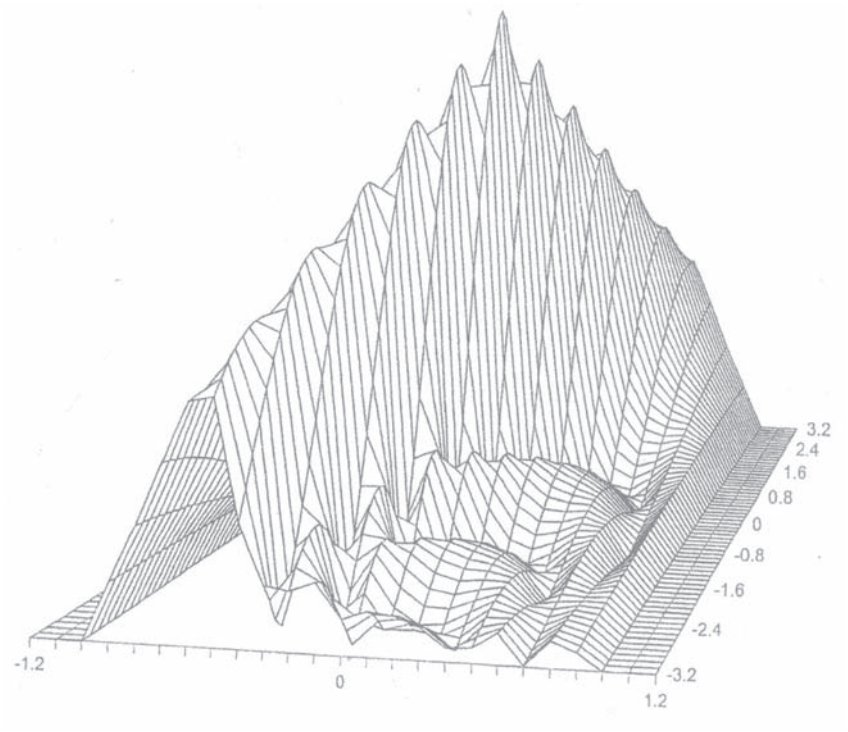


Figure 5.6 Ambiguity surface of the LFM pulse.

5.4 Costas-Coded Frequency Hopping Modulation

The Costas-coded waveform [3] has been analyzed with great interest because the ambiguity function resembles an ideal thumbtack. A rectangular pulse of duration

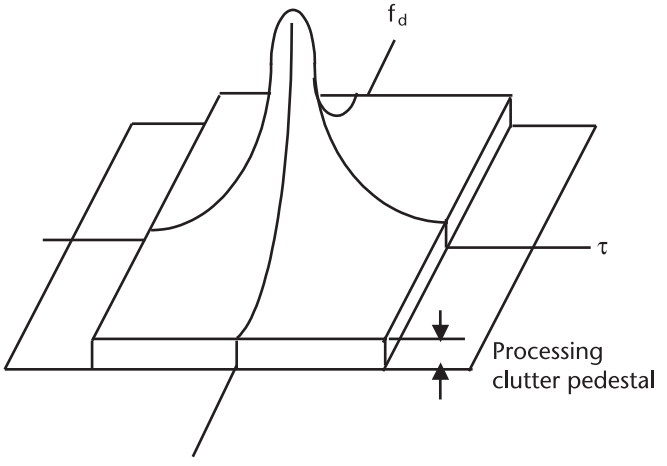
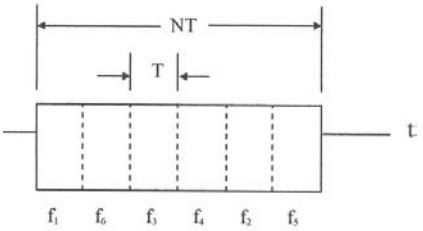


Figure 5.7 An ideal ambiguity surface, thumbtack-shape.

NT is divided into N subpulses where each subpulse has different and distinct frequency assigned.



The (6x6) time-frequency matrix shown in Figure 5.8 is one of many permutations available in time-frequency assignments. As a matter of fact there are N! such matrices for N=6: N!=720. Among the 720 matrices only 116 are Costas-coded waveforms. The Costas-coded signal is defined such that the ambiguity function produces a central peak at the origin and much lower sidelobes else- where in the τ-f_d plane.

Example of time-frequency matrices for N=4, N!=24 are shown in Figure 5.9 below; only 12 of them are Costas-code. Costas-code is indicated by a square bracket under the matrix. Matrices marked by a circular bracket do not belong to Costas-code. Whether or not a matrix belongs to Costas-code is determined by a triangle difference matrix originally reported by Costas [3], and later in a search algorithm presented by Golomb [4].

We note that the code at the upper left corner is that of a quantized staircase FM up-chirp and that the code at the lower right corner is that of a quantized staircase FM down-chirp. We have seen that a linear FM pulse has an elliptical contour sheared off the principal axis. From the point of view of thumbtack-like requirement, a linear FM is the worst choice even though linear FM has the versatility of adjusting the time-bandwidth product at will.

We shall derive the ambiguity function of a Costas-coded signal for N=5. The transmitted pulse is shown in Figure 5.10. A rectangular pulse of NT seconds long transmits five distinct and different frequencies for T seconds.

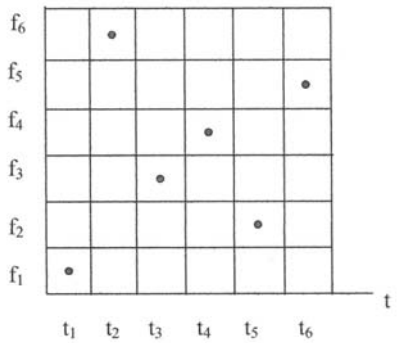


Figure 5.8 Time-frequency assignment matrix.

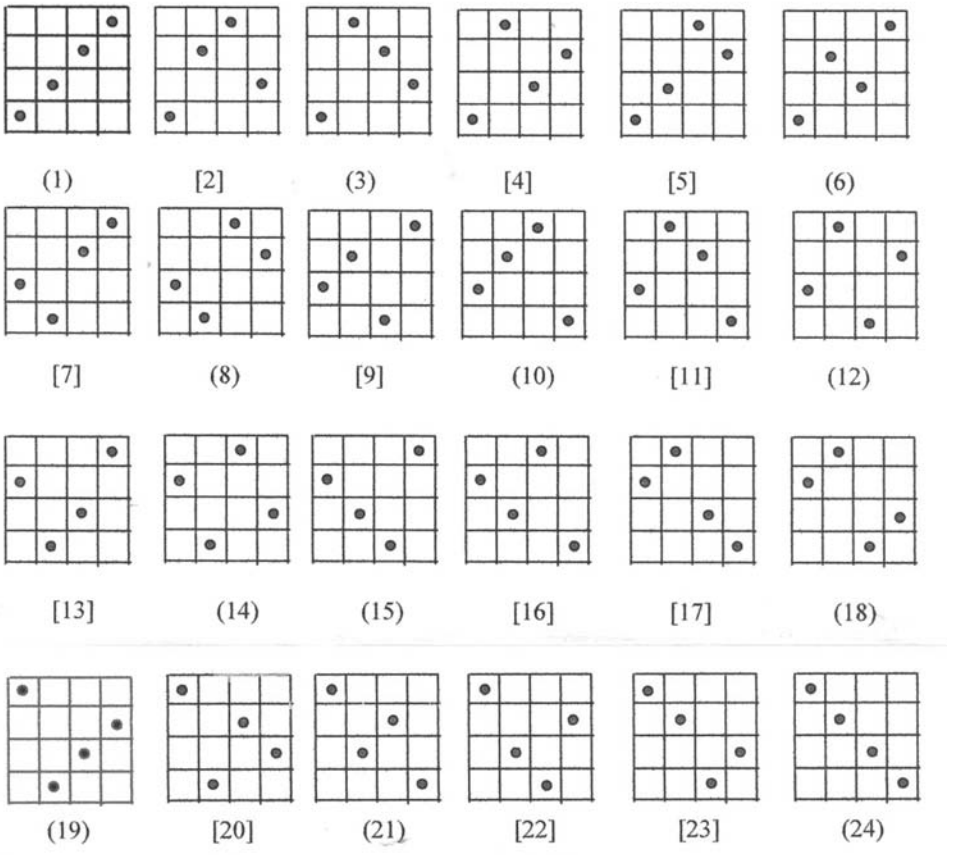


Figure 5.9 Costa-coded signals.

The transmitted signal u(t) is defined by,

$$u(t) = \sum_{n=0}^{N-1} u_n(t - nT) \tag{5.8}$$

and the complex envelope representation of each subpulse is given by,

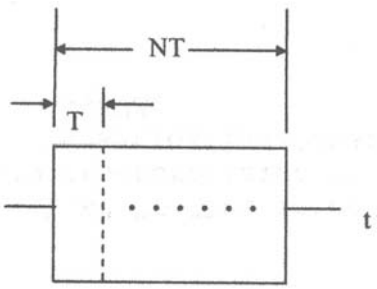


Figure 5.10 Costas-coded signal.

$$u_n(t) = \begin{cases} e^{+j2\pi f_n t} & \tau \le t \le T \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

The response function is obtained by substituting (5.8) and (5.9) into (5.1).

$$\chi(\tau, f_d) = \frac{1}{NT} \int_{-\infty}^{\infty} u^*(\sigma) u(\sigma - \tau) e^{+j2\pi f_d \sigma} \, d\sigma$$

$$= \frac{1}{NT} \int_{-\infty}^{\infty} \sum_{n=0}^{N-1} e^{-j2\pi f_n(\sigma - nT)} \sum_{m=0}^{N-1} e^{+j2\pi f_m(\sigma - \tau - mT)} e^{+j2\pi f_d \sigma} \, d\sigma$$

Let σ-nT = t, then σ-mT = t+(n-m)T and dσ = dt. Then, interchanging the order of summations and integration yields

$$\chi(\tau, f_d) = \frac{1}{NT} \sum_{n=0}^{N-1} e^{+j2\pi f_n nT} \sum_{m=0}^{N-1} e^{-j2\pi f_m[\tau - (n-m)T]} \int_{-\infty}^{\infty} e^{-j2\pi(f_n - f_m - f_d)t} \, dt \tag{5.10}$$

First we integrate correlation over one chip. Later we analyze the integration over NT. The reason will become clear very shortly.

$$\int_{\tau}^{T} e^{-j2\pi f_{nmd} t} \, dt = e^{-j2\pi f_{nmd}\tau} \, e^{-j2\pi f_{nmd}(T-\tau)} \, \frac{\sin[\pi f_{nmd}(T - \tau)]}{\pi f_{nmd}}$$

where $f_{nmd} = f_n - f_m - f_d$ for short notation.

Substituting the above result into (5.10), we obtain

$$\chi(\tau, f_d) = \frac{1}{NT} \sum_{n=0}^{N-1} e^{+j2\pi f_d nT} \sum_{m=0}^{N-1} e^{-j2\pi f_m[\tau - (n-m)T]} \, e^{-j2\pi f_{nmd}(T+\tau)} \cdot \left(1 - \frac{\tau}{T}\right) \frac{\sin\left[\pi f_{nmd} T\left(1 - \frac{\tau}{T}\right)\right]}{\pi f_{nmd} T\left(1 - \frac{\tau}{T}\right)} \tag{5.11}$$

We rewrite (5.11) as sum of two response functions; one for the auto-response function when n=m, and other cross-response when $n \neq m$.

$$\chi(\tau, f_d) = \chi_{nn}(\tau, f_d) + \chi_{nm}(\tau, f_d)$$

The auto-response function $\chi_{nn}(\cdot)$ is then

$$\chi_{nn}(\tau, f_d) = \frac{1}{N}\sum_{n=0}^{N-1} e^{+j2\pi f_d nT}\ e^{-j\pi f_n \tau}\ e^{+j\pi f_d (T+\tau)} \cdot \left(1-\frac{\tau}{T}\right)\ \frac{\sin\left[\pi f_d T\left(1-\frac{\tau}{T}\right)\right]}{\pi f_d T\left(1-\frac{\tau}{T}\right)} \tag{5.12}$$

The auto-response function reveals some interesting characteristics. First, we view the function at zero-delay Doppler cut.

$$\chi_{nn}(0, f_d) = \frac{1}{N}\sum_{n=0}^{N-1} e^{+j2\pi f_d nT}\ e^{+j\pi f_d T}\frac{\sin[\pi f_d T]}{\pi f_d T} = \frac{1}{N}\sum_{n=0}^{N-1} e^{+j\pi f_d T(2n+1)}\frac{\sin[\pi f_d T]}{\pi f_d T}$$

and

$$|\chi_{nn}(0, f_d)| = \left|\frac{\sin[\pi f_d T]}{\pi f_d T}\right| \tag{5.13}$$

Equation (5.13) is the form of familiar sin z/z shown in Figure 5.11(a). We note that Doppler resolution of this Costas-coded signal is identical to that of a LFM pulse of NT seconds.

The zero-Doppler delay cut is given by

$$\chi_{nn}(\tau, 0) = \frac{1}{N}\sum_{n=0}^{N-1} e^{-j2\pi f_n \tau}\left(1-\frac{\tau}{T}\right)$$

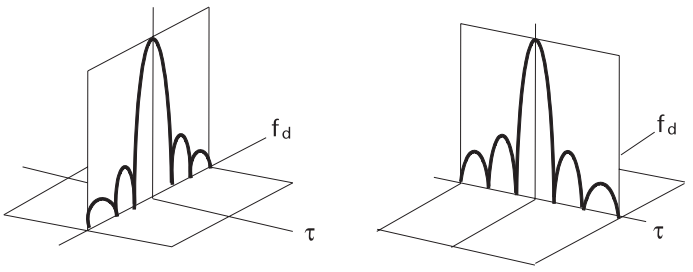


Figure 5.11 (a) Zero-delay Doppler cut. (b) Zero-Doppler delay cut.

Since the summation of f_n in proper sequence is immaterial, we can write the above expression as

$$\chi_{nn}(\tau,0) = \frac{1}{N}\sum_{n=0}^{N-1} e^{-j2\pi f_n \tau}\left(1-\frac{\tau}{T}\right)$$

$$= \frac{1}{N} e^{-j\pi(N-1)\tau}\left(1-\frac{\tau}{T}\right)\frac{\sin[\pi N\tau]}{\sin(\pi\tau)} \tag{5.14}$$

and $$|\chi_{nn}(\tau,0)| = \frac{1}{N}\left(1-\frac{|\tau|}{T}\right)\left|\frac{\sin(\pi N\tau)}{\sin(\pi\tau)}\right|. \qquad \text{See Appendix 5A.}$$

Figure 5.11(b) indicates that the range resolution of Costas-code is much higher than that of a rectangular pulse of NT long with no modulation. The width of the main lobe is as narrow as a compressed pulse of LFM. By combining Figure 5.11(a, b), we can visualize that the auto-response surface is peaked at the origin and decreases as $|\tau|$ and $|f_d|$ increased, a thumbtack-like without being sheared off the principal axis as in LFM.

Next we derive an expression for the cross-response function where the integration of Equation (5.10) must be from τ to NT, and $n \neq m$. Following the steps we had taken in the case of auto-response, we obtain,

$$\chi_{nm}(\tau,f_d) = \frac{1}{N(N-1)}\sum_{n=0}^{N-1} e^{+j2\pi f_d nT}\sum_{m=0\ m\neq n}^{N-1} e^{-j2\pi f_m[\tau-(n-m)T]}$$

$$\cdot\ e^{-j\pi f_{nmd}(NT+\tau)}\left(1-\frac{\tau}{NT}\right)\frac{\sin\left[\pi f_{nmd}NT\left(1-\frac{\tau}{NT}\right)\right]}{\pi f_{nmd}NT\left(1-\frac{\tau}{NT}\right)} \tag{5.15}$$

Two principal cuts of the cross-response function are difficult to visualize, for two exponential summations have different summing indices and their phases would interact. The cross-response function with $f_d = 0$ is given by

$$\chi_{nm}(\tau,0) = \frac{1}{(N-1)}\sum_{m\neq n}^{N-1} e^{-j2\pi f_m[\tau-(n-m)T]}\ e^{-j\pi f_{nm}(NT+\tau)}$$

$$\cdot\ \left(1-\frac{\tau}{NT}\right)\frac{\sin[\pi f_{nm}NT]}{\pi f_{nm}NT} \tag{5.16}$$

and with $\tau = 0$ is given by

$$\chi_{nm}(0,f_d) = \frac{1}{N(N-1)}\sum_{n=0}^{N-1} e^{+j2\pi f_d nT}\sum_{m\neq n}^{N-1} e^{+j2\pi f_m(n-m)T}$$

$$\cdot\, e^{-j2\pi f_{nmd}NT}\frac{\sin\left[\pi f_{nmd}NT\left(1-\frac{\tau}{NT}\right)\right]}{\pi f_{nmd}NT\left(1-\frac{\tau}{NT}\right)} \tag{5.17}$$

An examination of $\chi_{nm}(\tau, 0)$ reveals that the zero-Doppler delay cut will be zero when τ is zero or ± an integer since f_{nm} is always an integer; therefore, the argument of sine function is ± an integer multiple of π.

An examination of $\chi_{nm}(0, f_d)$ reveals that the zero-delay Doppler cut will peak when $|f_n - f_m| = f_d$ and fall to zero when f_d is either zero or ± an integer.

We conclude that the cross-response function would either peak or fall to zero at the integer multiple points on the τ–f_d plane. The locations of peaks can be easily determined by a transparent overlay on the original time-frequency matrix. By shifting the overlay to the right (or to the left) along the delay axis or shifting the overlay up (or down) along the Doppler axis in increment demonstrated by [5], we will find the locations of the peaks. An example of the shifting and locating the peaks is shown in Figure 5.12. There are N(N-1) minor peaks.

The auto-response function is confined to the inner square where the cross-response function has a near zero level, indicated by dark lines. Two principal cuts of the cross-response functions, (5.16) and (5.17) are shown in Figure 5.13(a, b) with magnified vertical scales.

We observe that the gain of the Costas-coded signal is N and that the ambiguity surface resembles a thumbtack with a few sidelobes spikes around the central main lobe (see Figure 5.14). The average height of the clutter pedestal is 1/N, and a few spikes reach 2/N. In order to reduce the height of the spikes of the sidelobes we have to increase the number of subpulses N.

A compound scheme of frequency plus phase code sequence has produced moderate success [6] in reducing the sidelobe levels. A width modulation subpulses,

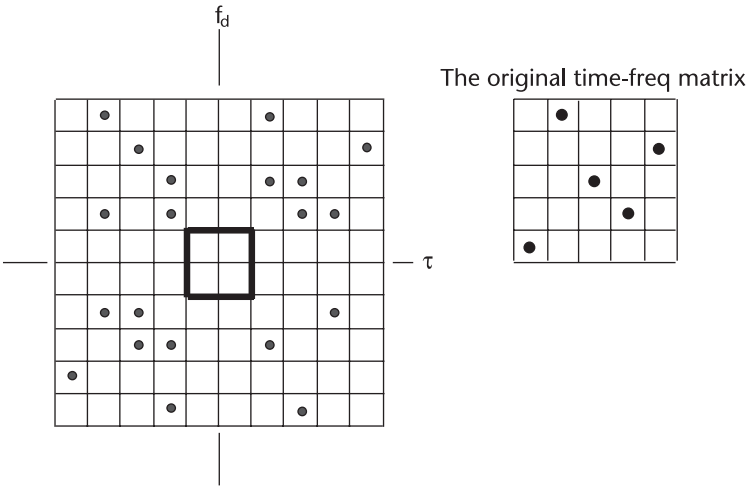


Figure 5.12 Locations of cross-ambiguity peaks.

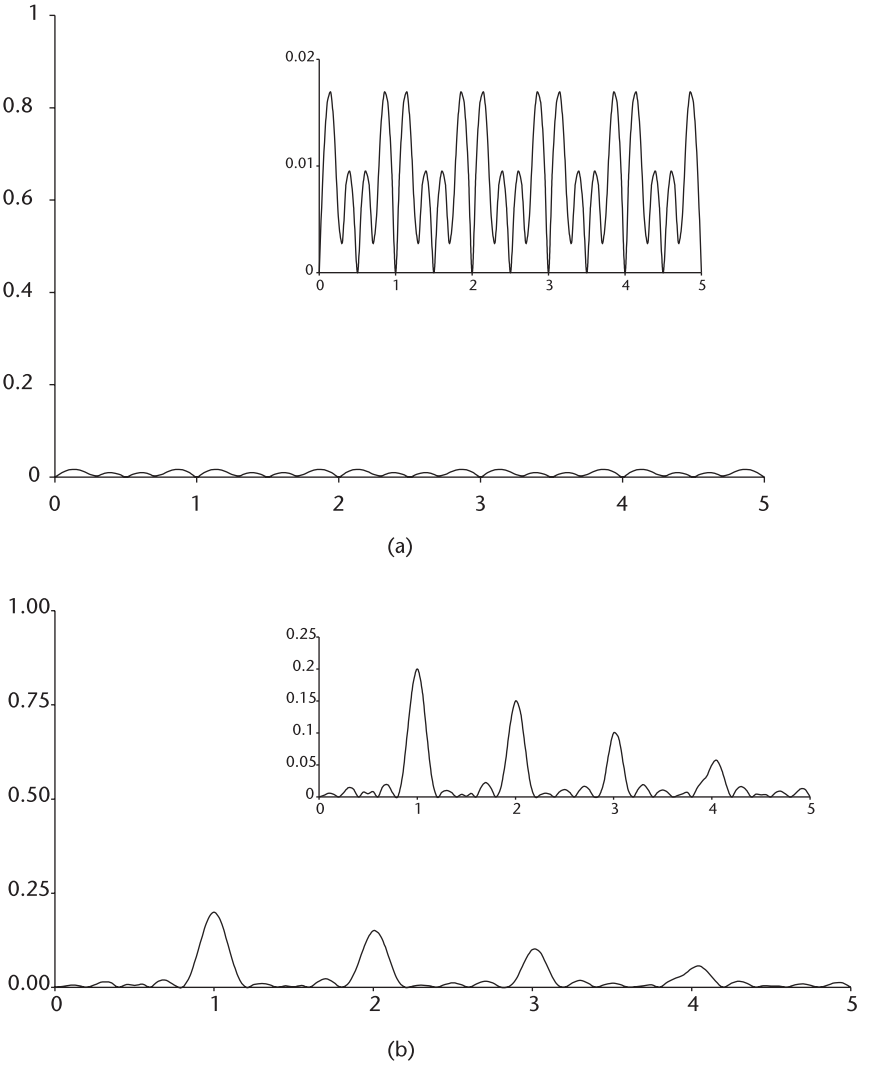


Figure 5.13 (a) Zero-Doppler delay cut. (b) Zero-delay Doppler cut.

unequal pulsewidth control should be entertained. Polarization modulation on the subpulses is another consideration.

The original object of Costas was underwater target detection where the phase coherency is impossible to maintain and the clutter environment is extremely severe. Costas-coded frequency hopping is a near optimum waveform for a thumbtack-like ambiguity function (Figure 5.15). The search for a better waveform design should continue [7].

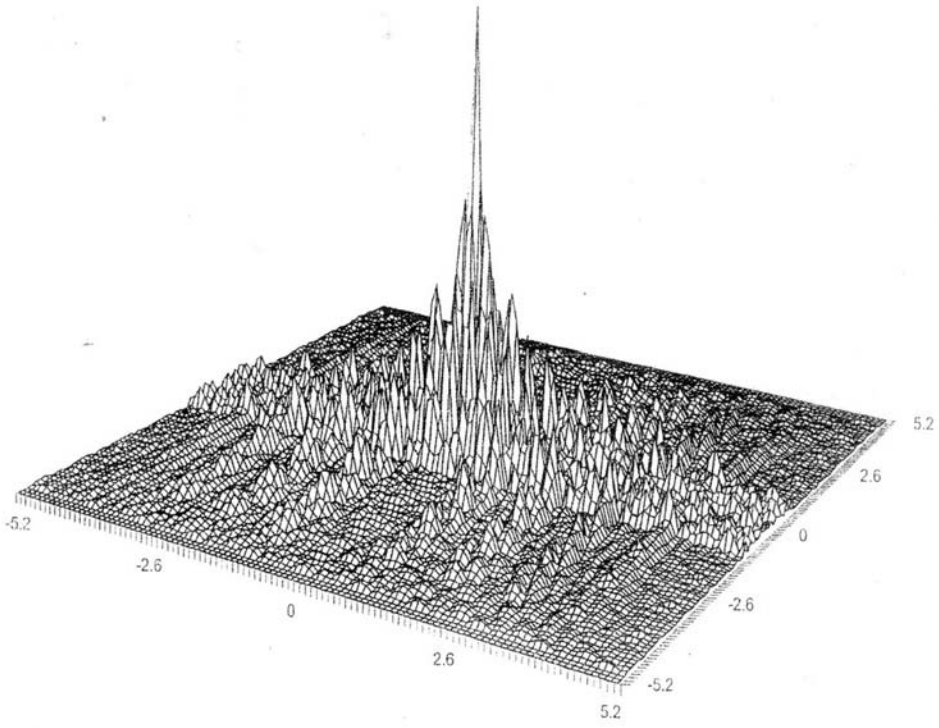


Figure 5.14 Ambiguity surface of Costas-coded signal, N=5.

List of Progams

Program	*Feature*
(1) AMB_RECT.CPP	Ambiguity function of a rectangular pulse with single constant frequency
(2) AMB_CLFM.CPP	Ambiguity function of a rectangular pulse with coherent linear frequency modulation
(3) X(T_0)FM.CPP	Zero-Doppler delay cut of CLFM
(4) X(0_F)FM.CPP	Zero-delay Doppler cut of CLFM
(5) Xnm(T_0).CPP	Cross-response function, zero-Doppler delay cut
(6) Xnm(0_F).CPP	Cross-response function, zero-delay Doppler cut
(7) AMB_COST.CPP	Ambiguity function of Costas-coded signal

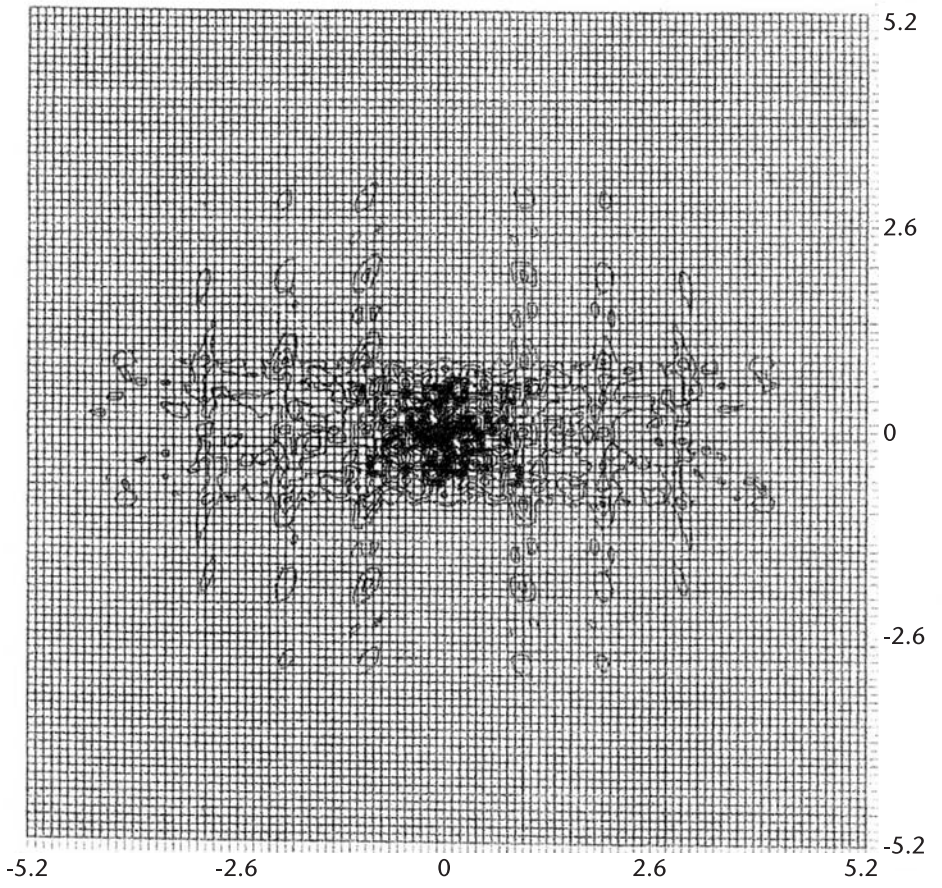


Figure 5.15 Contour map, Costas-coded signal.

References

[1] Woodward, P. M., *Probability and Information Theory with Application to Radar*, London: Pergamon Press, 1953, and Dedham, MA: Artech House, 1980.

[2] Nathanson, F. E., *Radar Design Principle, Signal Processing, and Environment*, New York, NY: McGraw-Hill, 1969, pp. 309–310.

[3] Costas, J. P., "A Study of a Class of Detection Waveform Having Nearly Ideal Range-Doppler Ambiguity Properties," *Proc. IEEE*, Vol. 72, No. 8, Aug. 1984.

[4] Colomb, S.W., and H. Taylor, "Constructions and Properties of Costas Arrays," *Proc. IEEE*, Vol. 72, No. 9, Sept. 1984.

[5] Levanon, N., *Radar Principles*, New York. NY: John Wiley & Sons, 1988.

[6] Freedman, A., and N. Levanon, "Staggered Costas Signals," *IEEE Trans*., Vol. AES-22, No. 6, Nov. 1986.

[7] Levanon, N., and E. Mozeson, *Radar Signals,* IEEE Press & Wiley Interscience, 2004.

Selected Bibliography

Rihaczek, A. W., *Principles of High Resolution Radar,* New York, NY: McGraw-Hill, 1969, and Norwood, MA: Artech House, 1996.

Appendix 5A

1. $$\sum_{n=0}^{N-1} x^n = \frac{1- x^N}{1- x} = \frac{x^{N/2}}{x^{1/2}} \frac{x^{N/2} - x^{-N/2}}{x^{1/2} - x^{-1/2}}$$

   Let $x = e^{+j2\pi\tau}$

   $$\Sigma\, e^{+j2\pi m\tau} = \frac{e^{j\pi N\tau}}{e^{j\pi\tau}} \quad \frac{e^{j\pi N\tau} - e^{-j\pi N\tau}}{e^{j\pi\tau} - e^{-j\pi\tau}} = e^{\,j\pi(N-1)\tau} \frac{\sin(\pi N\tau)}{\sin(\pi\tau)}$$

2. $\left.\frac{\sin(\pi N\tau)}{\sin(\pi\tau)}\right|_{\tau=0}$ at $\tau = 0$ is obtained by L'Hospital rule:

   $$\left.\frac{\sin(\pi N\tau)}{\sin(\pi\tau)}\right|_{\tau=0} = \left.\frac{\cos(\pi N\tau)\,\pi N}{\cos(\pi\tau)\,\pi}\right|_{\tau=0} = N$$

CHAPTER 6

Array Antennas

6.1 Introduction

In this chapter, we study the basic principle of array antenna synthesis techniques and show the radiation patterns of several array antennas.

An array antenna may be linear, circular, or elliptical (rectangular). The array antenna may consist of fewer than ten elementary radiating sources arranged in a line or tens to thousands arranged on a planar grid.

The elementary radiator may be a slot on the wall of a rectangular waveguide, a dipole or crossed-dipole on a ground plane, a rectangular pyramidal horn, or an open-ended waveguide. These elementary radiating sources can be arranged in a linear array, a circular aperture, or an elliptical (or rectangular) aperture.

Three examples of the elementary radiator are shown in Figures 6.1 through Figure 6.3 [1–4]. The radiation pattern of an array antenna is the product of the elementary radiator and the array pattern.

6.2 Linear Array

A linear array antenna is constructed by arranging the radiating elements on a line. The powers radiating from the elements may be equal or different from one another by design.

To begin our design of a linear array, we take a pair of isotropic radiating point sources arranged as shown in Figure 6.4, and derive the field strength at point P.

A phase delay of a wave radiating from A on the left with respect to that on the right is,

$$L = d \sin \quad , \frac{2}{} = \frac{}{L}$$

$$= \frac{2 \ L}{} = \frac{2 \ d}{} \sin$$

where λ is the wavelength of transmitted carrier frequency in the propagating medium. The effective field strength at point P will be the vectorial sum of radiations from the pair.

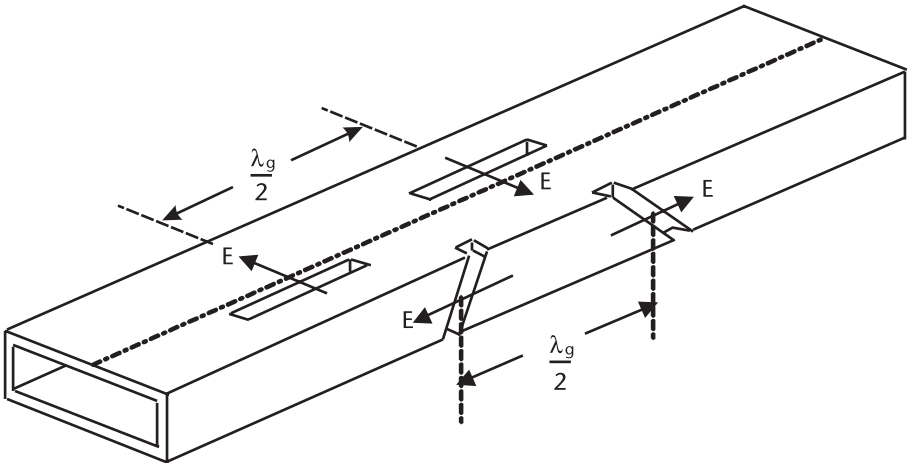
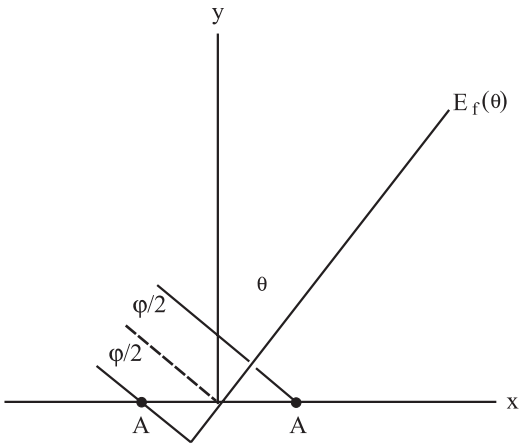


Figure 6.1 Radiating slots on the wall of a rectangular waveguide, off-centered longitudinal slot on the broad wall, or inclined slot on the narrow wall.



$$E_f(\theta) = A\, e^{-j\varphi/2} + A\, e^{j\varphi/2} = 2A\cos\left(\frac{\varphi}{2}\right) = 2A\cos\left(\frac{\pi d}{\lambda}\sin\theta_0\right) \tag{6.1}$$

When we have multiple pairs, the effective field strength would be the total vectorial sums of all pairs.

$$\begin{aligned} E_f(\theta) &= 2A_0\cos\left(\frac{\pi d}{\lambda}\sin\theta_0\right) + 2A_1\cos\left(\frac{3\pi d}{\lambda}\sin\theta_1\right) + 2A_2\cos\left(\frac{5\pi d}{\lambda}\sin\theta_2\right) + \cdots \\ &= \sum_{i=0}^{N-1} 2A_i\cos\left[\frac{(2i+1)\pi d}{\lambda}\sin\theta_i\right] \end{aligned} \tag{6.2}$$

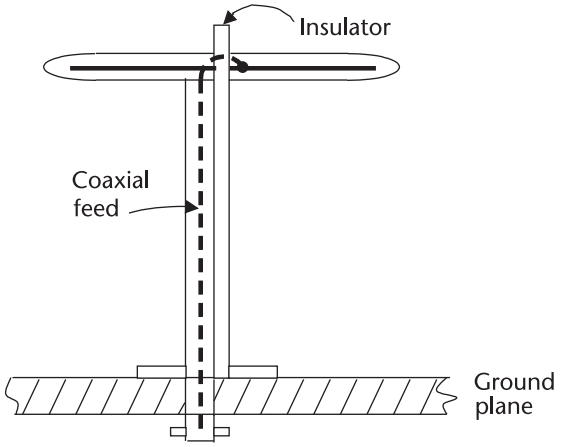


Figure 6.2 Dipole radiator on ground plane.

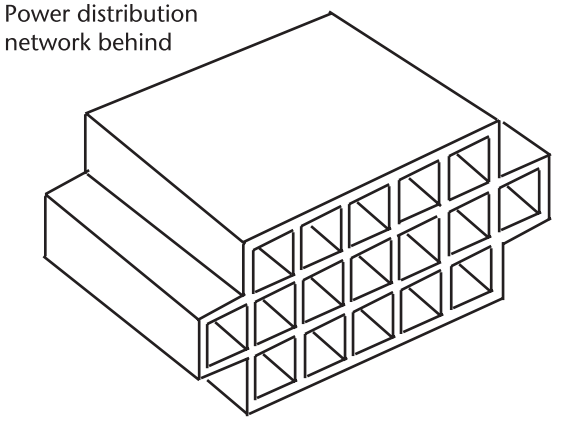


Figure 6.3 Open-ended square waveguides stacked to form an array. The square waveguides may be replaced by rectangular waveguides, or pyramidal horns.

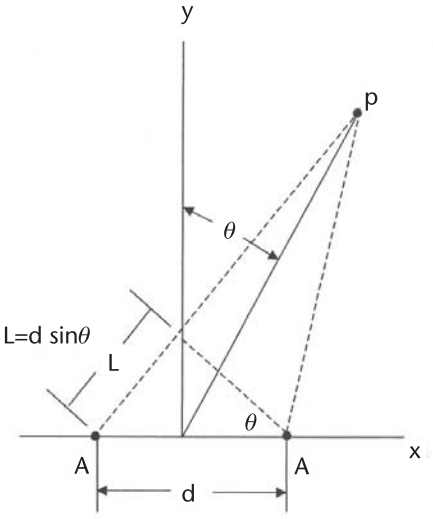


Figure 6.4 A pair of isotropic point sources.

where N is the number of element pairs in the array. Point P is far out with respect to the dimension of the linear array so that angle θ_i is assumed to be equal for all pairs. The far-field pattern would be the product of the element pattern and array pattern.

$$E_f = E_i \, E_a$$

where

E_i: field pattern of individual pair

E_a: field pattern of array aperture

When the powers radiated from the pairs are all equal in (6.2),

$$A_1 = A_2 = \ldots = A_n$$

we call the array a "uniform" array. The magnitudes of A_i can be made equal to the coefficients of certain polynomials such as Chebyshev, Taylor (a modified Chebyshev), or function such as Lambda, Hamming, and many more [5–8]. We study a few of them.

The radiation patterns of a linear array with 14 pairs of isotropic point sources are programmed in ANT_LINE.CPP. The program selects the element excitation amplitude distribution from the following set:

(1) Uniform;
(2) Chebyshev;
(3) Taylor ($\bar{n} = 5$).

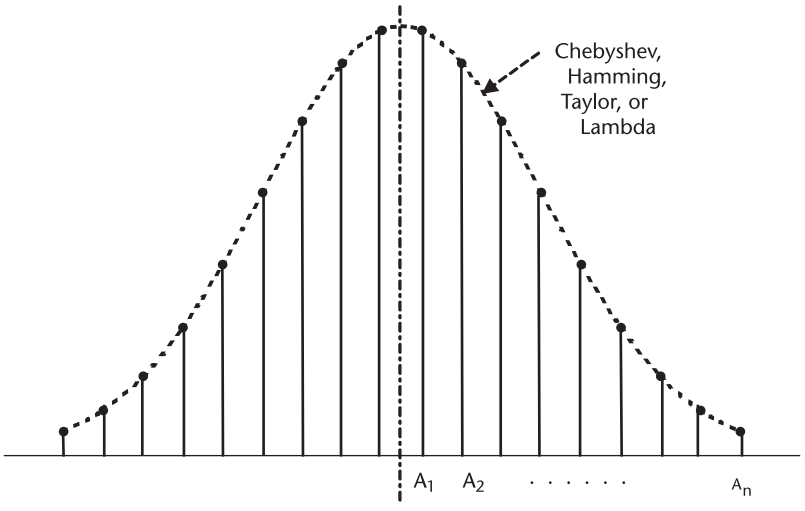


Figure 6.5 Element excitation amplitude distribution.

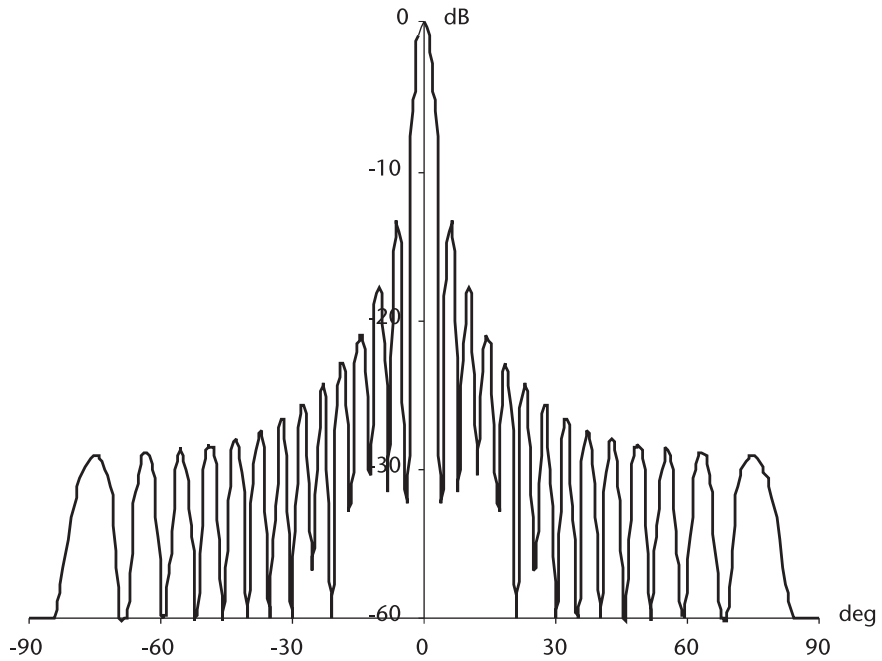


Figure 6.6 Linear array, uniform.

Different excitation amplitude distributions produce unique patterns. They are shown in Figures 6.6 through 6.8.

The uniform distribution produces the highest gain and the narrowest beamwidth, but very poor sidelobe levels. The Chebyshev distribution produces equal sidelobe levels. The sidelobe level is a design parameter of the Chebyshev distribution [5]. The Taylor distribution is an interesting one. The closest pairs of the sidelobes have an approximately equal sidelobes but the next sidelobe pair is slightly below it, the next pair out are further lower, and so on [6–8].

The Table 6.1 summarizes the three linear arrays.

The gain of Taylor excitation distribution is higher than Chebyshev can be seen when we compare Figure 6.7 and 6.8. The Chebyshev distribution radiates more power through the sidelobes than the Taylor distribution does. This is always true when the 3-dB beamwidth is equal or nearly equal within a one-tenth of degree. The difference in gains between the two distributions will diminishes when the highest sidelobe is reduced below –30 dB or lower and simultaneously increased the $\bar{n}$ of the Taylor distribution.

Certainly an array design is a compromise between the gain, beamwidth, sidelobe level and the rate of decrease of the sidelobe levels.

Table 6.1

	Uniform	*Chebyshev*	*Taylor*
Maxim gain (dB)	14.47	14.14	14.32
Loss (dB) 3 dB	—	0.33	0.15
Beamwidth (deg)	3.6	3.8	3.9

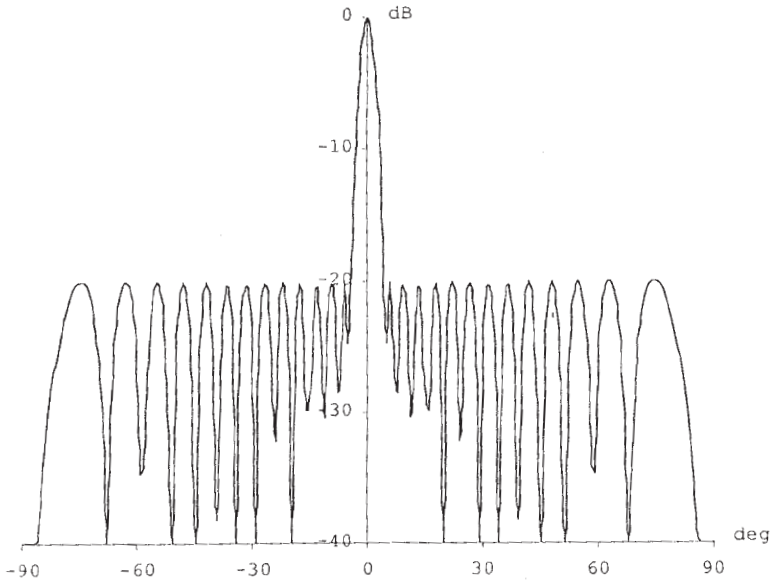


Figure 6.7 Linear array, Chebyshev, –20-dB sidelobes.

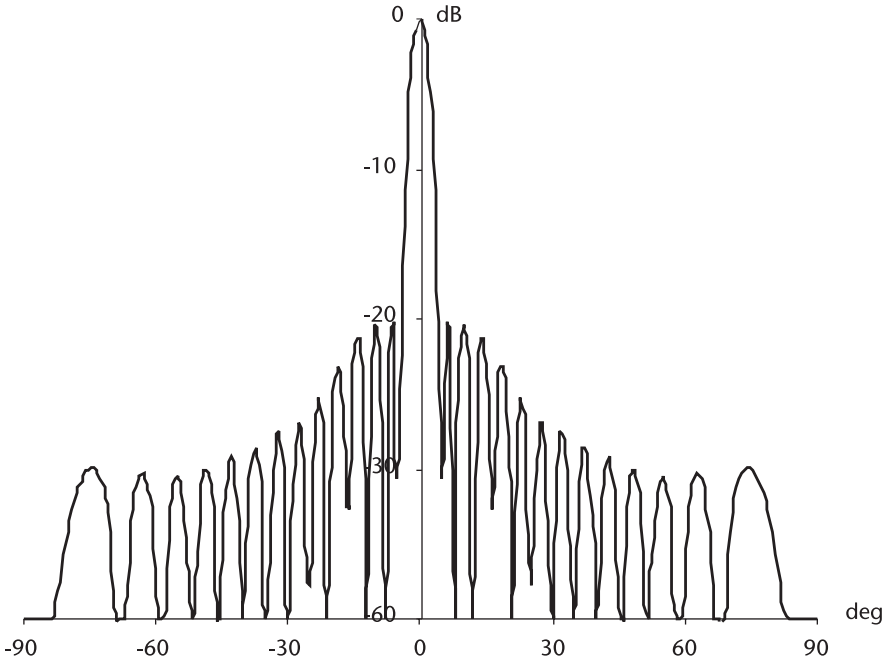


Figure 6.8 Linear array, Taylor, –20-dB sidelobe.

The maximum gain is given by [1], Equation (5.4.2).

$$D = \left(\sum_{-N}^{N} I_n\right)^2 \Big/ \sum_{-N}^{N} {I_n}^2 = \left(2\sum_{0}^{N-1} I_n\right)^2 \Big/ 2\sum_{0}^{N-1} {I^2}_n$$

(odd) (even)

The odd number of radiators in an array is awkward when a differential pattern is desired, for the central element in an array interferes the differential pattern. Readers should be aware of the difference between the directivity and gain. Both of them are dimensionless, pure numeric. The maximum directivity is defined as the power density, watt/m^2, in the boresight direction divided by the power density averaged over all directions. The directivity and gain differs when the power received from the transmitter is absorbed by antenna structure. A slotted waveguide array is usually assumed to be lossless, hence directivity is equal to gain.

A horizontal linear array is usually covered with a flared horn in order to control the vertical radiation pattern. A sketch of such an antenna is shown with inclined slots on the narrow wall of a rectangular waveguide.

6.3 Circular Aperture Array

The radiating elements may be the slots in waveguides, dipoles, or crossed dipoles on a ground plane, open-ended squares or rectangular wave-guides, pyramidal horns, and they are arranged to form a circular aperture.

The powers radiated from the elements may be equal or different from the adjacent elements by design. The radiating power of an element at (m, n) A_mA_n, is a product of distribution functions on the x-axis and y-axis as shown in Figure 6.10, the principle of separable excitation distributions.

Those elements outside the radius are deleted (excluded) from the pattern computation. See Figure 6.11 below and Figure 6.A.1 at the end of this chapter.

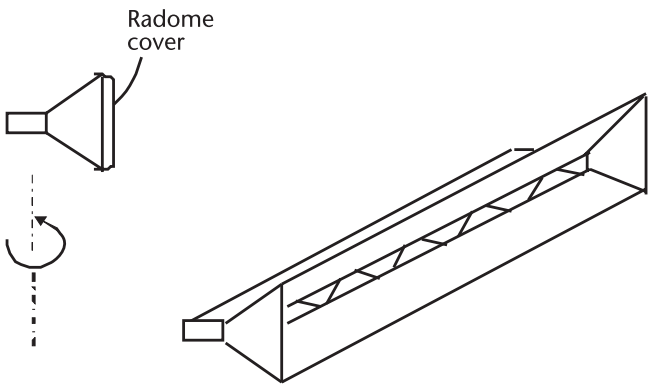


Figure 6.9 Line array antenna with flaredhorn.

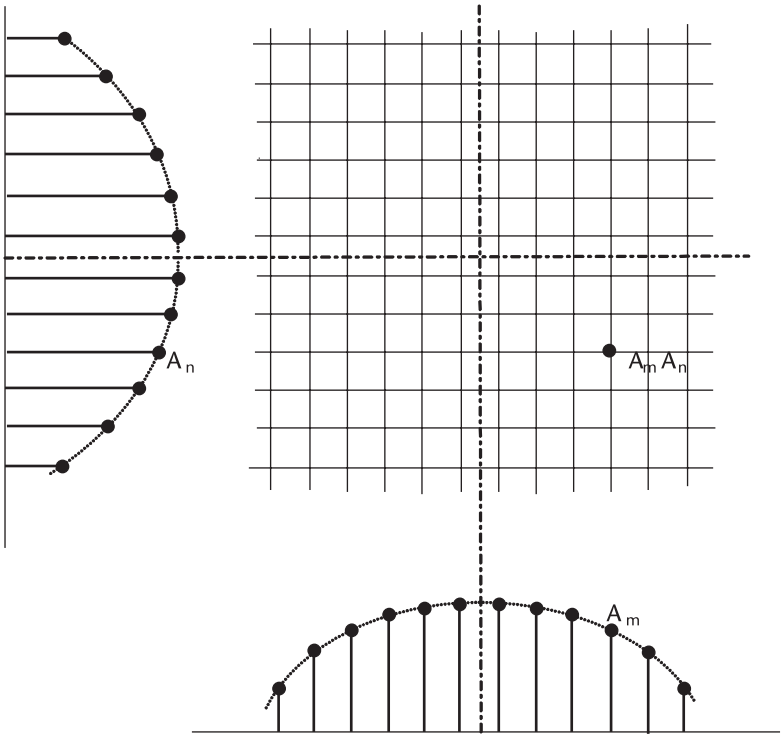


Figure 6.10 Elementary excitation amplitude distribution.

The effective field strength at point P contributed by the element at A_mA_n is shown in Figure 6.12.

The field strength at P, located at far distance compared with the dimension of the aperture is given by,

$$E_f(\theta,\varphi) = \sum_{m=1}^{M}\sum_{n=1}^{N} A_mA_n \cos\left(\frac{m\pi d_x}{\lambda}\sin\theta\cos\varphi\right)\cos\left(\frac{n\pi d_x}{\lambda}\sin\theta\sin\varphi\right) \qquad (6.3)$$

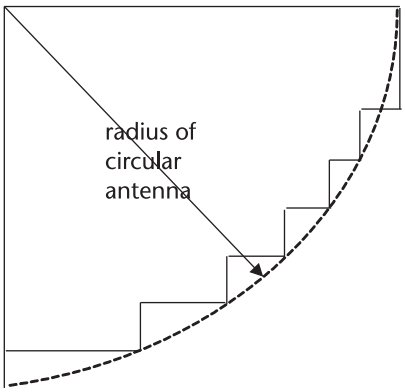


Figure 6.11 Circular aperture array by deleting elements outside of radius. See Figure 6.A.1 for details.

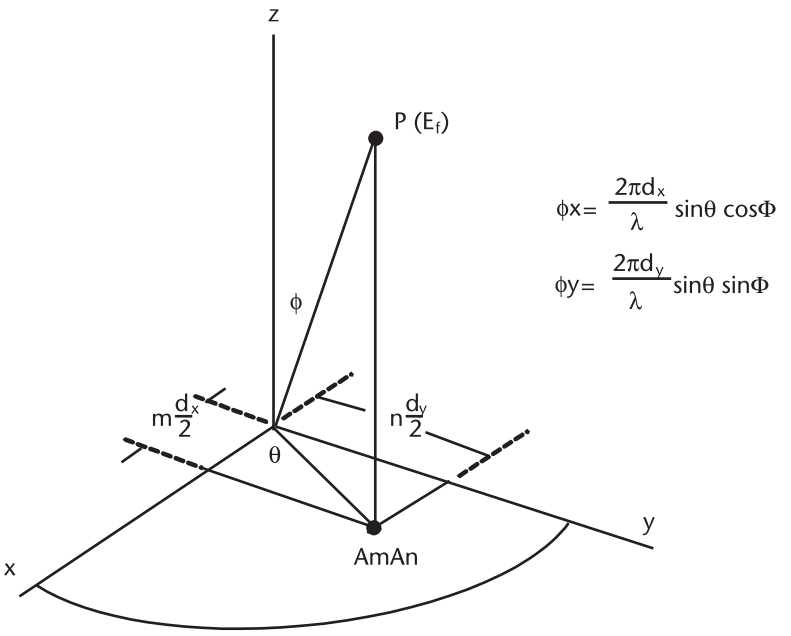


Figure 6.12 Phase of an Element A_mA_n in Spherical Coordinates.

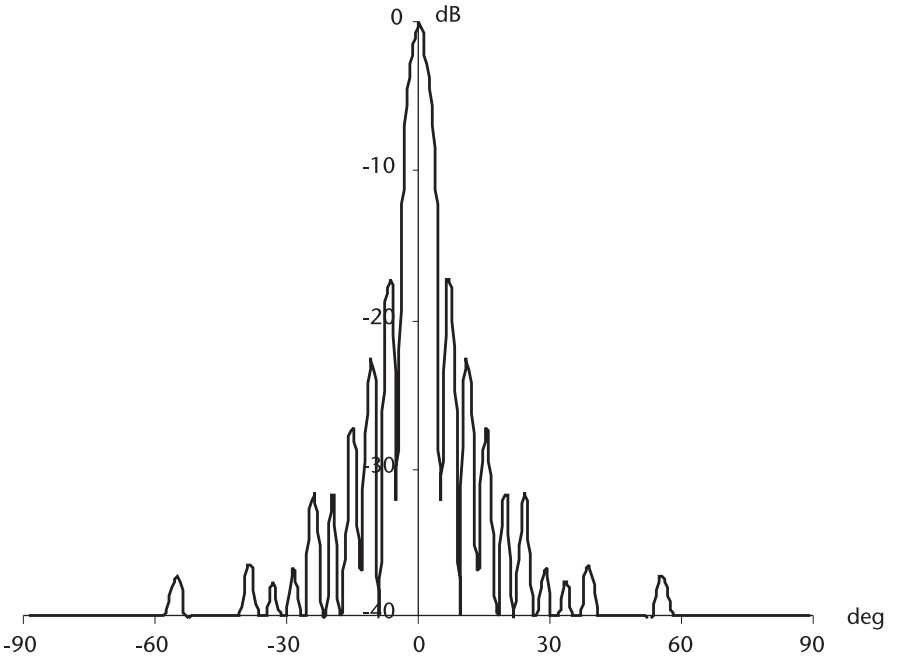


Figure 6.13 Circular aperture, uniform–uniform.

Table 6.2 summarizes the results.

Table 6.2

	Uniform–Uniform	*Chebyshev–Chebyshev*	*Taylor–Taylor*
Max grain (dB)	28.06	*	26.24
Gain Loss	—	*	1.82
Beamwidth (degrees)	4.1	*	4.4

* Read the closing remark at the end of this chapter.

where M and N are the number of elements in x and y coordinates, and d_x and d_y are the spacing in x and y coordinates.

We compute the radiation patterns of the following three primary distributions:

(1) Uniform–uniform;
(2) Chebyshevz–Chebyshev;
(3) Taylor–Taylor ($\bar{n} = 5$).

There are three other combinations of two distributions when we consider different distributions on x- and y-axis.

(1) Uniform–Chebyshev;
(2) Uniform–Taylor;
(3) Chebyshev–Taylor.

A circular aperture is symmetric; therefore, there are three combinations only. No permutation is necessary. For an elliptical aperture the permutations should be considered for the beamwidth control in the vertical and horizontal planes.

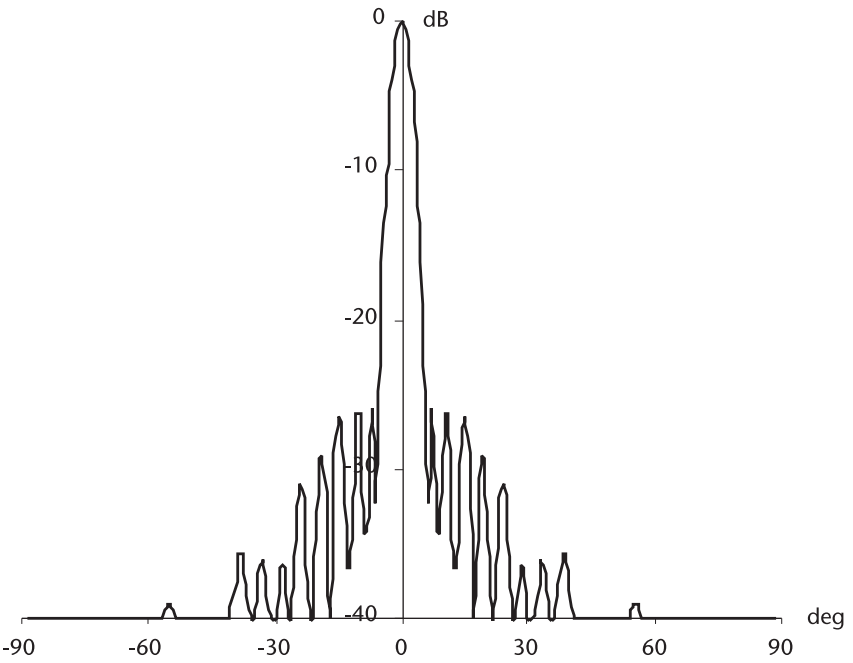


Figure 6.14 Circular aperture, Taylor–Taylor.

The radiation patterns are computed in ANT_CIR.CPP for the primary distributions, shown in Figures 6.13. and 6.14.

6.4 Elliptical Aperture Array

The radiating elements may be arranged to form an elliptical (or rectangular) aperture. Figure 6.15 shows an example of such an array antenna. The element excitation distributions could be the coefficients of polynomials or a selected function. They may be equal or different for *x*- and *y*-axes. For a rectangular aperture array, ANT_ELIP.CPP has a selection command in the programs.

We shall compute the radiation patterns of the following three primary distributions. There are six more combinations of the functions mentioned previously. As with the circular array, two separable linear distribution functions have been applied to *x*-axis and *y*-axis, and elements deleted to form the elliptical aperture (see Figure 6.A.2 at the end of this chapter).

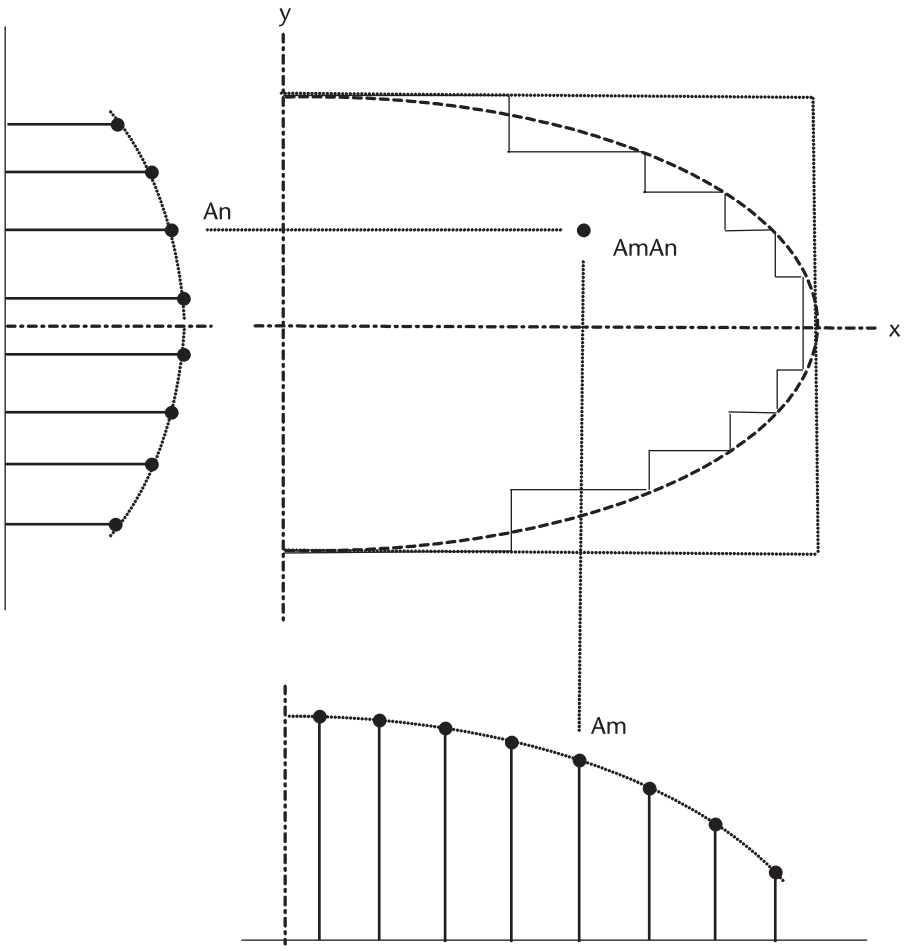


Figure 6.15 Elliptical (or rectangular) aperture array, two separable excitation distribution functions.

(1) Uniform–uniform;
(2) Chebyshev–Chebyshev;
(3) Taylor–Taylor ($\bar{n} = 5$).

The effective field strength at point P in the far field is given by, identical to (6.3).

$$E_f(\theta,\varphi) = \sum_{m=1}^{M}\sum_{n=1}^{N} A_m A_n \cos\left(\frac{m\pi d_x}{\lambda}\sin\theta\ \cos\varphi\right)\cos\left(\frac{n\pi d_x}{\lambda}\sin\theta\ \cos\varphi\right)$$

The radiation patterns are programmed in ANT_ELIP.CPP by selecting the excitation function by command. Two of them are shown in Figures 6.16 and 6.17.

Two radiation patterns are shown, one with $\varphi = 0^o$ and another $\theta = 90^o$. Intermediate patterns can be obtained by changing declaration in the program.

Table 6.3 summarizes the elliptical aperture array antennas.

6.5 Monopulse Array Antenna

The word monopulse is derived from an antenna's capability to detect target position in azimuth and elevation planes simultaneously by a single pulse. A monopulse antenna is an essential component of a tracking radar system. A angle error signal from the difference channels would drive a servomechanism (mechanical or electronic) to track a target. Phased array antennas have virtually eliminated the mechanical servo-structure.

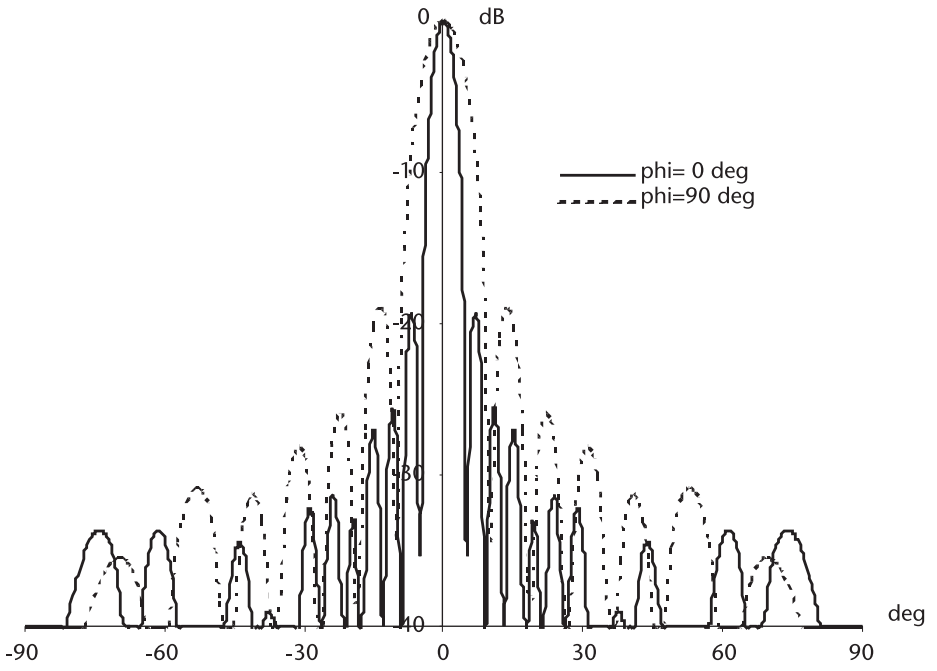


Figure 6.16 Elliptical array, uniform–uniform.

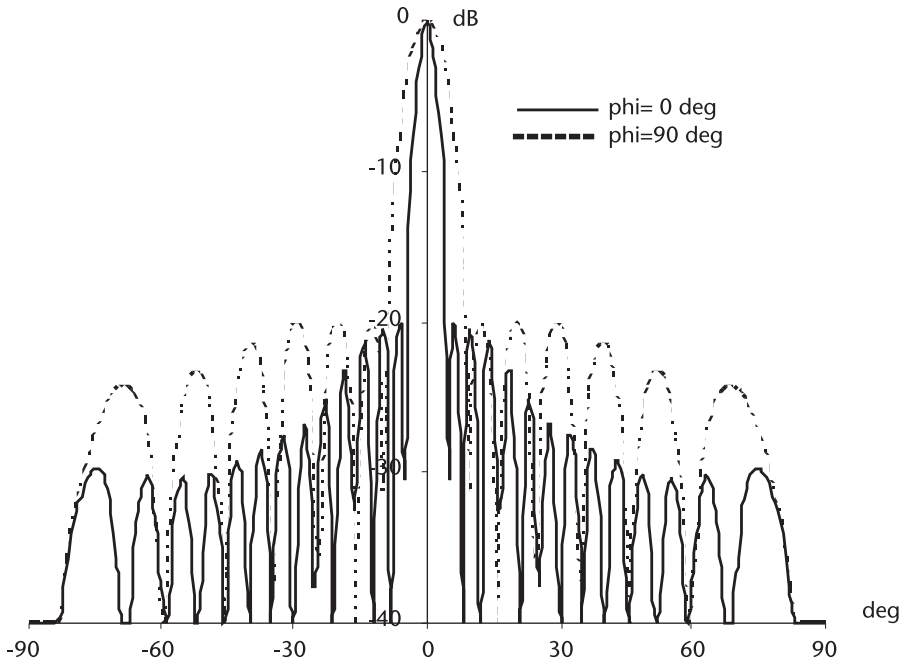


Figure 6.17 Elliptical array, Taylor–Taylor.

The principle of the monopulse antenna can be visualized by Figure 6.18. Imagine that an antenna aperture is divided into four separated subapertures and connected as shown.

The sum channel is connected to the transmitter/receiver through a circulator-isolator-limiter combination as in an ordinary radar front end. The radiation patterns of the sum channel have been discussed in Sections 6.3 and 6.4, circular or elliptical. The response of difference channels can be analyzed by pairs of two point sources as shown in Figure 6.19.

Note the difference between (6.2) and (6.4). The former is the response of difference channel and the latter is that of sum channel. The response of azimuth difference channel and that of elevation difference channel is given by

Table 6.3

	Uniform–Uniform	*Chebyshev–Chebyshev*	*Taylor–Taylor*
Max grain (dB)	24.83	Δ	23.86
Loss	—	Δ	0.97
Beamwidth	*5.3		*5.7
3dB (deg)	** 8.6		** 8.8

Δ Read the closing remark at the end of this chapter.
* Beamwidth parallel to the major axis.
** Beamwidth parallel to the minor axis.

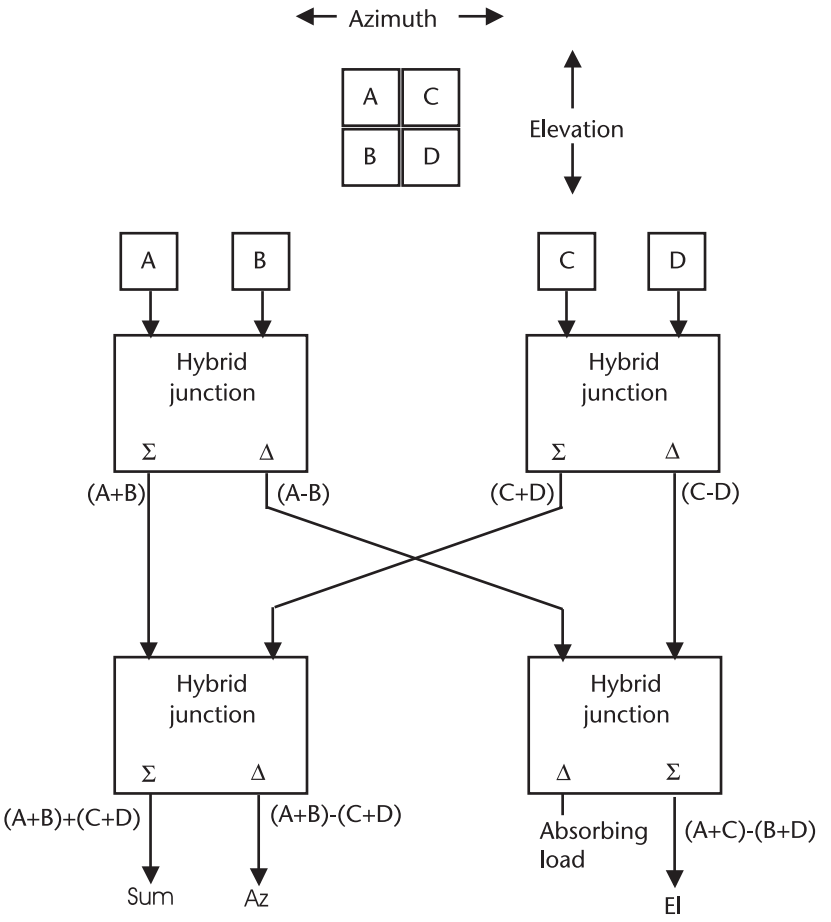


Figure 6.18 Basic monopulse array antenna structure.

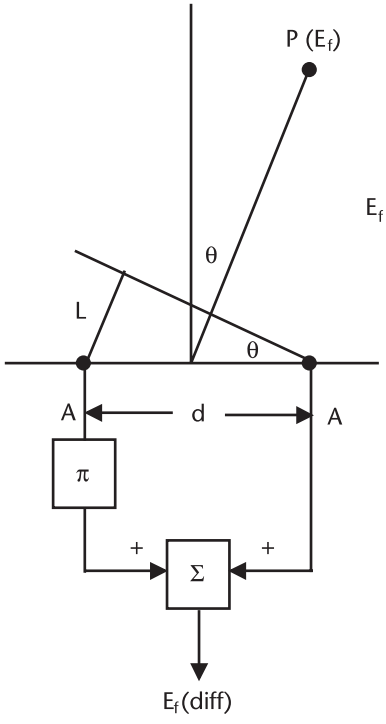
$$E_{f(az\ diff)} = \sum_{m=1}^{M}\sum_{n=1}^{N} A_m A_n \sin\left(\frac{m\pi d_x}{\lambda}\sin\theta\cos\varphi\right)\cos\left(\frac{n\pi d_y}{\lambda}\sin\theta\sin\varphi\right) \quad (6.4)$$

$$E_{f(el\ diff)} = \sum_{m=1}^{M}\sum_{n=1}^{N} A_m A_n \cos\left(\frac{m\pi d_x}{\lambda}\sin\theta\cos\varphi\right)\sin\left(\frac{n\pi d_y}{\lambda}\sin\theta\sin\varphi\right) \quad (6.5)$$

where A_mA_n are the element distributions used for sum channel. For a circular array the response of azimuth channel would be identical to that of elevation channel, but rotated 90° about the beam axes, by a trigonometric identity,

$$\sin\left(\quad + \frac{}{2}\right) = \sin\quad \cos\frac{}{2} + \cos\quad \sin\frac{}{2} = \cos$$

provided the excitation amplitude distributions for both axes are the same. If not we expect the responses of two channels would be different from one another.



$$L = d\sin\theta, \quad \frac{\lambda}{L} = \frac{2\pi}{\varphi}$$

$$\varphi = \frac{2\pi L}{\lambda} = \frac{2\pi d}{\lambda}\sin\theta$$

$$E_f(\text{diff}) = A\exp\{-j\frac{\varphi}{2}\}\exp\{j\pi\} + A\exp\{j\frac{\varphi}{2}\}$$

$$= A[\exp\{j\frac{\varphi}{2}\} - \exp\{-j\frac{\varphi}{2}\}]$$

$$= j2A\sin\frac{\varphi}{2}$$

$$= j2A\sin(\frac{\pi d}{\lambda}\sin\theta)$$

Figure 6.19 Difference channel response.

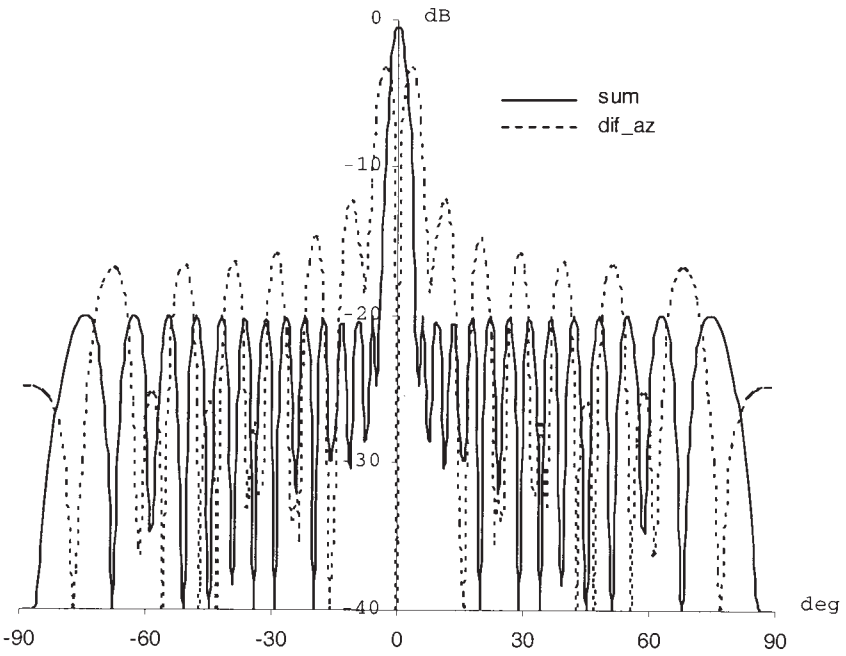


Figure 6.20 Monopulse, linear, Chebyshev.

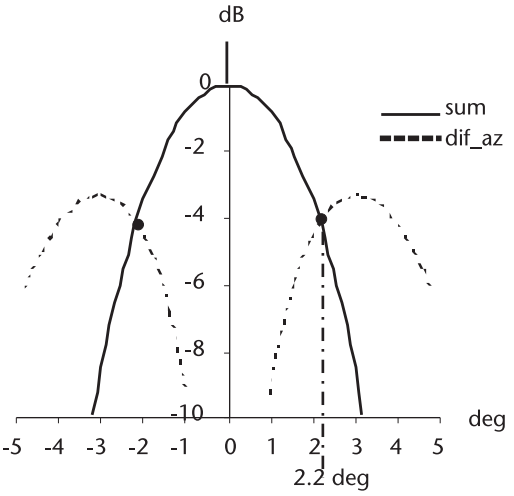


Figure 6.21 Crossover angle, linear, Chebyshev.

We programmed the sum and difference channels patterns for various excitation distributions for linear, circular and elliptical array. Three programs are written.

(1) MONO_LIN.CPP
(2) MONO_CIR.CPP
(3) MONO_ELP.CPP

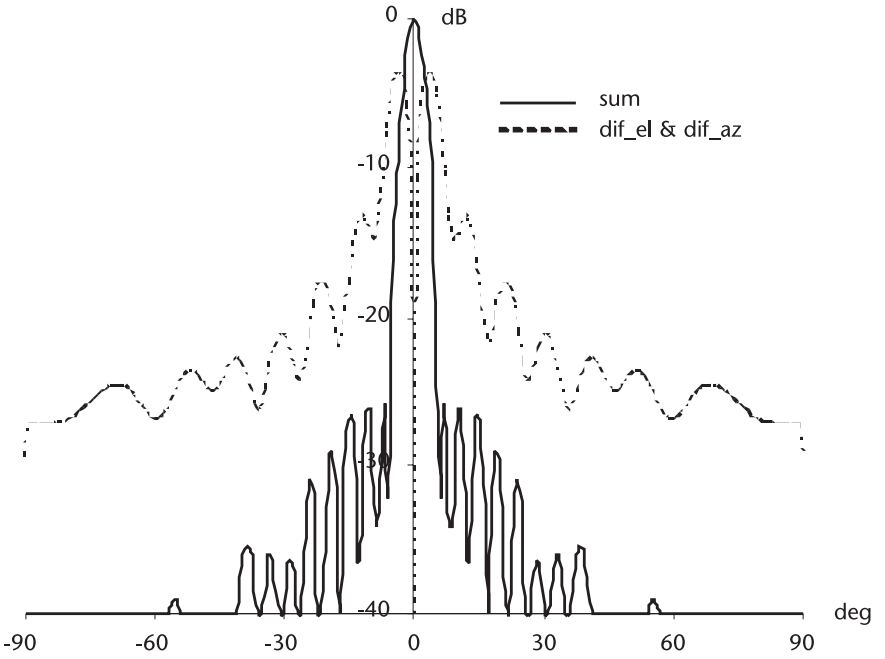


Figure 6.22 Monopulse, circular, Taylor.

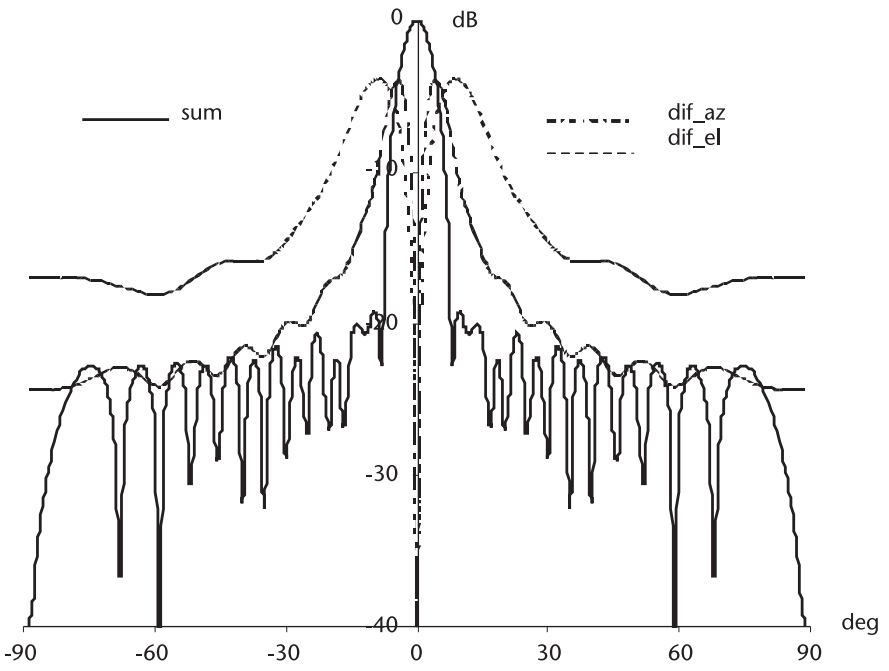


Figure 6.23 Monopulse, elliptic, Hamming.

and the results are shown in Figures 6.20 through 6.23. For a monopulse antenna the crossover angle between the sum and difference channel is a parameter to be considered seriously: the smaller crossover angle for a higher resolution tracking, a larger crossover angle for a higher acquisition probability.

There are a half dozen different signal processing techniques, amplitude comparison, phase comparison or combined one, to extract the error signal to steer the antenna [10].

There are two different crossover angles in elliptical monopulse antenna, one for $\varphi = 0°$, and another for $\varphi = 90°$.

6.6 Conclusion

1. Our analysis and computation of radiation patterns are based on the assumption that the radiating element is a point source, a hemispherical radiator on a very large ground plane, and the E-field and the H-field are identical in magnitude.

   Slots on the rectangular waveguide, dipoles, crossed-dipole, and pyramidal horns would not have such an ideal radiation pattern.

   The closest to a hemispherical radiation may be an open-ended square waveguide on a fairly large ground plane.

   The radiation pattern of an array is the product of array pattern and that of constituent elements.

$$E_{field} = E_{array} \times E_{element}$$

We have assumed throughout that the radius of the $E_{element}$ pattern is unity and that E- and H-field are identical and orthogonal. These assumptions are very difficult to meet in a practical design.

2. We have limited our demonstration computations to three of element excitation amplitude distributions because the coefficients of polynomials must not have zero end points. For a planar slotted waveguide array, the smallest coefficient should not be less than 0.05 of the largest coefficient for manufacturing tolerance.
3. We have used four significant digits in our excitation amplitude distributions. In a practical design control within three significant digits is extremely difficult. Imagine a power divider required for the open-ended waveguide with three-significant-digit accuracy. The radiation patterns presented should be taken as a design guide.
4. Planar slotted waveguide array antennas above the Ku band are rare, for the precision required to manufacture such an antenna is prohibitively expensive, while below L band they are bulky and heavy even with half-height waveguides. Slotted waveguide array antennas are usually found between the S-band and X-band. Outside of this range, a different radiator design or a reflector antenna is preferred.
5. Then why the array antenna? The aperture efficiency of an array antenna can be relatively high, between 70% to 76%, whereas the efficiency of reflector is between 50% to 55%. More importantly, array antennas can be designed with sidelobe level of –55 dB or lower. The lowest sidelobe level of reflector antenna may be around –30 to –35 dB at best.
6. It is clear that the best excitation distribution is Taylor distribution for all array apertures. The Taylor distribution combines the lower gain loss and the narrower beamwidth that are competitive with Chebyshev and it controls the rate of decrease of the sidelobes.

List of Programs

(1) ANT_LINE.CPP	Radiation patterns, linear array; uniform, Chebyshev, Taylor, Hamming, and Lambda excitation amplitude distribution.
(2) ANT_CIRC.CPP	Radiation patterns, circular array; uniform, Chebyshev, Taylor, Hamming, and Lambda excitation amplitude distribution.
(3) ANT_ELIP.CPP	Radiation patterns, elliptic array; uniform, Chebyshev, Taylor, Hamming, and Lambda excitation amplitude distribution.
(4) MONO_LIN.CPP	Radiation patterns, monopulse linear array.
(5) MONO_CIR.CPP	Radiation patterns, monopulse circular array.
(6) MONO_ELP.CPP	Radiation patterns, monopulse elliptical array.
(7) ANT_LINE.H	Header file, linear array for various excitation distributions.
(8) ANT_CIRC.H	Header file, circular array for various excitation distributions.
(9) ANT_ELIP.H	Header file, elliptical array for various excitation distribution.
(10) MONO_LIN.H	Header file, linear array for Chebyshev distribution.
(11) MONO_CIR.H	Header file, circular array for Taylor distribution.
(12) MONO_ELP.H	Header file, elliptical array for Hamming distribution.
(13) BESSELLI0.H	Computes the modified Bessel function of the first kind, zero order, line source.
(14) BSLI0CIR.H	Computes the Bessel function for a circular array.
(15) BSEI0ELP.H	Computes the Bessel function for an elliptical array.

References

[1] Elliott, R. S., *Antenna Theory and Design*, Englewood Cliffs, N.J.: Prentice-Hall, 1981.

[2] Stuzman, W. L., and G. A. Thiele, *Antenna Theory and Design*, New York: John Wily and Sons, 1981.

[3] Silver, S. (ed.), *Microwave Antenna Theory and Design*, MIT Radiation Laboratories Series, Vol. 12, New York: McGraw-Hill, 1949.

[4] Jasik, H. (ed.), *Antenna Engineering Handbook*, New York: McGraw-Hill, 1961.

[5] Brown, L. B. and G. A. Scharp, *Tschebyscheff Antenna Distribution, Beamwidth and Gain Table*, U.S. Naval Ordnance Lab., Report 4629, Federal Sci. & Tech. Information, RB 151 002, U.S. Department of Commerce, February 1956.

[6] Taylor, T. T., "Design of Line Source Antenna for Narrow Beamwidth and Low Sidelobes," *IRE, Trans AP*, Vol. 3, January 1955.

[7] Bickmore, R. W., and R. J. Spellmire, A Two-Parameter Family of Line Sources, Hughes Aircraft Co., Tech Memo. 595, 1965.

[8] Spellmire, R. J., Tables of Taylor Aperture Distributions, Hughes Aircraft Co., Tech Memo. 581, 1958.

[9] Sherman, S. M., *Monopulse Principles and Techniques*, Dedham, MA: Artech House, 1984.

[10] Love, A. W. (ed.), *Reflector Antennas*, New York: IEEE Press, 1978.

Appendix 6A

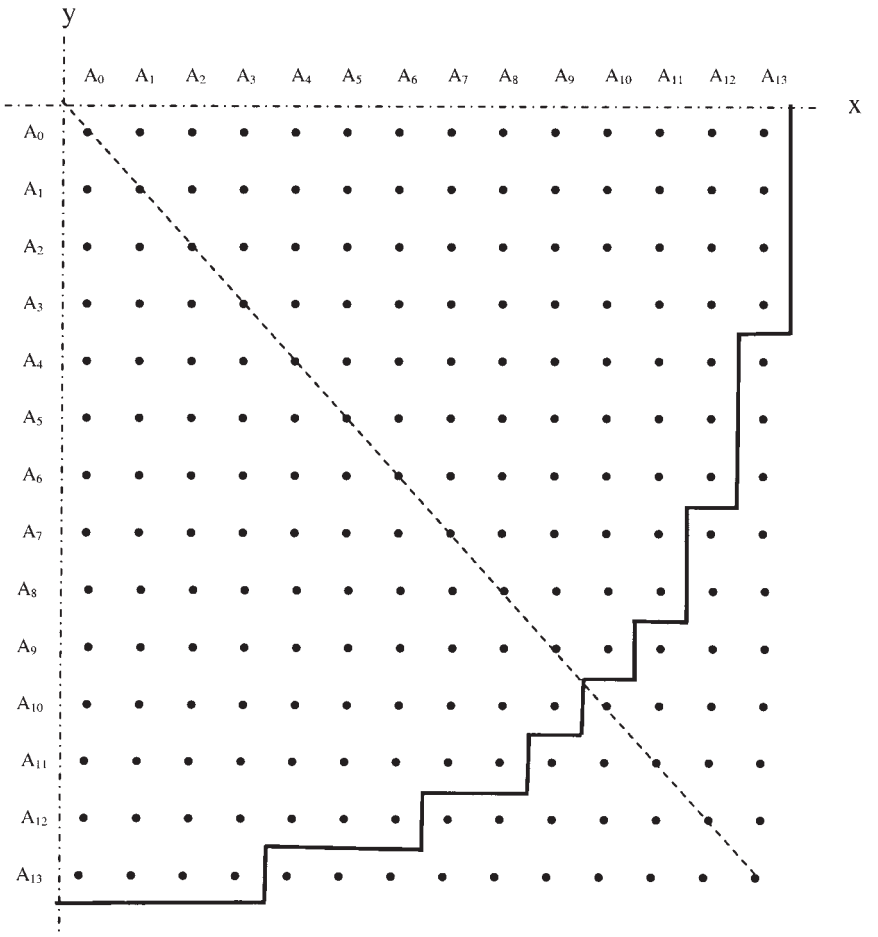


Figure 6.A.1 Circular aperture by eliminating some elements from square grid points.

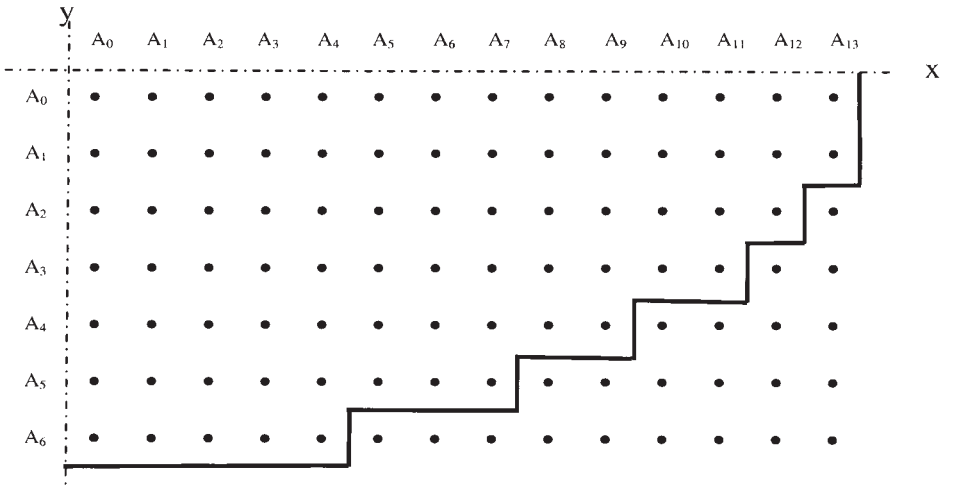


Figure 6.A.2 Elliptical aperture by eliminating some elements from a rectangular grid points.

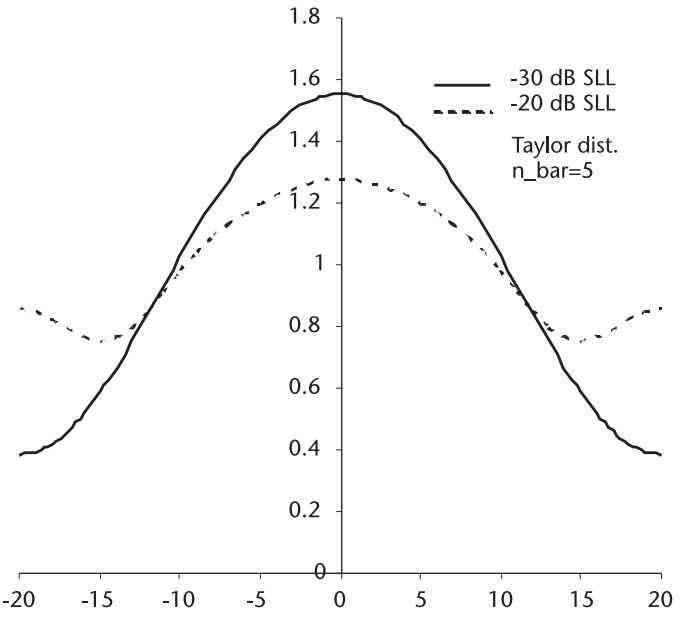


Figure 6.A.3 Taylor distributions.

Closing Remarks

For the circular aperture we put the element excitation distribution for the Chebyshev as:

```
cheby[]={1.0000, 0.9908, 0.9724, 0.9455, 0.9104,
         0.8679, 0.8189, 0.7643, 0.7052, 0.6428,
         0.5782, 0.5127, 0.4472, 1.3590};
```

The distribution is correct for rows 0, 1, 2, and 3. For rows 4, 5, and 6, the distribution should be recomputed, which will be different from the distribution given above, and the distribution for rows 7 and 8 should be recomputed, which will be different from the rows above, and so on. We haven't done that for simplicity and brevity. This applies to the column distributions as well. The same arguments apply to the elliptical array.

CHAPTER 7

Target Detection

7.1 Introduction

In this chapter we study the principles of radar target detection. We analyze the probability of detection, the probability of false alarm, the threshold (bias) level and pulse integration. We investigate how to determine the threshold so that a desired detection probability and the false alarm probability are obtained and how many return pulses are to be integrated.

Let us examine a basic structure of pulse integration shown in Figure 7.1. The IF signal plus noise is detected by a diode detector and converted to digital levels by an A/D converter. The digital numbers are stored in a bank of memories through a summer whose other inputs are the fed-back numbers from a scalar Q. The length of the memory bank is equal to the number of range bins of display the indicator (maximum detection range). The feedback path includes the scalar Q that prevents an overflow in the memory bank by dividing the output of memory bank by 1/2, 3/4, 7/8, . . . (N-1)/N. The output of the range bin memory is sequentially compared with the threshold V_{Th} by a comparator. When the output of the memory exceeds the threshold we declare that a target is present in that range bin; otherwise no target is declared. The basic structure is called a noncoherent recirculating accumulator, or pulse integrator, for short.

Let us examine how the pulse integrator detects a target signal embedded in noise. We simulate three targets: target 1, target 2, and target 3; a strong target, an intermediate target, and a weak target, respectively. For a simulation we take Rayleigh noise, uniformly and identically distributed throughout. (See Chapter 2 for the characteristics of Rayleigh noise.) Rayleigh noise has mean $\sigma(\pi/2)^{1/2}$ and variance $\sigma^2(2-\pi/2)$, and we assume $\sigma=1$.

$$\begin{aligned}\text{Noise power} &= \text{mean squared} + \text{variance}\\ &= (1.253314)^2 + 0.429204\\ &= 2 \text{ watts}\end{aligned}$$

$$\begin{aligned}\text{Signal power}\quad \text{first target:} &\quad 4.0\times4.0 = 16 \text{ watts}\\ \text{second target:} &\quad 2.0\times2.0 = 4 \text{ watts}\\ \text{third target:} &\quad 1.0\times1.0 = 1 \text{ watt}\end{aligned}$$

Thus, the SNR in power is

$$\begin{aligned}\text{First target:} &\quad 16/2 = 8 = 9.03 \text{ dB};\\ \text{Second target:} &\quad 4/2 = 2 = 3.01 \text{ dB};\\ \text{Third target:} &\quad 1/2 = 0.5 = -3.01 \text{ dB}.\end{aligned}$$

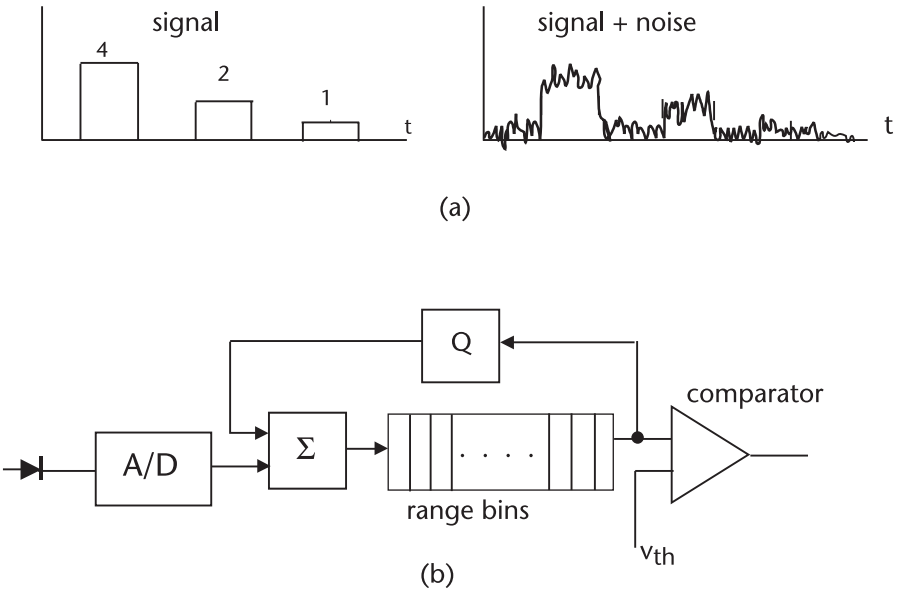


Figure 7.1 Pulse integrator.

The third target is buried under the noise, we would say. The signal plus noise is envelope detected, A/D-converted, stored in the range bin memories, and recirculated eight times. Read PULSEDET.CPP and PULSEDET.DAT to see how the signal plus noise accumulates as they circulate once, twice, three times,…, eight times. The target levels are identical for all eight sweeps but Rayleigh noise samples are different from sweep to sweep. After eight recirculations/accumulations we compare the levels of the memory bank with a threshold (bias) level V_{Th}. We have varied V_{Th} from 90 to 115 in increments of five and counted the number of false declarations, the number of missed detections, and the detection success as percentage. Read the footnote at the end of the program. The result of the program is summarized in the following:

V_{Th} is set at:	*Number of false declarations*	*Number of missed detections*	*Detection success*
90	13	15	88%
95	7	21	84%
100	4	34	73%
105	3	47	63%
110	1	57	55%
115	0	70	45%

We note that as we increased the threshold level, the number of false declaration decreased (good news), but the number of missed detection increased (bad news), and the detection success suffered (bad news). If we desire to have a detection success of 50%, the threshold bias level must be set at somewhere between 110 and 115. On the other hand, if we desire a detection success of 90% the signal-to-noise ratio of the weak target must be increased or we must circulate more than eight times.

The threshold V_{Th}, the false alarm probability P_{fa}, the probability of detection P_d, and the SNR (in power) are all interlocked. If we alter one parameter the other three parameters will be affected. When we specify three parameters the fourth parameter will be automatically determined. The number of pulses integrable N is dictated by the pulse repetition frequency, antenna 3-dB beamwidth, and the antenna rotation rate.

$$N = (\theta/\dot{\theta})\, f_{PRF}$$

So far we have conducted an inexact heuristic experiment to show the relationship among the detection probability, the false alarm probability, the threshold bias level, the SNR, and the number of pulses integrated. In the next sections we derive the mathematical expression for each term and their interlocked relationship.

Before we proceed we should discuss the law governing the diode detector in Figure 7.1. A diode conductance characteristic is shown in Figure 7.2.

What would be the best characteristic of the diode detector for target detection?

Marcum [1] takes the maximum likelihood ratio of Rician random variables (signal plus noise) to Rayleigh random variables (noise only) in order to investigate the best diode detector characteristics.

$$\begin{aligned} MLR &= \prod_{i=1}^{N} v_i \exp\left\{\frac{-(v_i^2 + S^2)}{2}\right\} I_o(v_i S) \Big/ \prod_{i=1}^{N} v_i \exp\left\{\frac{-v_i^2}{2}\right\} \\ &= \exp\{-S^2/2\} \prod_{i=1}^{N} I_o(v_i S) \end{aligned} \tag{7.1}$$

where

v_i: Noise samples;
S: Signal amplitude;
I_o: Modified Bessel function of first kind, zero order.

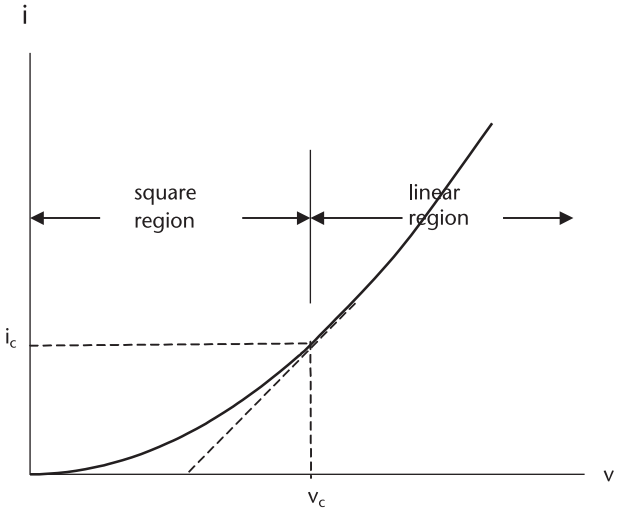


Figure 7.2 Diode conductance characteristic.

Taking the natural logarithm of both sides of (7.1) gives,

$$\sum_{i=1}^{N} \ln[I_o(v_i S)] = \ln \text{ MLR} + S^2$$

or

$$\sum_{i=1}^{N} \ln[I_o(v_i S)] = \lambda$$

The λ is a constant that would determine the presence or absence of target; when the summation is equal to or greater than λ, a target present is declared; otherwise no target is declared. Thus the best diode detector law should be a logarithmic.

$$y = \ln[I_o(vS)]$$

Assume that the signal level is very low, below the conduction knee, V_c, which implies that a large number of pulses must be integrated to reach the λ. Then,

$$I_o(vS) = 1 + \frac{v^2S^2}{4} + \frac{v^4S^4}{2^5 2!} + \cdots$$

$$y = \ln[I_o(vS)] = \ln\left[1 + \frac{v^2S^2}{4} + \cdots\right] \approx \frac{v^2S^2}{4}$$

Therefore, a square-law detector is the best choice. On the other hand, if the signal level is high, then

$$I_o(vS) = \frac{\exp\{vS\}}{\sqrt{2\pi vS}}\left(1 + \frac{1}{8vS} + \cdots\right) \approx \frac{\exp\{vS\}}{\sqrt{2\pi vS}}$$

$$y = \ln[I_o(vS)] \approx vS - \frac{1}{2}\ln(2\pi vS) \approx vS$$

Thus, a linear detector is the choice. A smaller number of pulses would be integrated to reach λ. Marcum [1] presented the performance of the two detectors as shown in Figure 7.3, where the ordinate is the decibel advantage of the square-law detector over the linear detector.

Figure 7.3 shows there is negligible difference between two detector laws. Note that for N=1 and N=70 the two detectors are identical in performance. For small N, a linear detector is better by an amount not exceeding 0.12 dB. For large N, a square-law detector is better than a linear detector by an amount that asymptotically approaches 0.19 dB as N becomes larger than 10,000. (We rarely integrate more than 100 return pulses, if ever.)

The practical implementation of the detector by a diode whether the diode is a hot-carrier diode, germanium, or silicon diode exhibits a square-law characteristic for small signals and a linear characteristic for large signals, thereby approximating

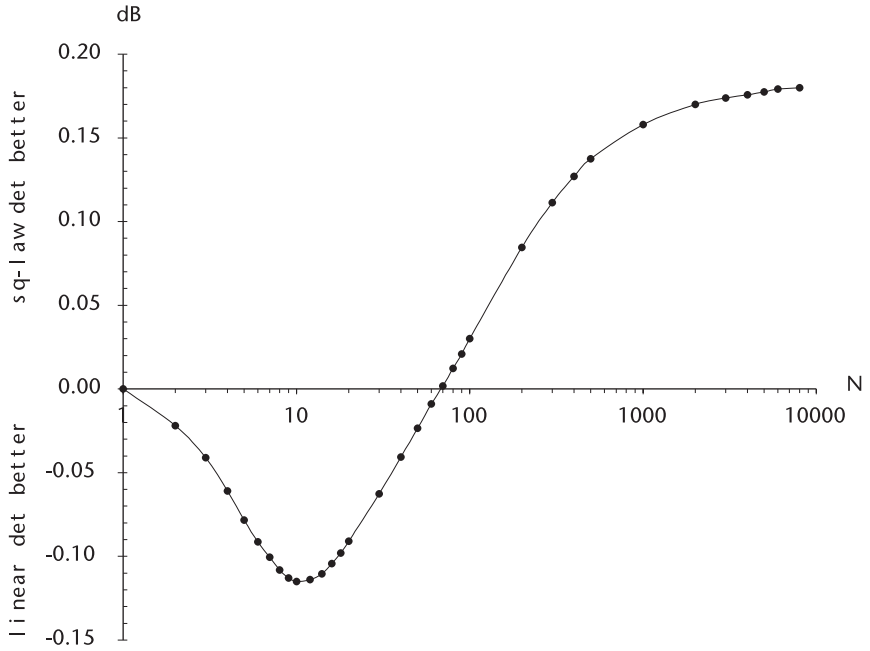


Figure 7.3 Comparison between the linear and square-law detector.

the ideal logarithmic detector law. For certain mathematical erivations a square-law is more often convenient. The performance difference between the two-detectors law is less than 0.12 dB when the number of pulses integrated is less than, say, 100.

7.2 Probability of Detection and the False Alarm Probability for Marcum's Target Model

This section follows the classic work of Marcum [1]. Marcum analyzed the detection problem when the target is nonfluctuating, that is, the target is a pherical object several wavelength in diameter so that the target cross-section is constant for all observation angles. Swerling has extended the problem to cover when the target cross-section is fluctuating from one observation to the next. The problem of fluctuating Swerling's target models will be discussed in detail in the following sections.

The mathematical analysis starts with help from Figure 7.4. The probability density function (pdf) of Rayleigh noise and that of the Rician pdf (signal plus noise) are given by,

$$f_N = \frac{v}{\sigma^2} \exp\left\{\frac{-v^2}{2\sigma^2}\right\}, \qquad \text{Rayleigh pdf} \tag{7.2}$$

$$f_{S+N} = \frac{v}{\sigma^2} \exp\left\{\frac{-1}{2\sigma^2}(v^2 + S^2)\right\} I_o\left(\frac{vS}{\sigma^2}\right) \qquad \text{Rician pdf} \tag{7.3}$$

where

v: Instantaneous noise amplitude;
σ^2: The variance of noise;
S: Instantaneous signal amplitude;
I_o: Modified Bessel function of first kind, zero order.

The probability of false alarm is given by

$$P_{fa} = \int_{V_{Th}}^{\infty} \frac{v}{\sigma^2} \exp\left\{\frac{-v^2}{2\sigma^2}\right\} dv \tag{7.4}$$

If we let

$$\frac{v}{\sqrt{2}\sigma} = x, \quad dv = \sqrt{2}\,\sigma\, dx,$$

then

$$\frac{v}{\sigma^2} = \frac{1}{\sigma}\left(\frac{v}{\sigma}\right) = \frac{\sqrt{2}}{\sigma}\left(\frac{v}{\sqrt{2}\sigma}\right)$$

Substitution of the variable leads to an expression for P_{fa}.

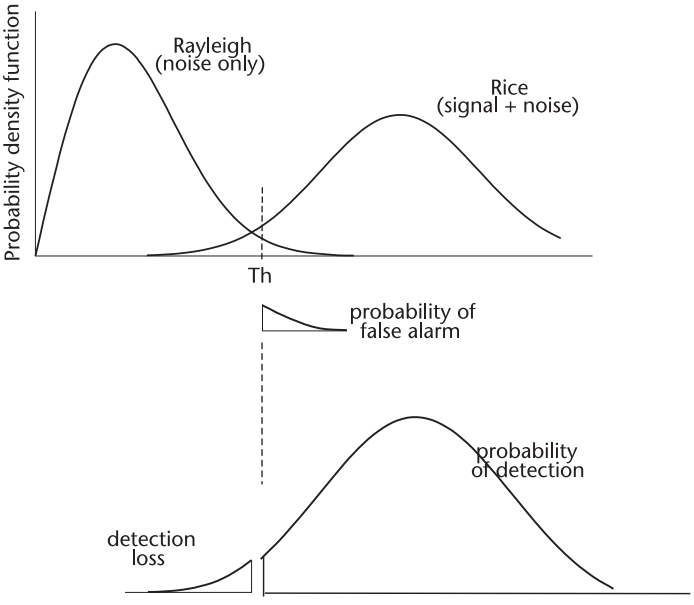


Figure 7.4 Probability density function, Rayleigh and Rician and the threshold.

$$P_{fa} = \int_{V_{Th}/\sqrt{2}\sigma}^{\infty} \frac{\sqrt{2}}{\sigma} \, x \, \exp\{-x^2\} \, \sqrt{2}\sigma \, dx$$

$$= \int_{V_{Th}/\sqrt{2}\sigma}^{\infty} 2x \, \exp\{-x^2\} \, dx = \exp\left\{\frac{-V_{Th}^2}{2\sigma^2}\right\} \tag{7.5}$$

We define two terminologies frequently used in detection literature: the threshold level and bias level, which we have used ambiguously in the past discussion.

Taking the natural logarithm of (7.5) we have,

$$\ln P_{fa} = \frac{-V_{Th}^2}{2\sigma^2}, \quad V_{Th} = \sqrt{-2\sigma^2 \ln(P_{fa})} \tag{7.6}$$

If we set P_{fa} = 1.0E-6 and the variance of noise σ^2 is unity, σ^2 = 1,

$$V_{Th} = 5.2526$$

On the other hand, we could define the bias level y_b as follows from (7.5),

$$P_{fa} = \exp\left\{\frac{-V_{Th}^2}{2\sigma^2}\right\} = \exp\{-y_b\}, \quad y_b = -\ln(P_{fa}) \tag{7.7}$$

If we desire that P_{fa} = 1.0E-6, regardless of the variance of noise,

$$y_b = -\ln (1.0E\text{-}6) = 13.8155$$

Thus the threshold level is in the unit of volts provided the variance of noise is known, whereas the bias level is the dimensionless power ratio. Readers may have preference for one over the other; however, when noise power varies over one surveillance sector to another, and the threshold is automatically adjusted to follow, the bias level is the preferred. We remind ourselves that the threshold and the bias given above are for the case of single-pulse detection. The bias level for multiple pulses will be presented shortly.

Next we derive an expression for the detection probability P_d, given from 7.3 and Figure 7.4.

$$P_d = \int_{V_{th}}^{\infty} \frac{v}{\sigma^2} \exp\left\{\frac{-1}{2\sigma^2}(v^2 + S^2)\right\} I_o\left(\frac{vS}{\sigma^2}\right) dv$$

For convenience we change the variable as follows:

$$\alpha = \frac{v}{\sigma} \quad \text{and} \quad \beta = \frac{S}{\sigma}, \quad \text{then} \quad dv = \sigma d\alpha$$

The integrand becomes

$$\frac{v}{\sigma^2}\exp\left\{\frac{-1}{2\sigma^2}(v^2+S^2)\right\}I_o\left(\frac{vS}{\sigma^2}\right)$$

$$=\frac{v}{\sigma^2}\exp\left\{-\left(\frac{v^2}{2\sigma^2}+\frac{S^2}{2\sigma^2}\right)\right\}I_o\left(\frac{v}{\sigma}\frac{S}{\sigma}\right)$$

$$=\frac{\alpha}{\sigma}\exp\left\{-\frac{(\alpha^2+\beta^2)}{2}\right\}I_o(\alpha\beta)$$

The detection probability is then,

$$P_d=\int_{V_{th}/\sigma}^{\infty}\alpha\exp\{-\frac{(\alpha^2+\beta^2)}{2}\}I_o(\alpha\beta)\,d\alpha=Q(\alpha,\beta) \tag{7.8}$$

Interested readers should consult Brennan [2]. We don't pursue any further the Q function in this form, since the form (7.8) is good for only single-pulse detection, N=1. Anticipating N > 2, perhaps N=4, 8, 16 …, we plan to compute the detection probability through the characteristic function.

The characteristic function and the probability density function are a Fourier transform pair [3, 4], and the characteristic function of multiple pulses is easily obtained by raising the characteristic function of a single pulse to the Nth power.

$$f_x(x)=\frac{1}{j2\pi}\int_{-\infty}^{\infty}C_x(j\omega)\exp\{-j\omega x\}\,d\omega \tag{7.9a}$$

$$C_x(j\omega)=\int_{-\infty}^{\infty}f_x(x)\exp\{j\omega x\}\,dx \tag{7.9b}$$

and,

$$C_N=(C_1)^N \tag{7.10}$$

Marcum and Swerling have presented the characteristic functions in their reports and we have tabulated them in Section 7.5.1. The characteristic function of Marcum's target model, a nonfluctuating, constant cross-section target is given by

$$C_1(p)=\frac{\exp\{-x\}\exp\left\{\frac{x}{1+p}\right\}}{(p+1)} \tag{7.11}$$

$$C_N(p)=(C_1)^N=\frac{\exp\{-Nx\}\exp\left\{\frac{Nx}{1+p}\right\}}{(p+1)^N} \tag{7.12}$$

where

C_1: Characteristic function for N=1;
C_N: Characteristic function for N≥2;
N: Number of pulses;
p: Fourier transform variable, $j\omega$.

The probability density function for multiple pulses is obtained by Fourier transformation of $C_N(p)$.

$$f(t) = \frac{1}{j2} \int_{-j\infty}^{j\infty} C_N(p) \exp\{pt\}\, dp$$

$$= \frac{1}{j2} \exp\{-Nx\} \exp\{-t\} \int_{-j\infty}^{j\infty} \frac{\exp\{t(p+1)\} \exp\left\{\frac{Nx}{p+1}\right\}}{(p+1)^N} d$$

We replace the two exponential terms of integrand by summation of infinite series.

$$\exp\{Nx/(p+1)\} = \sum_{k=0}^{\infty} \frac{1}{k!} \left(\frac{Nx}{p+1}\right)^k$$

$$\exp\{t(p+1)\} = \sum_{m=0}^{\infty} \frac{1}{m!} [t(p+1)]^m$$

then,

$$f(t) = \frac{1}{j2\pi} \exp\{-Nx\} \exp\{-t\} \int_{-j\infty}^{j\infty} \sum_{k=0}^{\infty} \sum_{m=0}^{\infty} \frac{(Nx)^k t^m (p+1)^m}{k!\, m!} dp$$

$$= \exp\{-Nx\} \exp\{-t\} \sum_{k=0}^{\infty} \frac{(Nk)^k}{k!} \frac{t^{(N-1+k)}}{(N-1+k)!}$$

The integration is by the residue theorem in the complex plane. Since f(t) is probability density function, the detection probability is obtained by integrating t from y_b to infinity,

$$P_d = \int_{y_b}^{\infty} f(t)dt \overset{\text{or}}{=} 1 - \int_0^{y_b} f(t)dt$$

$$= \int_{y_b}^{\infty} \exp\{-Nx\} \exp\{-t\} \sum_{k=0}^{\infty} \frac{(Nk)^k}{k!} \frac{t^{(N-1+k)}}{(N-1+k)!}\, dt$$

Interchanging the order of integration and summation, the detection probability P_d is given by,

$$P_d = \sum_{k=0}^{\infty} \frac{e^{-Nx}(Nk)^k}{k!} \int_{y_b}^{\infty} \frac{e^{-t}t^{(N-1+k)}}{(N-1+k)!}dt \tag{7.13}$$

The integral is a form of the incomplete Gamma function [5]. The Gamma function has an interesting property,

$$\int_0^{\infty} \frac{e^{-x}x^m}{m!}dx = \left[-e^{-x}\sum_{r=0}^{m}\frac{x^{m-r}}{(m-r)!}\right]_0^{\infty} = 1$$

so that the incomplete Gamma function can be expressed as

$$\int_0^{y_b} \frac{e^{-x}x^m}{m!}dx = \left[-e^{-x}\sum_{r=0}^{m}\frac{x^{m-r}}{(m-r)!}\right]_0^{y_b} = 1 - e^{-y_b}\sum_{r=0}^{m}\frac{x^{m-r}}{(m-r)!}$$

or,

$$\int_{y_b}^{\infty} \frac{e^{-x}x^m}{m!}dx = 1 - \int_0^{y_b} \frac{e^{-x}x^m}{m!}dx = e^{-y_b}\sum_{r=0}^{m}\frac{x^{m-r}}{(m-r)!}$$

Substituting the above result into (7.13), we have finally obtained an expression for P_d of Marcum's target model for N pulses.

$$P_d = \sum_{k=0}^{\infty} \frac{e^{-Nx}(Nk)^k}{k!} \sum_{r=0}^{N-1+k} \frac{e^{-y_b}y_b^r}{r!} \tag{7.14}$$

Equation (7.14) cannot be programmed directly since both summations involve the index that must go up to infinity. Reference [6] has some suggestions how to handle the problem.

The infinite summation can be handled as follows, using shorthand notations:

$$\begin{aligned}
&\sum_{k=0}^{\infty} \frac{e^{-Nx}(Nk)^k}{k!} \sum_{r=0}^{N-1+k} \frac{e^{-y_b}y_b^r}{r!} \\
&= \left[\sum_{k=0}^{L}(x,k) + \sum_{k=L+1}^{\infty}(x,k)\right] \cdot \sum_{r=0}^{N-1+k}(y_b,r) \\
&= \sum_{k=0}^{L}(x,k) \cdot \sum_{r=0}^{N-1+k}(y_b,r) + \sum_{k=L+1}^{\infty}(x,k) \cdot \sum_{r=0}^{N-1+k}(y_b,r) \\
&= \sum_{k=0}^{L}(x,k) \cdot \sum_{r=0}^{N-1+k}(y_b,r) + \left[1 - \sum_{k=0}^{L}(x,k)\right] \cdot \sum_{r=0}^{N-1+k}(y_b,r)
\end{aligned}$$

All summation indices are finite now. For a sufficiently large L the bracketed expression of the second term can be made as small as we wish, (i.e., 1.0E-12 or less).

$$\left[1 - \sum_{k=0}^{L} (x,k)\right] \cdot \sum_{r=0}^{N-1+k} (y_b,r) => \varepsilon \sum_{r=0}^{N-1+L} (y_b,r)$$

The last summation, being the incomplete Gamma integral, approaches unity for sufficiently large L.

Therefore, the detection probability will be given by the first term, a product of two summations.

$$P_d = \sum_{k=0}^{L} (x,k) \cdot \sum_{r=0}^{N-1+k} (y_b,r) + \text{remainder}$$

$$= \sum_{k=0}^{L} \frac{e^{-Nx}(Nx)^k}{k!} \sum_{r=0}^{N-1+k} \frac{e^{-y_b} y_b^r}{r!} + \text{remainder} \qquad (7.15)$$

Considering the highest useful P_d is 0.999999, if we ascertain that the remainder is less than 0.000001 we can safely terminate the infinite summation at a large L.

Equation (7.15) is programmed in PD_(0)_1.CPP. The (0) identifies that the target is Marcum's nonfluctuating, constant cross-section target. The _1 signifies that the number of pulse integrated is one, N=1. The false alarm probability is set at P_{fa}=1.0E-6, and the bias Y_b=13.8155 per Eq (7.7). The detection probability is computed as SNR is incremented in step of 1 dB.

SNR (dB)	*Index L*	P_d
0.0	17	0.000122
1.0	18	0.000229
2.0	19	0.000453
3.0	20	0.000941
4.0	22	0.002040
5.0	23	0.004585
6.0	25	0.010554
7.0	28	0.024530
8.0	30	0.056120
9.0	33	0.122824
10.0	37	0.248049
11.0	41	0.444041
12.0	47	0.679386
13.0	54	0.874437
14.0	62	0.972142
15.0	72	0.997221
16.0	84	0.999905
17.0	99	0.999997
18.0	117	0.999999
19.0	139	0.999999
20.0	153	0.999999

The results are stored in PD_(0)_1.DAT and a graph is shown in Figure 7.5. The DAT and the graph indicate that SNR required for 90% detection probability is 13.18 dB. For a detection probability of 50%, the required SNR is 11.2 dB.

For multiple-pulse integration, N=2, 4, 8, . . . , we must compute the bias level Y_b that corresponds to the number of pulses to be integrated. The y_b for N>1 should be different from that for N=1.

The probability density function of noise alone at the output of a square-law detector is given by Marcum, DiFranco, and Rubin [8].

$$f_N(y) = \frac{e^{-Y} y^{N-1}}{(N-1)!} \tag{7.16}$$

where

N: Number of noise pulses;
y: Instantaneous noise power.

The probability of false alarm is determined by integrating (7.16) over an appropriate limit.

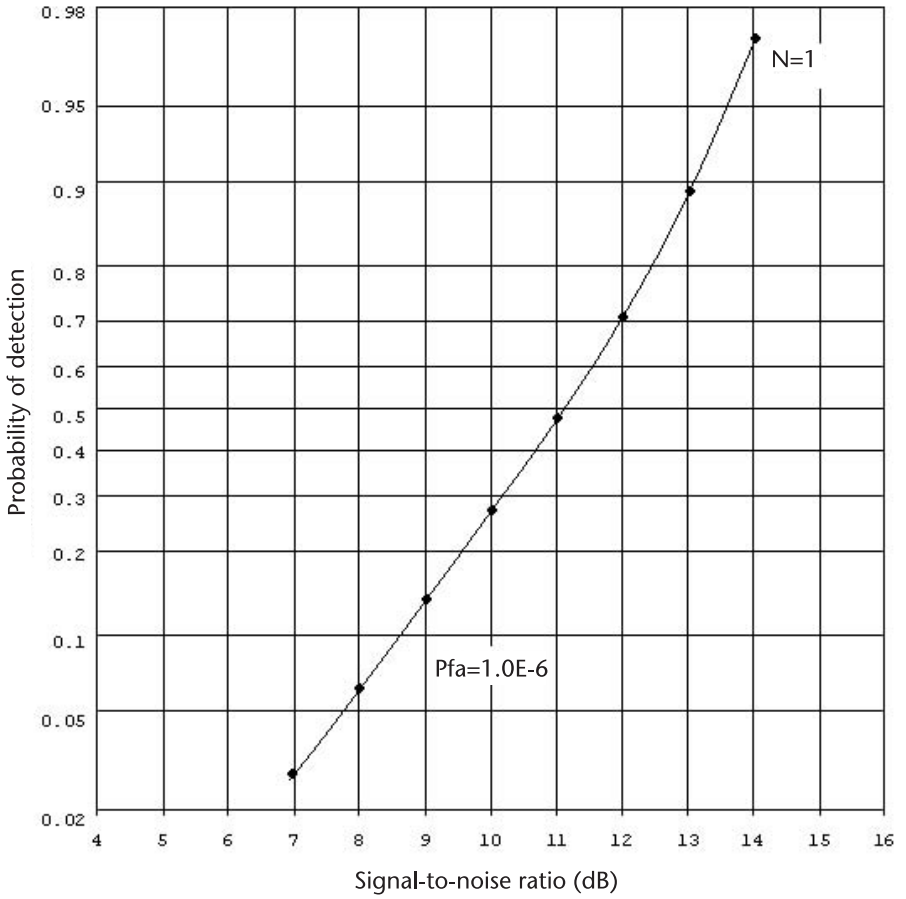


Figure 7.5 P_d vs SNR, Marcum's target model.

$$P_{fa} = \int_{y_b}^{\infty} \frac{e^{-Y} y^{N-1}}{(N-1)!}\, dy = 1 - \int_{0}^{y_b} \frac{e^{-Y} y^{N-1}}{(N-1)!}\, dy \tag{7.17}$$

where y_b is the bias level required for a specified P_{fa} and number of pulses integrated. The integral is again a form of the incomplete Gamma function. The integral is replaced by sum of an infinite power series.

$$P_{fa} = \sum_{k=0}^{\infty} \frac{e^{-y_b} y_b^k}{k!} \tag{7.18a}$$

and

$$y_b = \ln \left[\frac{\sum_{k=0}^{N-1} y_b^k/k!}{P_{fa}} \right] \tag{7.18b}$$

Equation 7.18b is a transcendental form. We want to compute y_b when the number of pulses may be as large as 100. In order to solve (7.18b) we would employ either a bisection method, or the Newton-Raphson method, or some other root-finding methods. See Chapter 12.

For a fast convergence we adopt the Newton-Raphson method. This method is an iterative procedure to find the solution of a transcendental equation after we find the first derivative, and the derivative must not be zero.

$$y_{k+1} = y_k - \frac{f(y_b)}{f'(y_b)}$$

We let

$$f(y_b) = y_b - \ln \left[\frac{\sum_{k=0}^{N-1} y_b^k/k!}{P_{fa}} \right] = y_b - \ln \left[\frac{S(y_b,k)}{P_{fa}} \right]$$

or

$$S(y_b,k) = \sum_{k=0}^{N-1} y_b^k/k!$$

for a shorthand notation.

The first derivative of $f(y_b)$ with respect to y_b is,

$$f'(y_b) = 1 - \frac{S'(y_b,k)}{S(y_b,k)} = \frac{y_b^{N-1}}{S(y_b,k)(N-1)!}$$

Thus, the iterative algorithm to obtain the bias level y_b for N≥2 is given by

$$y_b(K+1) = y_b(K) - \frac{y_b - \ln\left[\frac{S(y_b,k)}{P_{fa}}\right]^{\frac{1}{N-1}}}{\frac{y_b}{S(y_b,k)(N-1)!}} \qquad (7.19)$$

Equation 7.19 is programmed in BIAS_YB.CPP for P_{fa}=1.0E-6, N=1 to N=100. The iteration is terminated when $y_b(k+1) - y_b(k) \le 1.0E\text{-}12$. The number of iterations required is dependent upon the initial estimate of $y_b(k)$. For example, we estimate (initialize) $y_b(1) \approx 13.0$ for N=1 per (7.7). The program produced $y_b(1)=13.815511$, a correct answer with six significant digits. The initial estimate of $y_b(2)$ is set at $y_b(1)$, the previous y_b. The initial estimate of $y_b(i+1)$ is set to $y_b(i)$, and so on. This is permissible since y_b is a monotonic function of N.

Pachares [8] has published y_b with four significant digits for N<150. Marcum reported y_b for N>150, and Meyer and Meyer [9] in a graphic form. With y_b

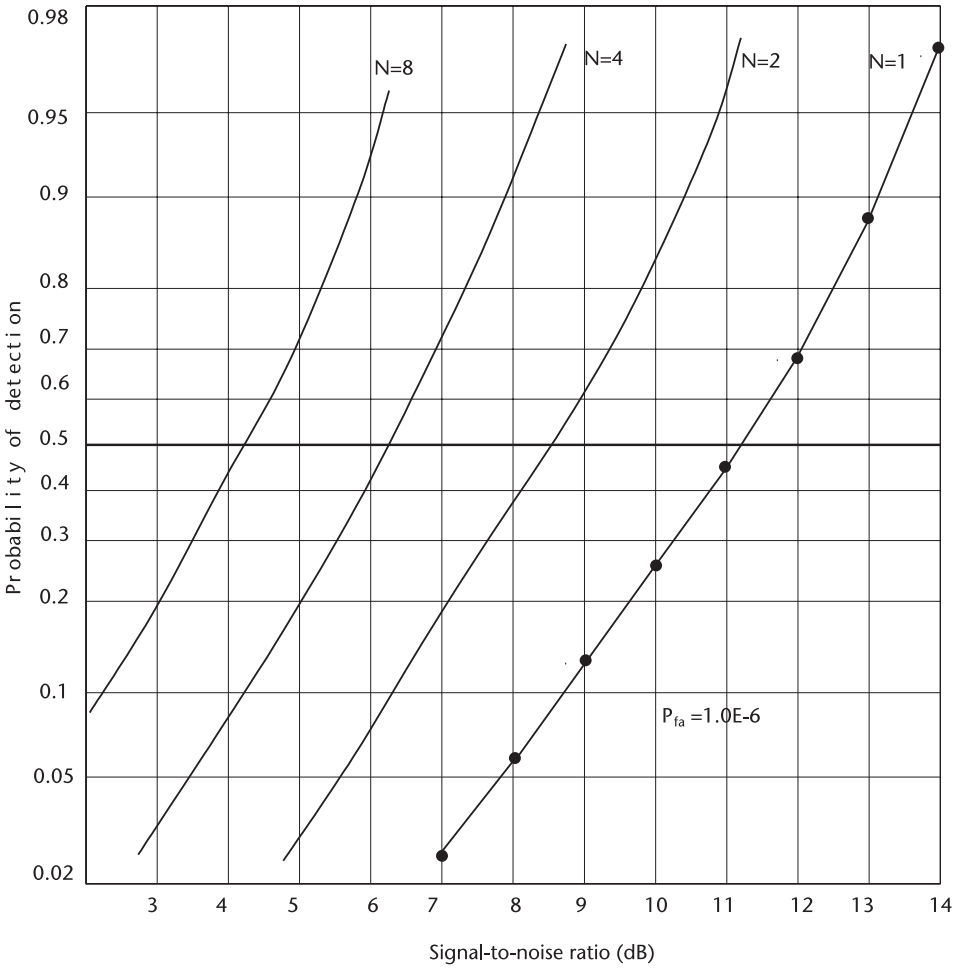


Figure 7.6 Pd versus SNR, Marcum's target.

computed for N ≥2 on hand we are ready to compute the detection probability for Marcum's nonfluctuating target model.

Equation (7.15) is programmed in PD_(0)_N.CPP for N=2, 4, 8, 16 and 32 with matching y_b and P_{fa}=1.0E-6. The results are stored in PD_(0)_N.DAT, are shown in Figure 7.6.

As an additional reference we mention Albersheim's work [10] on Marcum's target model. He published an empirical equation that shows the SNR per pulse required when the number of pulses to be integrated N is given and the detection probability and the false alarm probability are specified. His closed-form approximation is remarkably accurate, within two-tenths of a decibel. Interested readers should read an article by Tufts and Cann [11]. Albersheim's empirical approximation is

$$\mathrm{SNR}_{(\mathrm{db})} = -5.0\log_{10}(\mathrm{N}) + \left(6.2 + \frac{4.54}{\sqrt{\mathrm{N}+0.44}}\right)\log_{10}[\mathrm{A} + 0.12\mathrm{AB} + 1.7\mathrm{B}] \tag{7.20}$$

where

$$\mathrm{A} = \ln\left(\frac{0.62}{\mathrm{P_{fa}}}\right), \quad \mathrm{B} = \ln\left(\frac{\mathrm{P_d}}{1-\mathrm{P_d}}\right)$$

Equation (7.20) is programmed in DET_SNR.CPP, and the results are shown in Figure 7.7. The results and the graph indicated that the SNR per pulse required for Pfa=1.0E-6, N=1, 2, 4, . . . 64, agree astonishingly well with PD_(0)_N. CPP.

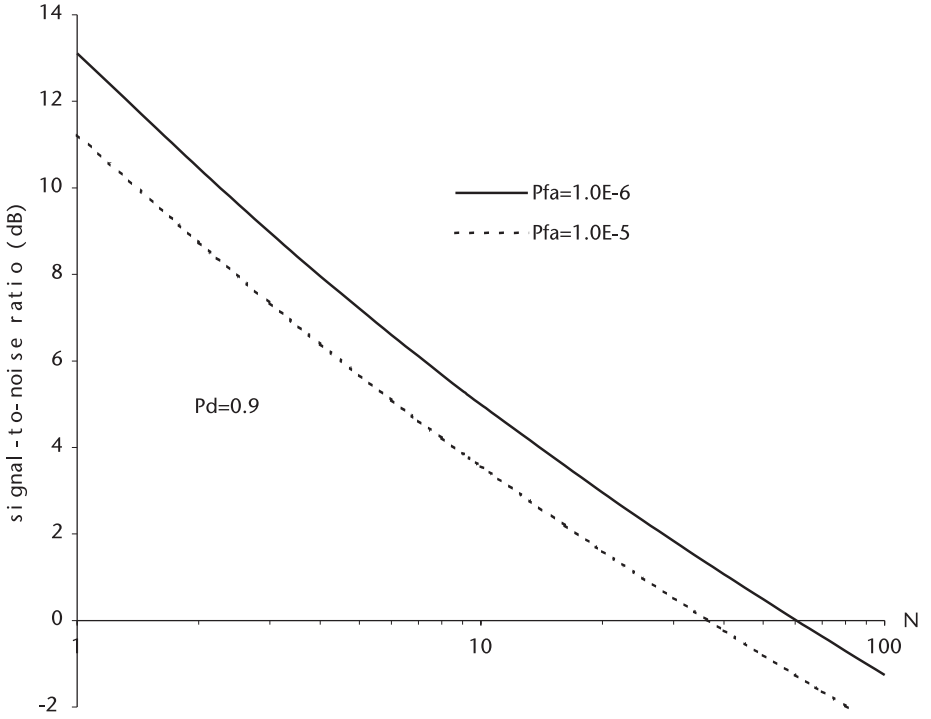


Figure 7.7 SNR required versus N, Albersheim's equation.

7.3 The Probability of Detection, Swerling Target Models

Swerling has extended Marcum's analysis to targets whose cross-sections fluctuate from one observation to the next. He classified targets into four basic models. (See Table 7.1.)

It is difficult to assign a target such as aircraft, battle tank, or surface ship, or farmland, or hill and mountain to any specific target model because the characteristics of a target cross-section are also dependent upon the operating parameters of the radar system. The number of published reports on target models is numerous and increasing [12].

It is claimed that the observed data on airborne aircraft targets agrees with the density distribution assumed as model 1 and 2. Models 3 and 4 apply to targets that can be represented as one large target surrounded by a number of small reflectors, one large target subjected to small changes in the aspect angle, a frigate or corvette with complex superstructures, or to a target observed by two-channel frequency diversity radar.

Target models 1 and 3 apply when the cross-section fluctuates from scan to scan. The return signals are highly correlated, that is to say, the pulse repetition rate is high with respect to the target dynamics. The f_{PRF}, high or low, and target motions, slow or fast, hold mutual relative meanings.

Target models 2 and 4 apply when the fluctuation is rapid, from one pulse to the next. The return signals are decorrelated, the pulse repetition rate is low with respect to the target dynamics, or the target aspect angle changes rapidly with respect to the pulse repetition frequency.

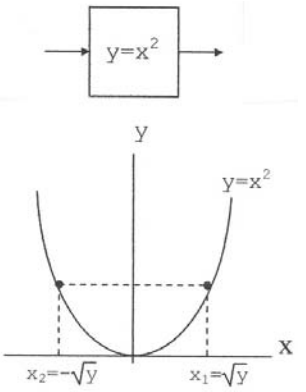
In this section we derive mathematical expressions for the detection probability of Swerling target model 1 through 4. The source codes for each model are written. Table 7.2 summarizes the equations for P_d.

We review the transformation of the probability density function at the output of square-law detector. What would be the output probability density function when a narrowband Gaussian noise is the input? Equation (7.21) provides an answer, Papoulis [3].

$$f_X(x) = \frac{1}{{}_x\sqrt{2}} \exp\left\{\frac{-x^2}{2\,{}_x^2}\right\} \rightarrow \boxed{y = x^2} \rightarrow f_Y(y) = ?$$

Table 7.1 Swerling's Target Classification Models

Target Model	*Probability Density Function of Target Cross-Section*	*Rate of Fluctuation of Cross-Section*
1	Rayleigh	Slow (scan-to-scan fluctuation)
2	Rayleigh	Fast (pulse-to-pulse fluctuation)
3	Chi-squared pdf 4 degrees of freedom	Slow(scan-to-scan fluctuation)
4	Chi-squared pdf 4 degrees of freedom	Fast (pulse-to-pulse fluctuation)

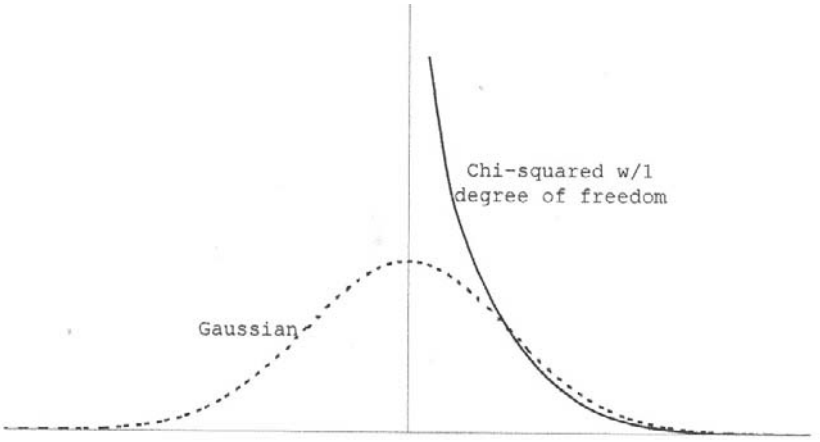


$$f_y(y) = \frac{f_x(x_1)}{|y'(x_1)|} + \frac{f_x(x_2)}{|y'(x_2)|} \qquad \left[y'(x) = \frac{d}{dx}y(x)\right]$$

$$= \frac{1}{2\sqrt{y}} \frac{1}{\sigma_x\sqrt{2\pi}} \exp\left\{\frac{-y}{2\sigma_x^2}\right\} + \frac{1}{2\sqrt{y}} \frac{1}{\sigma_x\sqrt{2\pi}} \exp\left\{\frac{-y}{2\sigma_x^2}\right\} \tag{7.21}$$

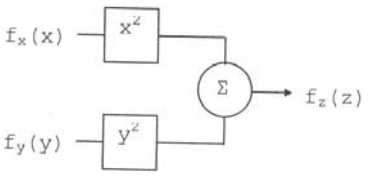
$$= \frac{1}{\sigma_x\sqrt{2\pi y}} \exp\left\{\frac{-y}{2\sigma_x^2}\right\} \tag{7.22}$$

Equation (7.22) is a chi-squared pdf with one degree of freedom.



Suppose a variable z is the sum of the square of two independent Gaussian random variables x and y with zero mean and an equal variance σ^2.

$$z = x^2 + y^2$$



$$f_x(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\{\frac{-x^2}{2\sigma^2}\}$$

$$f_y(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\{\frac{-y^2}{2\sigma^2}\}$$

The probability density function of z is easily obtained by convolution, for instance.

$$f_z(z) = \frac{1}{2\sigma^2} \exp\left\{\frac{-z}{2\sigma^2}\right\} \tag{7.23}$$

Equation (7.23) is a chi-squared pdf with two degrees of freedom. (See Figure 7.8) In general a random variable z would be sum of the squares of N independent Gaussian random variables.

$$z = x_1^2 + x_2^2 + x_3^2 + \cdots + x_N^2$$

When all x_i are Gaussian with zero mean and equal variance σ^2, the random variable z is said to have a probability density function that is chi-squared with N degrees of freedom. The probability density function of z is

$$f_z(z) = \frac{1}{\sigma^2 2^{N/2}\Gamma(N/2)} \left(\frac{z}{\sigma^2}\right)^{N/2-1} \exp\left\{\frac{-z}{2\sigma^2}\right\} \tag{7.24}$$

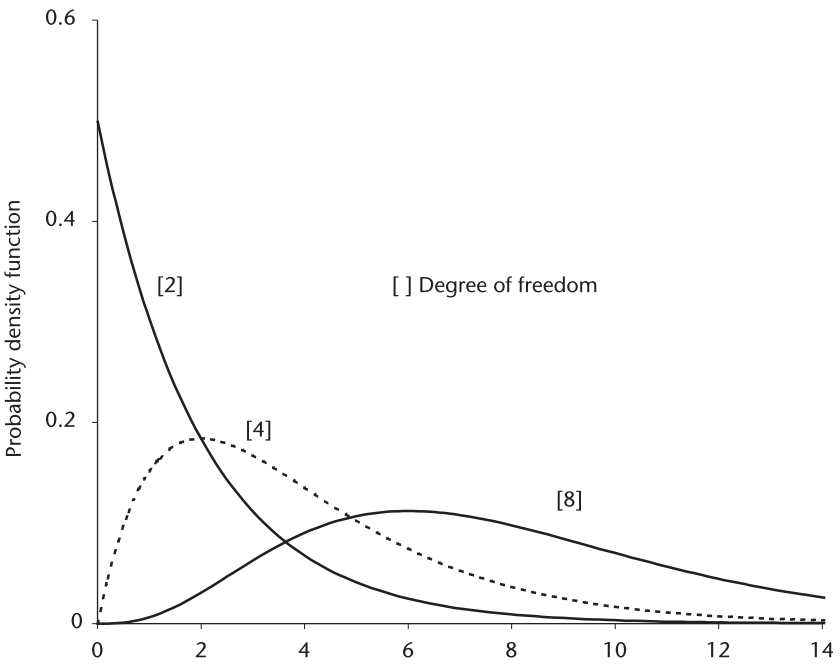


Figure 7.8 Chi-squared probability density function degree of freedom, 2, 4 and 8.

Chi-squared with four degrees of freedom is then

$$f_z(z) = \frac{z}{4\sigma^4} \exp\left\{\frac{-z}{2\sigma^2}\right\} \tag{7.25}$$

Equation 7.23 can be rewritten in a slightly different form when we substitute σ^2=x/2 in (7.23), and σ^2=x/4 in (7.25).

$$f_z(z) = \frac{1}{x} \exp\left\{\frac{-z}{x}\right\} \tag{7.26}$$

$$f_z(z) = \frac{4z}{x^2} \exp\left\{\frac{-2z}{x}\right\} \tag{7.27}$$

where

z: Signal-to-noise ratio in power;
x: Average of z over all target fluctuation.

Equation (7.26) is the probability density function of Swerling target models 1 and 2, and (7.27) is that of models 3 and 4.

7.3.1 Swerling Target Model 1

Swerling derived the characteristic function for model 1 from that of Marcum's nonfluctuating target for N pulses.

$$\begin{aligned} C_N(p) &= \int_0^\infty \frac{1}{x} \exp\left\{\frac{-z}{x}\right\} \frac{\exp\left\{-Nz\left(\frac{1}{p+1}\right)\right\}}{(p+1)^N} dz \\ &= \frac{1}{(p+1)^{N-1}[1+p(1+Nx)]} \end{aligned} \tag{7.28}$$

For a single pulse N=1,

$$C_1(p) = \frac{1}{1+p(1+x)} \tag{7.29}$$

The probability density function will be obtained by Fourier transform,

$$f_1(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} C_1(p) \exp\{pz\}\, dz = \frac{1}{1+x} \exp\left\{\frac{-z}{1+x}\right\} \tag{7.30}$$

Often we consult Fourier transform pairs prepared by Campbell and Foster [4]. Equations (7.29) and (7.30) are the pair given by the Campbell and Foster Eq. 438.

The detection probability for N=1 is found by integrating (7.30) over a proper limits.

$$P_d = \int_{y_b}^{\infty} f_1(z)\, dz = 1 - \int_0^{y_b} f_1(z)\, dz = \exp\left\{\frac{-y_b}{1+x}\right\} \tag{7.31}$$

The probability density function for N≥2 is obtained by,

$$f_N(z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} C_N(p) \exp\{-pz\}\, dz$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{1}{(p+1)^{N-1}[1+p(1+Nx)]} \exp\{-pz\}\, dz$$

The pair [581.7] of Campbell and Foster gives an expression for the probability density function.

$$\frac{1}{(p+\quad)(p+\quad)^{\quad-1}} \leftrightarrow \frac{1}{(\quad-1)(\quad+\quad)^{\quad-1}} \exp\{-\quad g\} \quad [\quad-1,(\quad-\quad)g]$$

$$= \frac{1}{(1+Nx)} \frac{1}{(\quad-1)\left(\frac{Nx}{1+Nx}\right)^{N-1}}$$

$$\times \exp\{-x/(1+Nx)\} \quad \left[\quad-1, \frac{Nxz}{1+Nz}\right]$$

$$= \frac{1}{Nx}\left(1+\frac{1}{Nx}\right)^{N-2} \exp\{-x/(1+Nx)\}$$

$$\times I\left[N-2, \frac{1}{\sqrt{N-1}} \frac{Nxz}{(1+Nx)}\right]$$

where I[·] is an Incomplete Gamma function of Pearson's form given by Abramowitz and Stegun [13], (Equation [6.5.6]).

The probability density function given above must be integrated over proper limits to obtain the detection probability. It is a daunting task indeed. A simpler way is found by Swerling by noting that

$$\frac{1}{(1+p)^{N-1}} = \frac{1}{(1+p)^{N-1}[1+p(1+Nx)]} + \frac{p(1+Nx)}{(1+p)^{N-1}[1+p(1+Nx)]}$$

$$= C_N(p) + p(1+Nx)C_N(p)$$

and,

$$\frac{z^{N-2}\exp\{-z\}}{(N-2)!} = f(z) + (1+Nx)f'(z)$$

The left-hand side is from the pair [431] of Campbell and Foster, and the differential on the right-hand side is from the pair [208] of Campbell and Foster. The detection probability is obtained by integrating both sides.

$$P_d = \int_{y_b}^{\infty} f_N(z)\, dz$$

$$= \int_{y_b}^{\infty} \frac{z^{N-2} \exp\{z\}}{(N-2)!}\, dz - (1+Nx) \int_{y_b}^{\infty} f'_N(z)\, dz \tag{7.32}$$

Since

$$P_d = 1 - \int_0^{y_b} f_N(z)\, dz$$

$$= 1 - \int_0^{y_b} \frac{z^{N-2} \exp\{-z\}}{(N-2)!}\, dz + (1+Nx)\, f_N(y_b)$$

Substituting

$$f_N(y_b) = \frac{1}{1+Nx} \frac{1}{\left[\frac{Nx}{1+Nx}\right]^{N-1}} \exp\{-y_b/(1+Nx)\}$$

$$\cdot I\left[N-2, \frac{y_b}{\sqrt{N-1}\left(1+\frac{1}{Nx}\right)}\right]$$

we obtain an expression of the detection probability P_d.

$$P_d = 1 - I\left[N-2, \frac{y_b}{\sqrt{N-1}}\right]$$

$$+ (1+Nx)^{N-1} I\left[N-2, \frac{y_b}{\sqrt{N-1}(1+Nx)}\right] \exp\left\{\frac{-y_b}{1+Nx}\right\}$$

$$= 1 - \sum_{k=N-1}^{\infty} \frac{\exp\{-y_b\}\, y_b^k}{k!}$$

$$+ \left(1+\frac{1}{Nx}\right)^{N-1} \sum_{k=N-1}^{\infty} \frac{\exp\{y_b/(1+Nx)\} \left[\frac{y_b}{(1+Nx)}\right]^k}{k!} \exp\left\{\frac{-y_b}{1+Nx}\right\} \tag{7.33}$$

Swerling found that both incomplete Gamma functions are very nearly unity for most cases of interest when P_{fa}=1.0E-6 or 1.0E-5. Therefore, two infinite summations could be replaced by unity. Then P_d would be given by

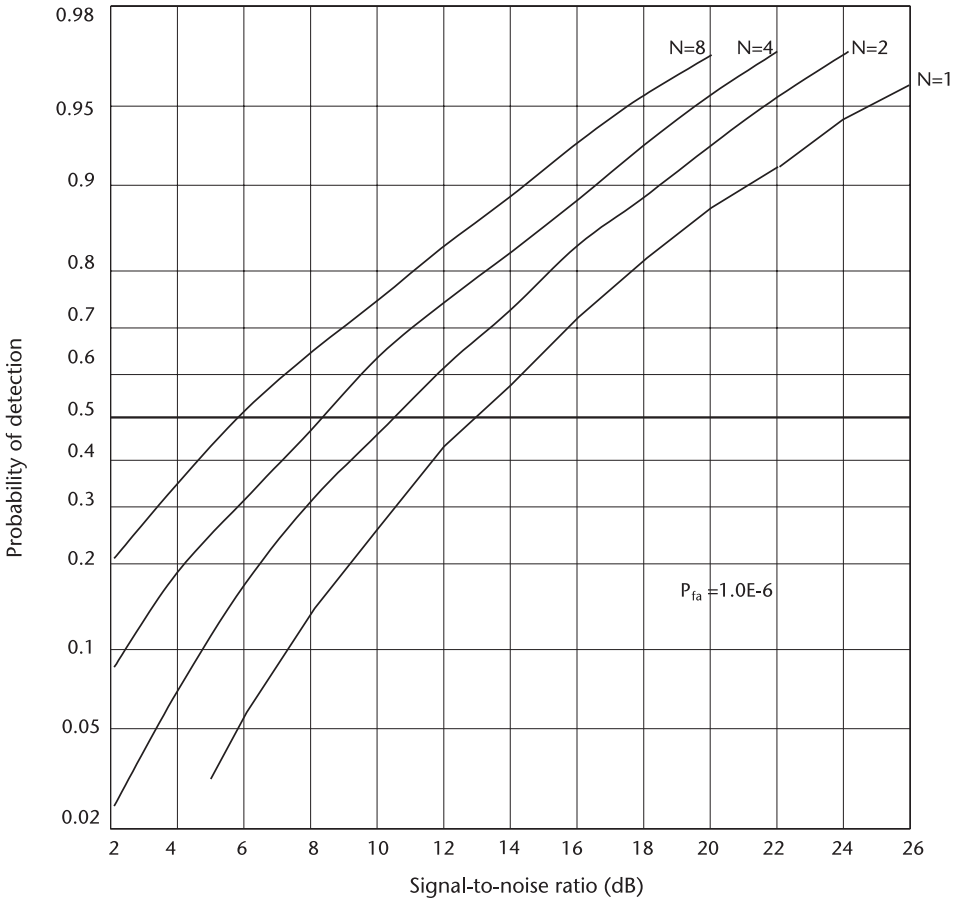


Figure 7.9 P_d versus SNR, Swerling target model 1. (For accurate reading, refer to corresponding DAT files.)

$$P_d = \left(1 + \frac{1}{Nx}\right)^{N-1} \exp\left\{\frac{-y_b}{1 + Nx}\right\} \tag{7.34}$$

Equation (7.31) for N=1 is programmed in PD_(1)_1.CPP. The (1) stands for Swerling target model 1, and '_1' stands for N=1. The data are stored in PD_(1)_1.DAT. The results are shown in Figure 7.9.

Equation (7.34) for N ≥ 2 is programmed in PD_(1)_N.CPP. The number of pulses integrated is N=2, 4, 8, 16 and 32. The data are stored in PD_(1)_N. DAT. The results are shown in Figure 7.9.

7.3.2 Swerling Target Model 2

The characteristic function for Swerling target model 2, N=1, is

$$C_1(p) = \int_0^\infty \frac{1}{x} \exp\left\{\frac{-z}{x}\right\} \frac{\exp\left\{-z\left(\frac{p}{p+1}\right)\right\}}{(p+1)}\, dz$$

$$= \frac{1}{1+p(1+x)} \tag{7.35}$$

The characteristic function for N≥2 is given by (7.35) raised to the Nth power. This is permissible since the return signals are assumed to be uncorrelated. For Swerling target model 1, we have to derive $C_N(p)$ first in order to obtain $C_1(p)$ since the return signals are correlated.

$$C_N(p) = \frac{1}{[1+p(1+x)]^N} \tag{7.36}$$

We note that the characteristic function of Swerling target 2, N=1, is identical to that of Swerling target 1, N=1. Thus, the detection probability of Swerling target 2, N=1 is identical to that of Swerling target 1, N=1.

The probability density function for N≥2 is given by pairs [431] of Campbell and Foster.

$$f_N(z) = \frac{1}{(1+x)^N(N-1)!} z^{N-1} \exp\left\{\frac{-z}{1+x}\right\} \tag{7.37}$$

Then

$$P_d = \int_{y_b}^\infty f_N(z)\, dz = 1 - \int_0^{y_b} f_N(z)\, dz$$

$$= 1 - \frac{1}{(1+x)^N} \int_0^{y_b} \frac{z^{N-1}\exp\{-z/(1+x)\}}{(N-1)!}\, dz$$

$$= 1 - I\left[\frac{y_b}{1+x}, N-1\right]$$

$$= 1 - \sum_{k=0}^{\infty} \frac{1}{k!} \exp\left\{\frac{-y_b}{1+x}\right\} \left(\frac{y_b}{1+x}\right)^k \tag{7.38}$$

Equation (7.38) is programmed in PD_(2)_N.CPP. The (2) stands for Swerling target model 2, and the "_N" stands for N pulses integrated. The infinite summation in k is terminated when the last incrementing term is less than 1.0E-9. The results are stored in PD_(2)_N.DAT, shown in Figure 7.10.

7.3.3 Swerling Target Model 3

The characteristic function of Swerling target model 3 is given by

$$C_N(p) = \frac{1}{(1+p)^{N-2}} \frac{1}{\left[1+p\left(1+\frac{Nx}{2}\right)\right]^2} \tag{7.39}$$

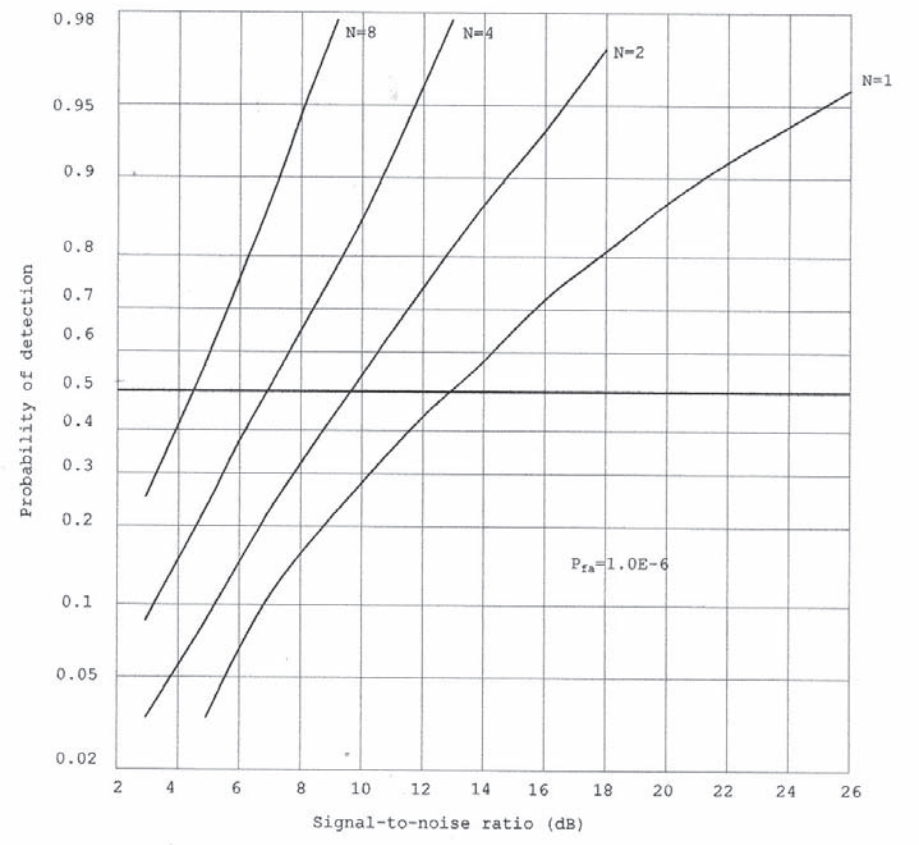


Figure 7.10 P_d versus SNR, Swerling target model 2. (Refer to DAT file for accurate reading.)

For N=1,

$$C_1(p) = \frac{1 + p}{\left[1 + p\left(1 + \frac{x}{2}\right)\right]^2} \tag{7.40}$$

For N=2,

$$C_2(p) = \frac{1}{[1 + p(1 + x)]^2} \tag{7.41}$$

The probability density function for N=1 is given by pairs [442] and [449.5] of Campbell and Foster. The pdf for N=2 is by pairs [442], and for N≥3 by pairs [581.1].

$$N = 1, \quad f(z) = \frac{1}{(1 + x/2)^2} \exp\left\{\frac{-z}{1 + x/2}\right\}\left(1 + \frac{z}{1 + 2/x}\right) \tag{7.42}$$

$$N = 2, \quad f(z) = \frac{z}{(1 + x/2)^2} \exp\left\{\frac{-z}{1 + x}\right\} \tag{7.43}$$

$$N \geq 3, \quad f(z) = \frac{1}{(1+Nx/2)^2(N-1)!} x^{N-1} \exp\{-z\} \; {}_1F_1\left[2; N; \frac{1}{1+2/Nx}\right] \quad (7.44)$$

where ${}_nF_m[\cdot]$ is a generalized hypergeometic function.

The probability of detection for N=1 and N=2 is obtained by direct integration.

$$N = 1, \qquad P_d = \left[1 + \frac{(x/2)y_b}{(1+x/2)^2}\right] \exp\left\{\frac{-y_b}{1+x/2}\right\} \quad (7.45)$$

$$N = 2, \qquad P_d = \left[1 + \frac{y_b}{1+x}\right] \exp\left\{\frac{-y_b}{1+x}\right\} \quad (7.46)$$

For N≥3, Swerling transformed the hypergeometric function to a combination of an exponential and two incomplete Gamma functions. Swerling found that the incomplete Gamma functions are nearly unity for the cases in which we are interested; P_{fa}=1.0E-6, and SNR=x is greater than 1.0 dB. He presented an approximate expression for N≥3.

$$P_d = \left(1 + \frac{2}{Nx}\right)^{N-2} \left[1 + \frac{y_b}{1+\frac{Nx}{2}} - \frac{2(N-2)}{Nx}\right] \exp\left\{\frac{-y_b}{1+\frac{Nx}{2}}\right\} \quad (7.47)$$

This is an exact equation when N=1 and N=2.

Myer and Mayer [9] obtained an exact expression for N≥2 by a contour integration in the complex plane through the residue theorem. It is a lengthy arduous path they have taken, Myer and Myer's appendix equation (A-85).

$$P_d = \frac{c y_b^{N-1} e^{-y_b}}{(N-2)!} + \sum_{k=0}^{N-2} \frac{y_b e^{-y_b}}{k!} + \frac{e^{-y_b}}{(1-c)^{N-2}} \left[1 - \frac{c(N-2)}{(1-c)} + c y_b\right] \left[1 - \sum_{k=0}^{N-1} \frac{1}{k!} (1-c)^k y_b^k e^{-y_b(1-c)}\right] \quad (7.48)$$

where c = 1/(1+Nx/2).

We found that the difference between Swerling's approximation (7.47) when N≥3 and Myer and Myer's exact equation (7.48) is practically nil for the range P_d >10% and P_{fa}=1.0E-6 through two separate programs.

Equation (7.47) is programmed in PD_(3)_1.CPP. The data are stored in PD_(3)_1. DAT. Myer and Mayer's exact equation is programmed in PD_(3)_N.CPP for N=2, 4, 6, 16, and 32. The results are stored in PD_(3)_N.DAT, shown in Figure 7.11.

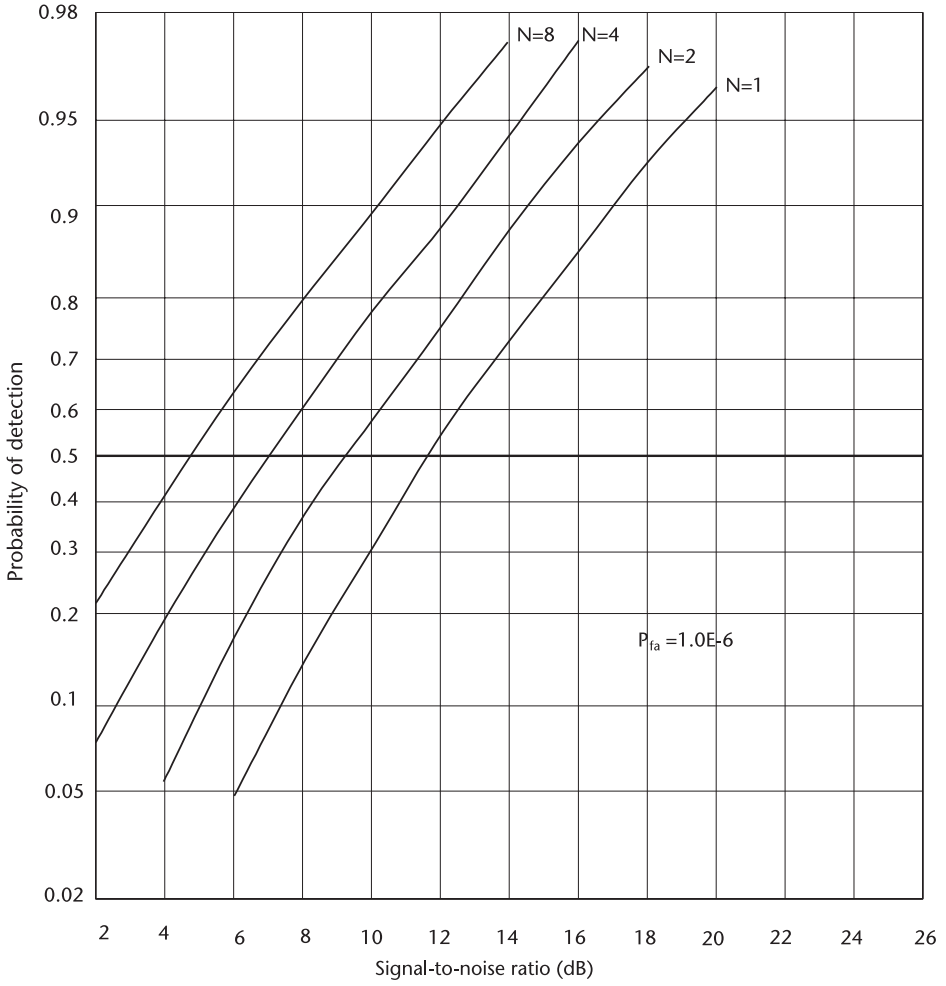


Figure 7.11 P_d versus SNR, Swerling target model 3.

7.3.4 Swerling Target Model 4

The characteristic functions for Swerling's target model 4 are given by,

$$C_1(p) = \frac{1+p}{[1+p(1+x/2)]^2} \tag{7.49}$$

$$C_N(p) = \frac{(1+p)^N}{[1+p(1+x/2)]^{2N}} \tag{7.50}$$

For N=1 the probability density function is identical to that of model 3, (7.42), and the detection probability (7.45).

For N≥2, Swerling employed the Gram-Charlier series or Edgeworth series to obtain the probability density function and integrated term by term to compute the detection probability. Since the Gram-Charlier is an infinite series, the last term to

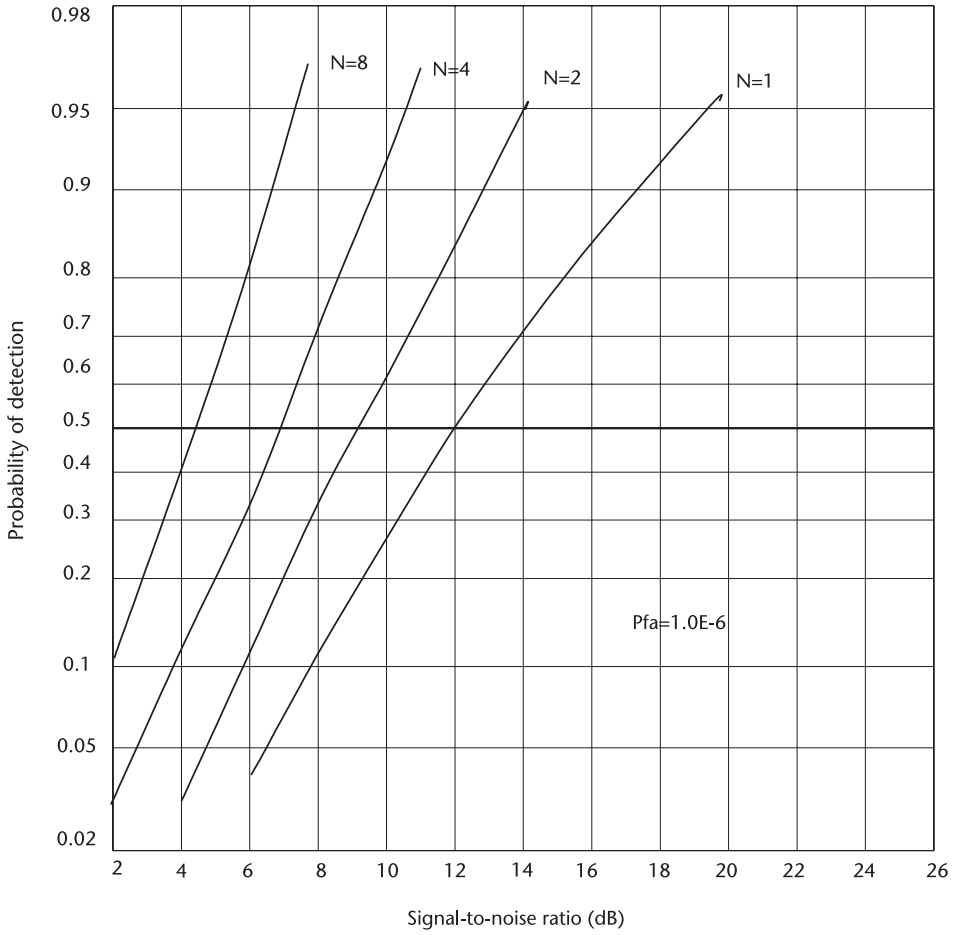


Figure 7.12 Pd versus SNR, Swerling target model 4.

be included must be less in magnitude comparable to the precision desired. It turned out that four terms of the series have been sufficient. Interested readers would benefit by consulting Helstrom [14] and DiFranco and Rubin [7].

Myer and Mayer obtained an exact expression for the detection probability for N≥2 through the method of contour integration in the complex plane via the theory of residue theorem. We do not repeat their derivation.

$$P_d = e^N \sum_{k=0}^{N} \frac{N!}{k!(N-k)!} \left(\frac{x}{2}\right)^{N-k} \sum_{r=0}^{2N-1-k} \frac{e^{-cy_b}(cy_b)^r}{r!} \tag{7.51}$$

where $c = \dfrac{1}{1+x/2}$.

Equation (7.51) is programmed in PD_(4)_N.CPP, shown in Figure 7.12.

7.4 Conclusion

The probability of detection for Marcum's and Swerling's target models with corresponding bias levels have been computed. Marcum's target is nonfluctuating, a constant cross-section for all observations.

Four Swerling target models are either Rayleigh or chi-squared fluctuating targets. The returned signals may be correlated (slow fluctuation) or completely uncorrelated (fast fluctuation). The expressions for the probability of detection of the five target models are listed in Table 7.2.

Another computation algorithm through the moment generation function in order to avoid a summation of infinite series has been reported with a fixed or adaptive bias level in [15]. We shall cover the adaptive detection problem in Chapter 10.

As more experimental data is accumulated and analyzed there appear to be other target models that suit better than the five models we have studied. A lognormal model and a Weibull model have been suggested; they may be either fluctuating or nonfluctuating, slow or fast. We shall cover Weibull target models in Chapter 10.

Swerling [16] states " . . . we must understand and establish the underlying true mechanism of physical scattering of an object so that resultant statistics of radar cross-section can be formulated." We do not wish to entertain any plausible target model merely because it is in an advanced textbook on probability and statistics. The study on radar target models is broad and expanding, and the explorative endeavors will continue [12, 17].

At the end of Section 7.2 we mentioned Albersheim's empirical formula for Marcum's target. His equation is very convenient for a preliminary system design. We demonstrate with an example. The basic radar equation of return signal power and the signal-to-noise ratio are

$$P_r = \frac{P_t G^2 \lambda^2 \sigma_o}{(4\pi)^3 R^4 \text{Loss}} \tag{7.52}$$

or

$$\frac{S}{N} = \frac{P_t G^2 \lambda^2 \sigma_o}{(4\pi)^3 R^4 \ \text{kTBw NF Loss}} \tag{7.53}$$

Where

P_r: Received power, watts;
P_t: Transmitted power, watts;
G: Antenna gain, dimentionless;
λ: Transmitted carrier wavelength, m;
σ_o: Target cross section, m^2;
R: Target range, m;
K: Boltzmann's constant, 1.3905E-23 Joule/K^o;
T: Absolute temperature of system in kelvin, K^o;
Bw: Predetection bandwidth, Hz;
Loss: Hardware plumbing loss and other losses;
NF: Noise figure of system.

Let us assign the following parameters for a preliminary system design:

P_r :	50.0 KW,	47.0 dBW;
G:	1000.0 (30 dB),	60.0 dB;
λ^2:	$(0.032\ m)^2$, X-band,	$-29.9\ dBm^2$;
σ_o:	$10.0\ m^2$,	$10.0\ dBm^2$;
$(4\pi)^3$:	1984.4,	32.98 dB;
K:	1.38054E-23 J/K°;	
T:	300.0 K°, 27 C°;	
Bw:	1.0 MHz;	
KTBw:	4.14E-15,	-143.83 dBW;
Loss:	-3.0 dB;	
NF:	-6.0 dB.	

From Albersheim's equation (7.20) or graph, Figure 7.7, we obtain the following:

- S/N required: Single pulse (13.11 dB)

$$P_d=0.9,\ P_{fa}=1.0E\text{-}6$$

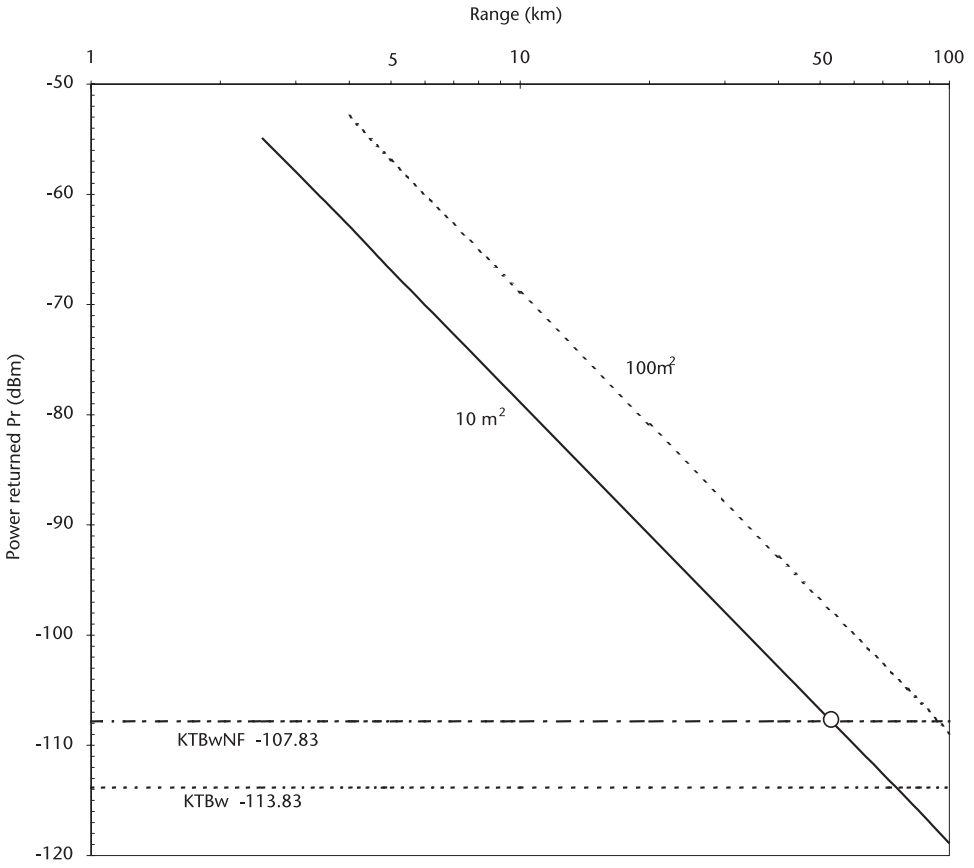


Figure 7.13 Computation of detection range.

- S/N required: Two pulses (10.46 dB)
- S/N required: Four pulses (7.96 dB)
- S/N required: Eight pulses (5.68 dB)

The power returned P_r is plotted in Figure 7.13. The slope of the returned power is –40 dB per decade. The noise power level is indicated by the dashed line. S/N=0.0 dB where the power line intercepts the KTBw·NF is marked by an open circle.

The maximum detection range where S/N must be at least 5.68 dB, eight pulses noncoherently integrated, is computed by summing the furthest right most column.

$$R^4 = 183.27 \text{ dBm}^4, \quad R = 45.82 \text{ dBm}, \quad R = 38.2 \text{ km}$$

Table 7.2 Characteristic Functions of Five Target Models

Target Model	*N=1*	*N≥2*
0	$C_1(p) = \dfrac{\exp\{-x\}\ \exp\left\{\dfrac{x}{1+p}\right\}}{1+p}$	$C_N(p) = \dfrac{\exp\{-Nx\}\ \exp\left\{\dfrac{Nx}{1+p}\right\}}{(1+p)^N}$
1	$C_1(p) = \dfrac{1}{1+p(1+x)}$	$C_N(p) = \dfrac{1}{(1+p)^{N-1} + [1+p(1+Nx)]}$
2	$C_1(p) = \dfrac{1}{1+p(1+x)}$	$C_N(p) = \dfrac{1}{[1+p(1+x)]^N}$
3	$C_1(p) = \dfrac{1}{[1+p(1+x/2)]^2}$	$C_N(p) = \dfrac{1}{(1+p)^{N-2}[1+p(1+Nx/2)]^2}$
4	$C_1(p) = \dfrac{(1+p)}{[1+p(1+x/2)]^2}$	$C_N(p) = \dfrac{(1+p)^N}{[1+p(1+x/2)]^{2N}}$

N: Number of pulses noncoherently integrated;
x: Average signal-to-noise ratio, in power;
p: Fourier transform variable, $j\omega$.

$$\begin{cases} C(j\ ) = \displaystyle\int_{\infty}^{-\infty} f(x)e^{j\ x}\,dx \\ f(x) = \dfrac{1}{j2}\displaystyle\int_{\infty}^{-\infty} C(j\ )e^{-j\ x}\,d \end{cases} \qquad \begin{cases} C(p) = \displaystyle\int_{\infty}^{-\infty} f(x)e^{pz}\,dx \\ f(z) = \dfrac{1}{2}\displaystyle\int_{\infty}^{-\infty} C(p)e^{-pz}\,dp \end{cases}$$

Table 7.3 The Probability of Detection

Target Model	*Probability of Detection*	
0	$P_d = \sum_{k=0}^{\infty} \frac{e^{-Nx}(Nx)^k}{k!} \sum_{r=0}^{N-1+k} \frac{e^{-y_b} y_b^r}{r!}$ for all N	for all N
1	$P_d = \exp\left\{\frac{-y_b}{1+x}\right\}$ $N = 1$	N=1
	$P_d = 1 - \sum_{k=N-1}^{\infty} \frac{\exp\{-y_b\} y_b^k}{k!} + \left(1+\frac{1}{Nx}\right)^{N-1} \sum_{k=N-1}^{\infty} \frac{\exp\{y_b/(1+Nx)\} \left[\frac{y_b}{(1+Nx)}\right]^k}{k!} \cdot \exp\left\{\frac{-y_b}{1+Nx}\right\}$ $N \geq 2$	N≥2
	$P_d = \left(1+\frac{1}{Nx}\right)^{N-1} \exp\left\{\frac{-y_b}{1+Nx}\right\}$ $N \geq 2$	N≥2
2	$P_d = \exp\left\{\frac{-y_b}{1+x}\right\}$ $N = 1$	N=1
	$P_d = 1 - \sum_{k=0}^{\infty} \frac{1}{k!} \exp\left\{\frac{-y_b}{1+x}\right\} \left(\frac{y_b}{1+x}\right)^k$ $N \geq 2$	N≥2
3	$P_d = \left[1 + \frac{(x/2)y_b}{(1+x/2)^2}\right] \exp\left\{\frac{-y_b}{1+x/2}\right\}$ $N = 1$	N=1
	$P_d = \left[1 + \frac{y_b}{1+x}\right] \exp\left\{\frac{-y_b}{1+x}\right\}$	
	$P_d = \left(1+\frac{2}{Nx}\right)^{N-2} \left[1 + \frac{y_b}{1+\frac{Nx}{2}} - \frac{2(N-2)}{Nx}\right] \cdot \exp\left\{\frac{-y_b}{1+\frac{Nx}{2}}\right\}$ $c = 1/(1+Nx/2)$ $N \geq 2$	N≥2
4	$P_d = \left[1 + \frac{(x/2)y_b}{(1+x/2)^2}\right] \exp\left\{\frac{-y_b}{1+x/2}\right\}$ $N = 1$	N=1
	$P_d = e^N \sum_{k=0}^{N} \frac{N!}{k!(N-K)!} \left(\frac{x}{2}\right)^{N-k} \sum_{r=0}^{2N-1-k} \frac{e^{-cy_b}(cy_b)^r}{r!}$ $N \geq 2$	
	$c = 1/(1+x/2)$	N≥2

N: Number of pulses noncoherently integrated;

x: Signal-to-noise ratio in power;

y_b: Bias level required to maintain a specified P_{fa}.

List of Programs

Program	*Features*
(1) PULSEDET.CPP	Pulse detection by recirculating accumulator
(2) BIAS_YB.CPP	Computes bias level, N=2 to N=100
(3) PD_(0)_1.CPP	Detection probability, Marcum target 0, N=1
(4) PD_(0)_N.CPP	Detection probability, Marcum target 0, N≥2
(5) PD_(1)_1.CPP	Detection probability, Swerling target 1, N=1
(6) PD_(1)_N.CPP	Detection probability, Swerling target 1, N≥2
(7) PD_(2)_1.CPP	(Did not program) Identical to PD_(1)_1.CPP
(8) PD_(2)_N.CPP	Detection probability, Swerling target 2, N≥2
(9) PD_(3)_1.CPP	Detection probability, Swerling target 3, N=1
(10) PD_(3)_N.CPP	Detection probability, Swerling target 3, N≥2
(11) PD_(4)_1.CPP	(Did not program) identical to PD_(3)_1.CPP
(12) PD_(4)_N.CPP	Detection probability, Swerling target 4, N≥2
(13) DET_SNR.CPP	Albersheim's detection equation
(14) BINOMIAL.H	Header file, computes binomial coefficients
(15) FACTORIA.H	Header file, computes factorial, N!
(16) BIAS_YB.H	Header file, computed bias level y_b

References

[1] Marcum, J. I. "A Statistical Theory of Target Detection by Pulsed Radar," RM-754, Dec. 1947, and Mathematical Appendix, RM-753, Jul. 1948, The Rand Corporation, Santa Monica, Calif. declassified and reprinted in IRE, *Trans. IT*, Vol. 6, Apr. 1960.

[2] Brennan, L. E. and I. S Reed, "A Recursive Method of Computing the Q Function," IEEE, *Trans. IT*, Vol. 6, Apr. 1965.

[3] Papoulis, A., *Probability, Random Variables, and Stochastic Processes,* New York, N.Y.: McGraw-Hill, 1965.

[4] Campbell, G. E. and R. M., Foster, *Fourier Integrals for Practical Applications,* Parsippany, N.J.: Van Nostrand, 1947.

[6] Knopp, K., *Theory and Application of Infinite Series,* Minola, N.Y.: Dover Publications, 1990.

[7] DiFranco, J. D., and W. Rubien, *Radar Detection,* Dedham, MA: Artech House, 1980.

[8] Pachares, J., "A Table of Bias Levels Useful in Radar Detection Problems," IRE, *Trans. IT*, Vol. 4, No.1, Mar. 1958, pp. 38–45.

[9] Myer, D. P. and H. A. Mayer, *Radar Target Detection,* New York, N.Y.: Academic Press, 1973.

[10] Albersheim, W. J., "A Closed-Form Approximation to Robertson's Detection Characteristics," *Proc. IEEE*, Jul. 1981.

[11] Tufts, D. W., and A. J. Cann, "On Albersheim's Detection Equation," IEEE, *Trans. AES*, Vol. 19, No. 4, Jul. 1983.

[12] Johnston, S. L., "Target Fluctuation Models for Radar System Design and Performance Analysis: An Overview of Three Papers," IEEE, *Trans. AES*, Vol. 3, No. 2, Apr. 1997.

[13] Abramowitz, M., and I. A. Stegun, *Handbook of Mathematical Functions with Formula, Graphs and Mathematical Tables,* Natl. Bureau of Standards, Series 55, 1964.

[14] Helstrom, C. W., *Statistical Theory of Signal Detection,* 2nd ed., New York, N.Y.: Pergamon Press, 1988.

[15] Hou, X-Y., N. Morinaga, and T. Namekawa, "Direct Evaluation of Radar Detection Probabilities," IEEE, *Trans. AES*, Vol. 23, No. 4, Jul. 1987.

[16] Swerling, P., "Recent Developments in Target Models for Radar Detection Analysis," *AGARD Avionics Technical Symposium*, May 1970.

[17] Barton, D. K., "Simple Procedures for Radar Detection Calculations," IEEE, *Trans AES*, Vol. 5, No. 5, Sept. 1969.

Selected Bibliography

Rihaczek, A.W., *Principles of Higher Resolution Radar,* Norwood, MA: Artech House, 1996.

Shnidman, D. A., "Radar Detection Probabilities and Their Calculation," IEEE, *Trans. AES*, Vol. 31, Jul. 1995.

Shnidman, D. A., "Evaluation of Probability of Detection for Several Target Fluctuating Models," *Tech. Note 1975-35*, Lincoln Lab., MIT, 1975.

Swerling, P., "Probability of Detection for a Fluctuating Target," The Rand Corporation, Mar. 1954, declassified and reprinted in IRE, *Trans. IT*, Vol. 6, Apr. 1960.

CHAPTER 8

Kalman Filter

8.1 Introduction

The term Kalman filter stirs excitement in the heart of every student of engineering and science. Kalman's epoch-breaking solution [1] to the problems of discrete state estimation has been applied successfully to a wide and diverse area including aerospace and marine navigation, power grid control, plant operation control, biomedical sample cultivation, prediction of demographic distribution, and even crop harvest forecast.

Scores of books and hundreds of research reports have been published, some theoretical, some applied, and others in between. This chapter is written for those students and engineers who plan to broaden their knowledge of Kalman filters in a quick order.

Estimation is a process of extracting information from available measurements, though the measurements are invariably contaminated with errors. The most important information we wish to extract from the measurements may be a prediction, such as the future position of an aircraft or ocean liner or the future power demand and the future plant control required to produce goods with specified tolerance.

The object of this chapter is to provide practical working familiarization in the subject of estimation, prediction, and computation algorithms that extract the best information from contaminated measurements.

An implementation of the Kalman filtrering process on a computer illuminates the importance of the finite word length of computing machines. The finite word length causes a loss of precision through round-off error, destruction of symmetry in a matrix, accumulation of error build-up, and so on. We sometimes encounter a numerically unstable algorithm that leads to an utterly meaningless result, even though the supporting theoretical equations are perfect.

Authors of numerous books and research reports have used symbols and notations of bewildering varieties that cause confusion and discouragement to beginners. We rectify this problem at the outset. We shall adhere to the following two rules in the symbols and notations. The state under investigation is discretized in time as past, present and future as shown in Figure 8.1, k–1, k and k+1, the immediate past, present and immediate future, respectively. The notation $\mathbf{x}_k$ denotes the present state in vector form. The overhead triangle caret (^) signifies an estimate of $\mathbf{x}_k$ based on all the past information. The notation $\mathbf{x}_{k+1}$ with a caret is an estimate of future states based on all the information including $\mathbf{x}_k$.

We introduce an estimate with an overhead tilde $\tilde{\mathbf{x}}_k$. This notation denotes an error between the true state $\mathbf{x}_k$, unknowable to observer, and an estimate $\hat{\mathbf{x}}_k$. We

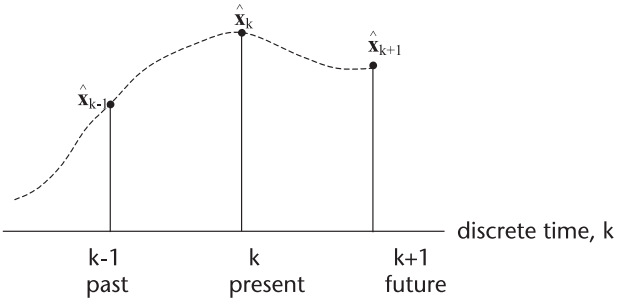


Figure 8.1 Estimate of state: past, present and future.

will never know the true state $\mathbf{x}_k$; we only surmise what it could be by the magnitude of $\tilde{\mathbf{x}}_k$; the smaller the $\tilde{\mathbf{x}}_k$, the closer $\hat{\mathbf{x}}_k$ is to the true state $\mathbf{x}_k$.

$$\tilde{\mathbf{x}}_k = \hat{\mathbf{x}}_k - \mathbf{x}_k \tag{8.1}$$

The Kalman filter is often called a recursive filter. Recursive filtering is a process that is not required to store all the past measurements to estimate the present state and predict the future state. An example is given to convey the concept of a recursive filter, or a recursive algorithm.

Suppose we have a lathe that produces precision machine screws of exactly one inch in length. Imagine we have produced 100 screws. Due to fatigue in the lathe and the nonuniformity of the steel rod from which the screws are made there will be a few screws ever so slightly longer or shorter than one inch. We wish to compute the accurate average length of the screws. We would do the following:

$$\hat{\mathbf{x}}_k = \frac{1}{k}\sum_{i=1}^{k} \mathbf{z}_i \tag{8.2}$$

where $\mathbf{z}_i$ is the measurement of each screw with precision micrometer. We would record all the measured lengths, 100 of them, sum them up and take the average. Suppose further that the lathe takes one minute to produce one screw. We would have to wait one hour and 40 minutes to produce 100 screws and need sometime to compute the average. The measurements may have unknown errors as the result of turning the knuckles too tightly or too loosely. The measurement uncertainty can be formulated as follows:

$$\mathbf{z}_k = \mathbf{x}_k + \mathbf{v}_k \tag{8.3}$$

where $\mathbf{z}_k$ is measured length, $\mathbf{x}_k$ is true length, and $\mathbf{v}_k$ is measurement error. When we say this screw is one inch long, we imply that the best estimated length of the screw is one inch. Back to the problem of one hour and 40 minutes of waiting. When the hundredth screw is produced the average length of screws is given by

$$\hat{\mathbf{x}}_k = \frac{1}{k}\sum_{i=1}^{k} \mathbf{z}_i \tag{8.4}$$

The expression above can be manipulated to separate the prior estimate, $\hat{\mathbf{x}}_k$, based on the average of 99 screws and the measurement of the hundredth screw.

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \frac{1}{k+1}\left[\sum_{i=1}^{k}\mathbf{z}_i + z_{k+1}\right] \\ &= \frac{k}{k+1}\left[\frac{1}{k}\sum_{i=1}^{k}\mathbf{z}_i\right] + \frac{1}{k+1}z_{k+1} \\ &= \frac{k}{k+1}\hat{\mathbf{x}}_k + \frac{1}{k+1}z_{k+1} \end{aligned} \tag{8.5}$$

Thus, (8.5) eliminates the storage (recording) problem and the waiting time, as long as the previous estimate $\hat{\mathbf{x}}_k$ has been retained. In light of this recursive concept, we refine Figure 8.1 as shown in Figure 8.2.

A few words on the superscripts (–) and (+). $\hat{\mathbf{x}}_k^-$ is an estimate based on all information available just before the measurement $\mathbf{z}_k$ is made. This is the a priori estimate and is the embodiment of all the past data, $\mathbf{x}_{k-1}$, $\mathbf{z}_{k-1}$ and $\mathbf{x}_{k-1}^+$. On the other hand, $\mathbf{x}_k^+$ is the a posteriori estimate just after the present measurement $\mathbf{z}_k$ is taken. $\mathbf{x}_{k+1}^-$ is a prediction based on all the past data plus the present measurement $\mathbf{z}_k$. All the vectors $\mathbf{x}$ shown in Figure 8.2 may have an overhead caret(^) or overhear tilde (~).

For a moment, imagine you stand at point k–1 in time. $\mathbf{x}_k^-$ is really a prediction based on all the information you have up to point k–1. Next, shift your position to the kth time (present). $\mathbf{x}_{k+1}^-$ is a predicted future state based on all the information you have at the kth time, $\mathbf{x}_k^-$, $\mathbf{z}_k$ and $\mathbf{x}_k^+$. The $\mathbf{x}_k$ with superscript (–) is a prediction made after the measurement $\mathbf{z}_{k-1}$ and the superscript (+) is an estimate immediately after the measurement $\mathbf{z}_k$. Incidentally $\mathbf{x}_{k+1}^+$, a dashed line at the extreme right, is a meaningless notation in as much as we cannot estimate the state without knowing the future measurement $\mathbf{z}_{k+1}$. We adhere to these two rules and notations throughout our discussions: overhead carret (^) and tilde (~), superscript (–) and (+).

This chapter presents several "real-world" programs, starting with an easier program followed by progressively more difficult ones. Some authors present seven

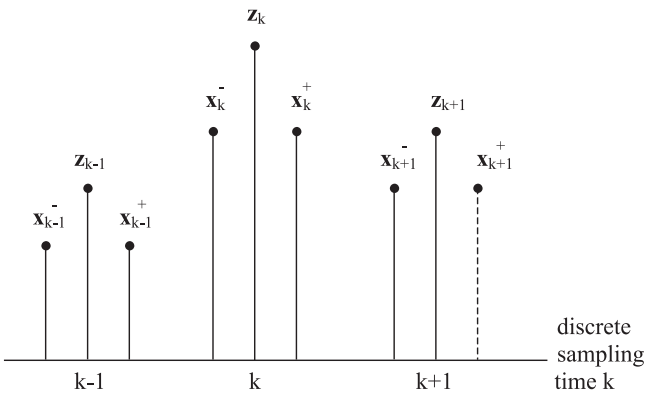


Figure 8.2 Estimate of $\mathbf{x}_k$ before and after measurement $\mathbf{z}_k$.

Kalman equations at the onset with unfamiliar symbols and notations. We prefer to avoid this practice. We shall present seven equations in an orderly sequence with numerical examples when necessary for the uninitiated.

8.2 Derivation of Kalman Filter Equations

Refer to Figure 8.3. A passenger airliner has a smooth flight plan. The pilot plans to fly at a constant velocity at a constant altitude on a straight path. Due to nonconstant engine thrust and nonuniform atmospheric conditions, the passenger airliner would traverse at varying velocities and an off-straight-line path.

The position of the aircraft can be expressed in the Cartesian coordinate system (CCS) as follows. The overhead single dot ($\dot{}$) is velocity and two dots ($\ddot{}$) acceleration. T is the discrete time measurement interval in seconds.

$$\begin{cases} x_{k+1} = x_k + \dot{x}_k T + 1/2\ddot{x}_k T^2 \\ \dot{x}_{k+1} = \dot{x}_k + \ddot{x}_k T \\ y_{k+1} = y_k + \dot{y}_k T + 1/2\ddot{y}_k T^2 \\ \dot{y}_{k+1} = \dot{y}_k + \ddot{y}_k T \end{cases} \tag{8.6}$$

Recast the above Newtonian motion equations into a more compact form with a_x and a_y, representing accelerations in x and y axis.

$$\begin{bmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ \dot{x}_k \\ y_k \\ \dot{y}_k \end{bmatrix} + \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \tag{8.7}$$

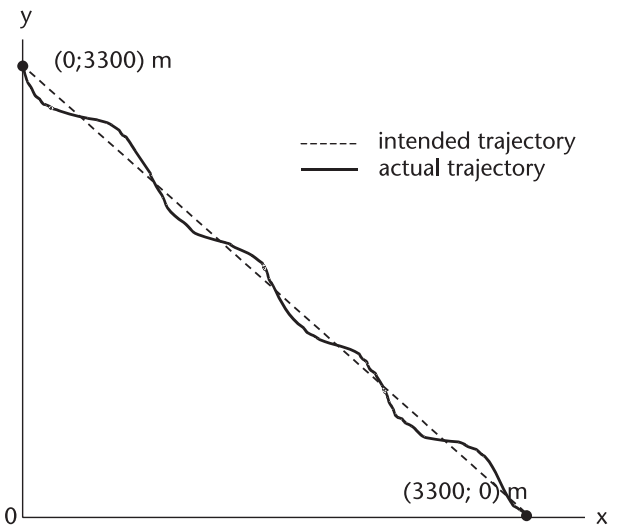


Figure 8.3 Flight trajectory.

Equation (8.7) is still cumbersome. Can we write it the following vector-matrix form?

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Gamma}\mathbf{w}_k \tag{8.8}$$

where

$$\mathbf{\Phi} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{\Gamma} = \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \quad \mathbf{w}_k = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

Equation (8.8) is the first of Kalman's seven equations and is called the state equation (the state of , or the position and the motion of, the passenger airliner).

The disturbance $\mathbf{w}_k$ is assumed to be distributed as Gaussian with mean zero and variance σ^2. The covariance matrix $\mathbf{Q}_k$ is given by,

$$\begin{aligned} \mathbf{Q}_k &= E\{\mathbf{w}_k, \mathbf{w}_k^T\} = E\left\{ \begin{bmatrix} a_x \\ a_y \end{bmatrix} \begin{bmatrix} a_x & a_y \end{bmatrix} \right\} \\ &= E\left\{ \begin{bmatrix} a_x^2 & a_x a_y \\ a_y a_x & a_y^2 \end{bmatrix} \right\} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \end{aligned}$$

We assumed that the accidental acceleration perturbations a_x and a_y at each measurement are completely uncorrelated and are small in magnitude, say ± 0.5g, uniformly distributed. (1g = 9.8 m/sec^2). (See Figure 8.4.) The expectation $E\{a_x a_y\}$ is identically zero, for a_x and a_y are completely uncorrelated. The variance of the uniformly distributed disturbance is given by,

$$\sigma_x^2 = \sigma_y^2 = \frac{1}{12}[4.9 - (-4.9)]^2 = 8.0 \text{ (m/sec}^2)^2$$

The passenger airliner's position is observed (or measured) by a tracking radar located at the origin. The radar will report the distance r_k and the angle θ_k in the line-of-sight (LOS) coordinate system.

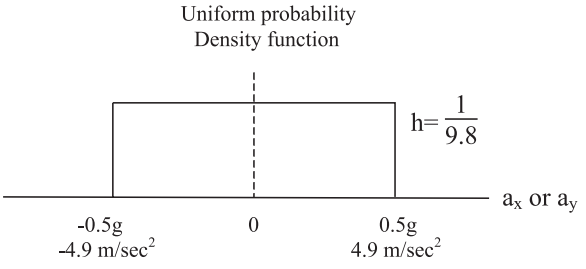


Figure 8.4 Acceleration perturbation distribution.

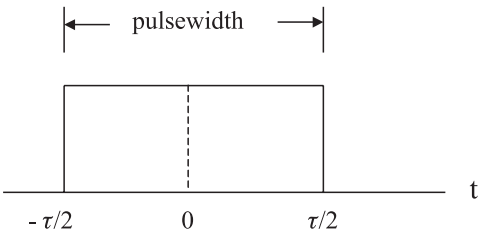


Figure 8.5 Finite pulsewidth and range measurement error.

$$\begin{cases} \hat{r}_k = r_k + n_r \\ \hat{\theta}_k = \theta_k + n_\theta \end{cases} \tag{8.9}$$

where n_r and n_θ are the errors in the range and angle measurements. The error in the range measurement is due to a finite transmitter pulsewidth. The true target position may be at the leading edge of the pulse, at the trailing edge, or somewhere between. (See Figure 8.5.)

The measurement error in angle is due to a finite antenna beamwidth. The target may be located at the upper edge of the beam, at the lower edge of the beam, or somewhere within the –3dB beamwidth, as shown in Figure 8.6.

We cast (8.9) in a vector-matrix form as we have done previously.

$$\begin{bmatrix} r_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_k \\ \dot{r}_k \\ \theta_k \\ \dot{\theta}_k \end{bmatrix} + \begin{bmatrix} n_r \\ n_\theta \end{bmatrix} \tag{8.10}$$

and further we write in a compact form,

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{8.11}$$

where

$$\mathbf{z}_k = \begin{bmatrix} r_k \\ \theta_k \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{v}_k = \begin{bmatrix} n_r \\ n_\theta \end{bmatrix}$$

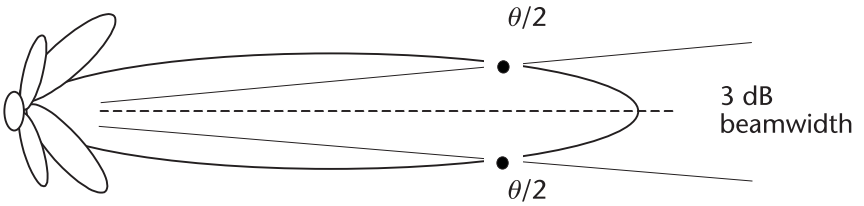


Figure 8.6 Antenna beamwidth and angle measurement error.

Equation (8.11) is the second equation of Kalman's seven. It is called the measurement equation. A numerical example of the covariance matrix $\mathbf{R}_k$ of $\mathbf{v}_k$ will be presented shortly.

Derivation of the Kalman gain

Please look at Figure 8.2. Suppose you are at time k. You see $\mathbf{x}_k^-$, $\mathbf{z}_k$, and $\mathbf{x}_k^+$ where $\mathbf{x}_k^-$ is the a priori estimate based on all information available at k-1; $\mathbf{z}_k$ is the present measurement; and $\mathbf{x}_k^+$ is the a posteriori estimate after $\mathbf{z}_k$ is taken. The a posteriori estimate can be expressed as

$$\mathbf{x}_k^+ = \mathbf{k}_k' \mathbf{x}_k^- + \mathbf{k}_k \mathbf{z}_k \tag{8.12}$$

where $\mathbf{k}_k'$ and $\mathbf{k}_k$ are yet to be determined. Equation (8.12) is identical in form to (8.5), the recursive estimation. We substitute the following two expressions into (8.12). One is the a priori estimate error and other the a posteriori estimate error.

$$\tilde{\mathbf{x}}_k^- = \mathbf{x}_k - \mathbf{x}_k^- \tag{8.13}$$

$$\tilde{\mathbf{x}}_k^+ = \mathbf{x}_k - \mathbf{x}_k^+ \tag{8.14}$$

Substituting (8.11), (8.13), and (8.14) into (8.12) yields,

$$\tilde{\mathbf{x}}_k^+ = [\mathbf{I} - \mathbf{k}_k' - \mathbf{k}_k \mathbf{H}]\mathbf{x}_k + \mathbf{k}_k' \tilde{\mathbf{x}}_k^- + \mathbf{k}_v \mathbf{v}_k \tag{8.15}$$

Some remarks on (8.15) follow: By definition, $E\{\mathbf{v}_k\} = 0$, that is, the means of the range and angle errors are identically zero. The second term involves $\tilde{\mathbf{x}}_k^-$, the a priori estimate error. Had we been correct in our computation in the past, this term would have been zero. Therefore, in order for the a posteriori estimate error $\tilde{\mathbf{x}}_k^+$ to be zero, the bracketed term must be a null matrix.

$$\mathbf{k}_k' = \mathbf{I} - \mathbf{k}_k \mathbf{H}$$

Substitution of the above results in (8.12) leads to an updated estimate.

$$\mathbf{x}_k^+ = [\mathbf{I} - \mathbf{k}_k \mathbf{H}]\, \mathbf{x}_k^- + \mathbf{k}_k \mathbf{z}_k$$

or

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{k}_k\, [\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-] \tag{8.16}$$

Equation (8.16) is the third equation of Kalman's seven. We don't know exactly what $\mathbf{k}_k$ is yet, but when we do, an updated estimate $\mathbf{x}_k^+$ can be computed from $\mathbf{x}_k^-$.

In order to obtain an expression for $\mathbf{k}_k$, Kalman gain matrix, we consider the a posteriori error covariance matrix $\mathbf{P}_k^+$defined below and force the error covariance matrix to a null matrix.

$$\mathbf{P}_k^+ = E\{\, \tilde{\mathbf{x}}_k^+ (\tilde{\mathbf{x}}_k^+)^T\} \rightarrow [0]$$

The residual error in (8.15) when $\mathbf{k}_k' = \mathbf{I} - \mathbf{k}_k\mathbf{H}$ is given by

$$\tilde{\mathbf{x}}_k^+ = [\mathbf{I} - \mathbf{k}_k\mathbf{H}\,]\,\tilde{\mathbf{x}}_k^- + \mathbf{k}_k\mathbf{v}_k$$

Substituting the above result in the expression for $\mathbf{P}_k^+$ we obtain,

$$\mathbf{P}_k^+ = \mathrm{E}\{[(\mathbf{I} - \mathbf{k}_k\mathbf{H})\tilde{\mathbf{x}}_k^- + \mathbf{k}_k\mathbf{v}_k][(\mathbf{I} - \mathbf{k}_k\mathbf{H})\tilde{\mathbf{x}}_k^- + \mathbf{k}_k\mathbf{v}_k]^{\mathrm{T}}\}$$

Expanding the expectation operation $\mathrm{E}\{[\cdot][\cdot]^{\mathrm{T}}\}$ using the following definition, we finally obtain an expression for the a posteriori error covariance matrix:

$$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{k}_k\mathbf{H}]\mathbf{P}_k^-[\mathbf{I} - \mathbf{k}_k\mathbf{H}]^{\mathrm{T}} + \mathbf{k}_k\mathbf{R}_k\mathbf{k}_k^{\mathrm{T}} \tag{8.17}$$

where

$$\begin{aligned}
\mathrm{E}\left\{\tilde{\mathbf{x}}_k^-(\tilde{\mathbf{x}}_k^-)^{\mathrm{T}}\right\} &= \mathbf{P}_k^- \\
\mathrm{E}\left\{\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}\right\} &= \mathbf{R}_k \\
\mathrm{E}\left\{\mathbf{v}_k(\tilde{\mathbf{v}}_k^-)^{\mathrm{T}}\right\} &= 0 \\
\mathrm{E}\left\{\tilde{\mathbf{v}}_k^-\mathbf{v}_k^{\mathrm{T}}\right\} &= 0
\end{aligned}$$

Equation (8.17) is the fourth equation of Kalman's seven. Next we want to find $\mathbf{k}_k$ such that $\mathbf{P}_k^+$ will be a null matrix. We take a partial derivative of $\mathbf{P}_k^+$ with respect to $\mathbf{k}_k$ and set it to a null matrix. We use the following identity.

$$\frac{\partial}{\partial \mathbf{A}}\mathrm{trace}\,[\mathbf{A}\mathbf{B}\mathbf{A}^{\mathrm{T}}] = \mathbf{A}\mathbf{B} + \mathbf{A}\mathbf{B}^{\mathrm{T}} = 2\mathbf{A}\mathbf{B}$$

The last identity is valid only when $\mathbf{B}$ is symmetric [2, 3].

$$-2(\mathbf{I}-\mathbf{k}_k\mathbf{H})\,\mathbf{P}_k^-\mathbf{H}^{\mathrm{T}} + 2\mathbf{k}_k\mathbf{R}_k = \mathbf{0}$$

Solving for $\mathbf{k}_k$, the Kalman gain matrix is given by

$$\mathbf{k}_k = \mathbf{P}_k^-\mathbf{H}^{\mathrm{T}}[\,\mathbf{H}\,\mathbf{P}_k^-\mathbf{H}^{\mathrm{T}} + \mathbf{R}_k\,]^{-1} \tag{8.18}$$

Equation (8.18) is the fifth equation of Kalman's seven. We shall show in a flow diagram, Figure 8.7, where the Kalman gain matrix should be placed in the recursive loop.

Substituting (8.18) into (8.17) and after some matrix manipulation we obtain another expression for $\mathbf{P}_k^+$:

$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{P}_k^-\mathbf{H}^{\mathrm{T}}\,[\,\mathbf{H}\,\mathbf{P}_k^-\mathbf{H}^{\mathrm{T}} + \mathbf{R}_k\,]^{-1}\,\mathbf{H}\,\mathbf{P}_k^-$$

or

$$\mathbf{P}_k^+ = [\,\mathbf{I} - \mathbf{k}_k\mathbf{H}]\,\mathbf{P}_k^-$$

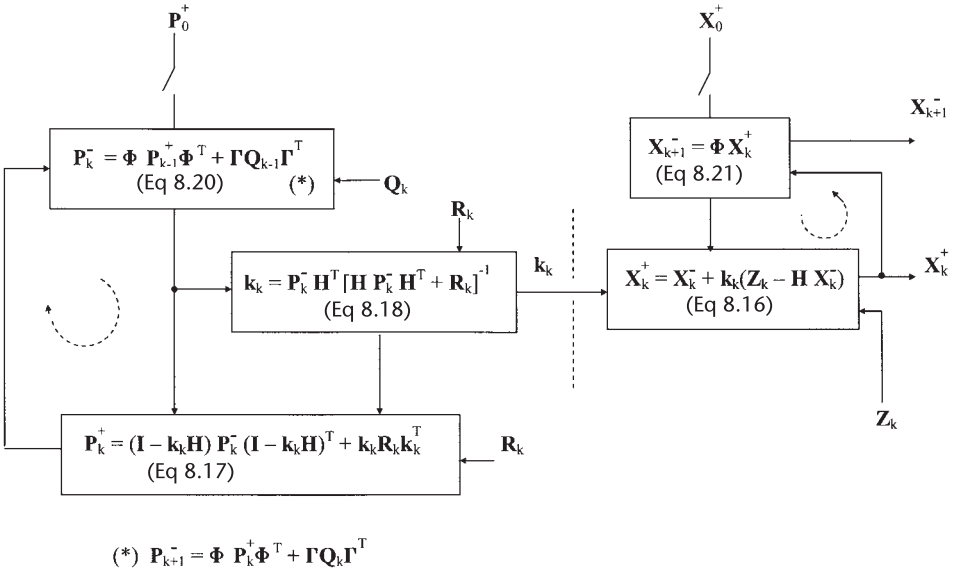


Figure 8.7 Flow diagram of Kalman recursive filter.

Of the three expressions for $\mathbf{P}_k^+$, the last one is the simplest, and it is used more frequently than the others; however, the last expression is computationally inferior as pointed out by Busy and Joseph [4]. An accumulation of round-off error can sometimes destroy the positive definiteness and symmetry of $\mathbf{P}_k^+$. The preferred one is (8.17).

Next we derive an expression for $\mathbf{P}_k^-$, the a priori error covariance matrix so that $\mathbf{P}_k^+$ and $\mathbf{P}_k^-$ can be linked in a recursive loop. We start with the definition of $\mathbf{P}_k^-$ and an error-free state $\mathbf{x}_k^-$.

$$\begin{aligned}\mathbf{P}_k^- &= \mathrm{E}\{\tilde{\mathbf{x}}_k^-(\tilde{\mathbf{x}}_k^-)^T\}\\ \mathbf{x}_k^- &= \boldsymbol{\Phi}_{k-1}\mathbf{x}_{k-1}^+\end{aligned} \tag{8.19}$$

Subtract $\mathbf{x}_k$ from both sides of (8.19) and substitute (8.8) for $\mathbf{x}_k$. We obtain the a priori error $\tilde{\mathbf{x}}_k^-$.

$$\begin{aligned}\mathbf{x}_k^- - \mathbf{x}_k &= \boldsymbol{\Phi}_{k-1}\mathbf{x}_{k-1}^+ - \mathbf{x}_k\\ &= \boldsymbol{\Phi}_{k-1}\mathbf{x}_{k-1}^+ - (\boldsymbol{\Phi}_{k-1}\mathbf{x}_{k-1} + \boldsymbol{\Gamma}\mathbf{w}_{k-1})\\ &= \boldsymbol{\Phi}_{k-1}[\mathbf{x}_{k-1}^+ - \mathbf{x}_{k-1}] - \boldsymbol{\Gamma}\mathbf{w}_{k-1}\end{aligned}$$

and

$$\tilde{\mathbf{x}}_k^- = \boldsymbol{\Phi}_{k-1}\tilde{\mathbf{x}}_{k-1}^+ - \boldsymbol{\Gamma}\mathbf{w}_{k-1}$$

Then, the a priori error covariance matrix is given by

$$\begin{aligned}\mathbf{P}_k^- &= \mathrm{E}\{\tilde{\mathbf{x}}_k^-(\tilde{\mathbf{x}}_k^-)^{\mathrm{T}}\}\\ &= \mathrm{E}\{[\boldsymbol{\Phi}_{k-1}\tilde{\mathbf{x}}_{k-1}^+ - \boldsymbol{\Gamma}\mathbf{w}_{k-1}]\,[\boldsymbol{\Phi}_{k-1}\tilde{\mathbf{x}}_{k-1}^+ - \boldsymbol{\Gamma}\mathbf{w}_{k-1}]^{\mathrm{T}}\}\\ &= \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1}^+\boldsymbol{\Phi}_{k-1}^{\mathrm{T}} + \boldsymbol{\Gamma}\mathbf{Q}_{k-1}\boldsymbol{\Gamma}^{\mathrm{T}}\end{aligned} \tag{8.20}$$

where

$$\mathrm{E}\,\{\mathbf{w}_{k-1}(\mathbf{w}_{k-1})^{\mathrm{T}}\} = \mathbf{Q}_{k-1}$$

We have used the uncorrelated relationship between $\mathbf{w}_{k-1}$ and $\tilde{\mathbf{x}}_{k-1}^-$ in arriving at (8.20):

$$\mathrm{E}\{\mathbf{w}_{k-1}(\tilde{\mathbf{x}}_{k-1}^-)^{\mathrm{T}}\} = 0$$
$$\mathrm{E}\{\tilde{\mathbf{x}}_{k-1}^-(\mathbf{w}_{k-1})^{\mathrm{T}}\} = 0$$

Equation (8.20) establishes a linkage between $\mathbf{P}_k^-$ and $\mathbf{P}_{k-1}^+$. This is the sixth equation of Kalman's seven. The seventh equation is simply

$$\mathbf{x}_{k+1}^- = \boldsymbol{\Phi}\,\mathbf{x}_k^+ \tag{8.21}$$

We have derived all seven equations of the Kalman filter. They are listed below as a summary. The recursive filter algorithm is shown in Figure 8.7 without subscripts for those matrices that are time-invariant.

The Kalman recursive filter consists of two loops as shown; the one on the left computes the error covariance matrix $\mathbf{P}_k^-$ and $\mathbf{P}_k^+$, producing the Kalman gain matrix $\mathbf{k}_k$. The loop on the right computes $\mathbf{x}_k^+$ and the predicted state $\mathbf{x}_{k+1}^-$. The two loops are connected by the Kalman gain $\mathbf{k}_k$.

We are ready to program the recursive filter except for the different coordinate system we have used for the passenger airliner: the state equation is in the CCS and the measurement equation in the LOS system.

$$\begin{cases}\mathbf{x}_{k+1} = \boldsymbol{\Phi}\mathbf{x}_k + \boldsymbol{\Gamma}\mathbf{w}_k & \text{(CCS)}\\ \mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k & \text{(LOS)}\end{cases} \tag{8.22}$$

The coordinate conversion will be discussed shortly.

Summary of Kalman filter equations:

State equation (8.8):	$\mathbf{x}_{k+1} = \boldsymbol{\Phi}\mathbf{x}_k + \boldsymbol{\Gamma}\mathbf{w}_k$
Measurement equation (8.11):	$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$
State update equation (8.16):	$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{k}_k[\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-]$
Predicted error covariance matrix (8.17):	$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{k}_k\mathbf{H}\,]\,\mathbf{P}_k^-\,[\mathbf{I} - \mathbf{k}_k\mathbf{H}\,]^{\mathrm{T}} + \mathbf{k}_k\mathbf{R}_k\mathbf{k}_k^{\mathrm{T}}$
Kalman gain matrix (8.18)	$\mathbf{k}_k = \mathbf{P}_k^-\mathbf{H}^{\mathrm{T}}[\mathbf{H}\mathbf{P}_k^-\,\mathbf{H}^{\mathrm{T}} + \mathbf{R}_k\,]^{-1}$

Estimated error covariance matrix (8.20).
(You may drop k–1 on Φ_{k-1}): $\mathbf{P}_k^- = \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1}^+\boldsymbol{\Phi}_{k-1}^T + \boldsymbol{\Gamma}\mathbf{Q}_{k-1}\boldsymbol{\Gamma}^T$

State predicted equation (8.21): $\mathbf{x}_{k+1}^- = \boldsymbol{\Phi}\mathbf{x}_k^+$

Some alternative forms for $\mathbf{k}_k$ are found in [5, 6].

In order to be consistent in coordinate systems in (8.22), the measurements $\mathbf{z}_k$ should be converted to CCS or the state equation $\mathbf{x}_{k+1}$ should be converted to LOS. Let us look at (8.11) once more.

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$

where

$$\mathbf{z}_k = \begin{bmatrix} r_k \\ \theta_k \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{v}_k = \begin{bmatrix} n_r \\ n_\theta \end{bmatrix}$$

We can certainly change $\mathbf{z}_k$ to CCS; $\mathbf{H}$ would remain unchanged as is shown below, but $\mathbf{v}_k$ is a problem. Let us look at Figure 8.8 in the next page that helps a coordinate transformation we have in mind: from LOS to CCS.

$$\begin{bmatrix} x \\ y \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k + [v_k^{LOS} \rightarrow v_k^{CCS}]$$

From Figure 8.8 we can write the following transformation equations:

$$\begin{cases} \Delta x = \Delta r \cos\theta - r\Delta\theta \sin\theta \\ \Delta y = \Delta r \sin\theta + r\Delta\theta \cos\theta \end{cases}$$

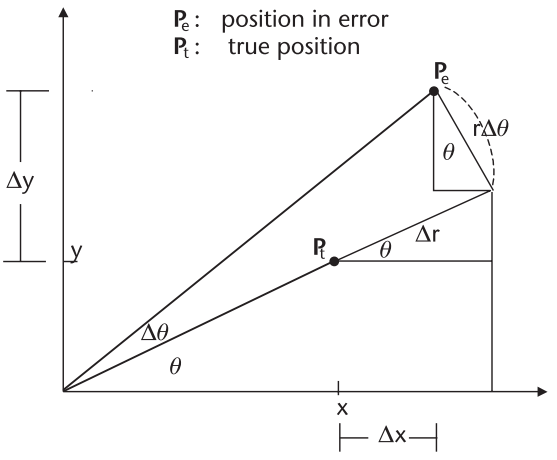


Figure 8.8 Coordinate transformation from (r,θ) to (x,y).

The above equations can be cast into a vector-matrix form,

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \begin{bmatrix}\cos\theta & -r\sin\theta\\ \sin\theta & r\cos\theta\end{bmatrix}\begin{bmatrix}\Delta r\\ \Delta\theta\end{bmatrix}$$

or

$$\begin{bmatrix}\Delta x\\ \Delta y\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}\begin{bmatrix}\Delta r\\ r\Delta\theta\end{bmatrix}$$

The first transform is called a Jacobian transformation, the second a unitary rotation transformation. The determinant of the Jacobian is r, the determinant of the unitary rotation is unity. The transformation of $\mathbf{v}_k^{LOS}$ to $\mathbf{v}_k^{CCS}$, or $\mathbf{R}_k(r,\theta)$ to $\mathbf{R}_k(x,y)$ is obtained by matrix multiplications

$$\mathbf{R}_k(x,y) = \mathbf{J}_k(r,\theta)\,\mathbf{R}_k(r,\theta)\,\mathbf{J}_k^T(r,\theta)$$

Substitution of the Jacobian and its transpose leads to an expression for $\mathbf{R}_k(x,y)$ in terms of $\mathbf{R}_k(r,\theta)$.

$$\begin{aligned}\mathbf{R}_k(x,y) &= \begin{bmatrix}\cos\theta_k & -r_k\sin\theta_k\\ \sin\theta_k & r_k\cos\theta_k\end{bmatrix}\begin{bmatrix}\sigma_r^2 & 0\\ 0 & \sigma_\theta^2\end{bmatrix}\begin{bmatrix}\cos\theta_k & \sin\theta_k\\ -r_k\sin\theta_k & r_k\cos\theta_k\end{bmatrix}\\ &= \begin{bmatrix}\sigma_r^2\cos^2\theta_k + \sigma_\theta^2 r_k^2\sin^2\theta_k & \tfrac{1}{2}\sin(2\theta_k)(\sigma_r^2-\sigma_\theta^2\sigma_r^2)\\ \tfrac{1}{2}\sin(2\theta_k)(\sigma_r^2-\sigma_\theta^2\sigma_r^2) & \sigma_r^2\sin^2\theta_k + \sigma_\theta^2 r_k^2\cos^2\theta_k\end{bmatrix}\end{aligned}$$

where the subscript k in σ_r and σ_θ has been dropped since they are time-invariant. The range and angle are measured (or reported) by the radar for each sample k. In order to incorporate $\mathbf{R}_k(x,y)$ into our computer program, we have to define σ_r^2 and σ_θ^2.

Let us assume that the transmitter pulsewidth is 1 ms and that the –3dB antenna beamwidth is 2 degrees. The target is equally likely located within these limits uniformly. Then the variance of the range error and that of angle are given by

$$\sigma_r^2 = \frac{(1.0e-6)^2}{12} \rightarrow \frac{(150\text{ m})^2}{12} = 1{,}850\text{ m}^2$$
$$\sigma_r = 43.3\text{ m}$$

and

$$\sigma_\theta^2 = \frac{(2^\circ)^2}{12} = \frac{(3.49e{-}2)^2}{12} = 1.015e\text{-}4\text{ radian-squared}$$

$$\sigma_\theta = 1.008e\text{-}2\text{ radian, or }\sigma_\theta = 0.58\text{ degree}$$

Finally we are ready to program the Kalman recursive filter shown in Figure 8.7. Note the injections of the initial values of $\mathbf{P}_0^+$ and $\mathbf{x}_0^+$ through one-pole switches at

the top. Once the initial values are injected the switch will be opened. The program for the passenger airliner will be given in the next section.

8.3 Passenger Airliner

First, we shall generate two flight trajectories: one trajectory for an intended straight line path and other actual trajectory perturbed by small accidental accelerations in both x and y (see Figure 8.3).

FLT_XY.CPP generates the ideal path, TRAJ_XY.CPP generates an actual path. The acceleration perturbation is assumed distributed uniformly with zero mean and variance $\sigma_x^2 = \sigma_y^2 = 8.0$ (m/sec)^2.

The measurement data is generated in MEA_XY.CPP. The measurement errors in range and angle are also assumed distributed uniformly with zero mean and variance $\sigma_r^2 = 1{,}850$ m^2, $\sigma_\theta^2 = 1.015\text{e}{-4}$ rad^2.

Kalman filter processing is programmed in KAL_XY.CPP with a header file MATRIX.H. All the relevant data computed is recorded. The data is listed as follows:

1. Pest.DAT: Estimated error covariance matrix;
2. Ppre.DAT: Predicted error covariance matrix;
3. K_gain.DAT: Kalman gain matrix;
4. Xest.DAT: Estimated state (estimated position);
5. Xpre.DAT: Predicted state (predicted position).

We have generated two additional data files from (4) and (5) above for a postflight analysis. They are listed as follows:

6. EST_RAZ.CPP: Estimated range and azimuth angle;
7. PRE_RAZ.CPP: Predicted range and azimuth angle.

The last two data files plus FLT_XY.DAT, TRAJ_XY.CPP and MEA_RAZ.DAT are used for a postflight analysis and graphic displays, Figures 8.9 and 8.10.

Postflight Analysis and Discussions

1. We notice that there is a large deviation between estimated and predicted ranges and azimuth angles during the earlier period. The large down-bulge is caused by an inappropriate initialization of $\mathbf{P}_0^+$ and $\mathbf{x}_0^+$. For the initialization of $\mathbf{x}_0^+$ we had injected the value shown below since we didn't know the velocity components.

$$\mathbf{x}_0^+ = \begin{bmatrix} x_0 \\ \dot{x}_0 \\ y_0 \\ \dot{y}_0 \end{bmatrix} = \begin{bmatrix} 0.00 \\ 0.00 \\ 3300.00 \\ 0.00 \end{bmatrix} \begin{matrix} \\ \dot{x}_0 = 0.00 \\ \\ \dot{y}_0 = 0.00 \end{matrix}$$

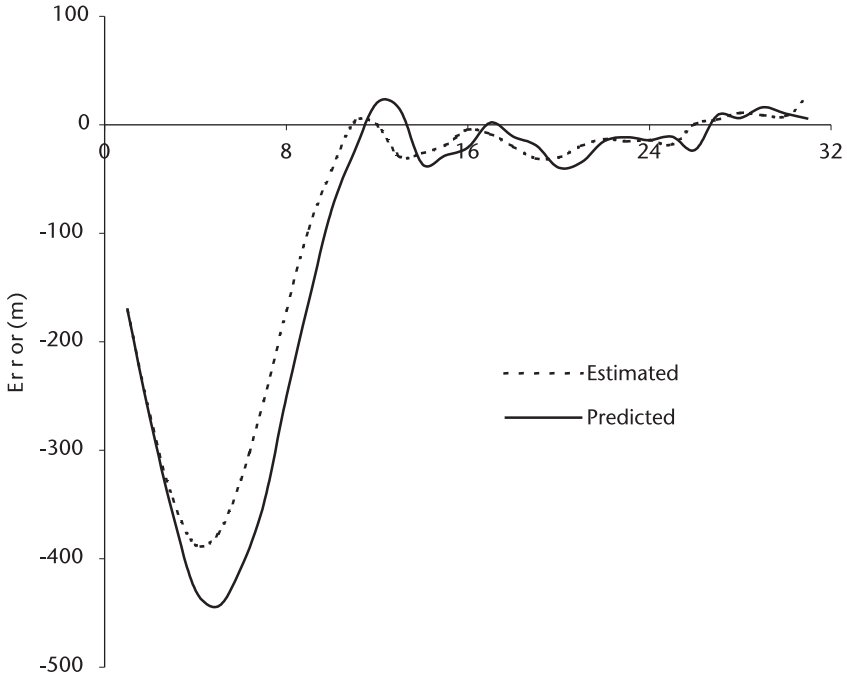


Figure 8.9 Errors in estimated and predicted range.

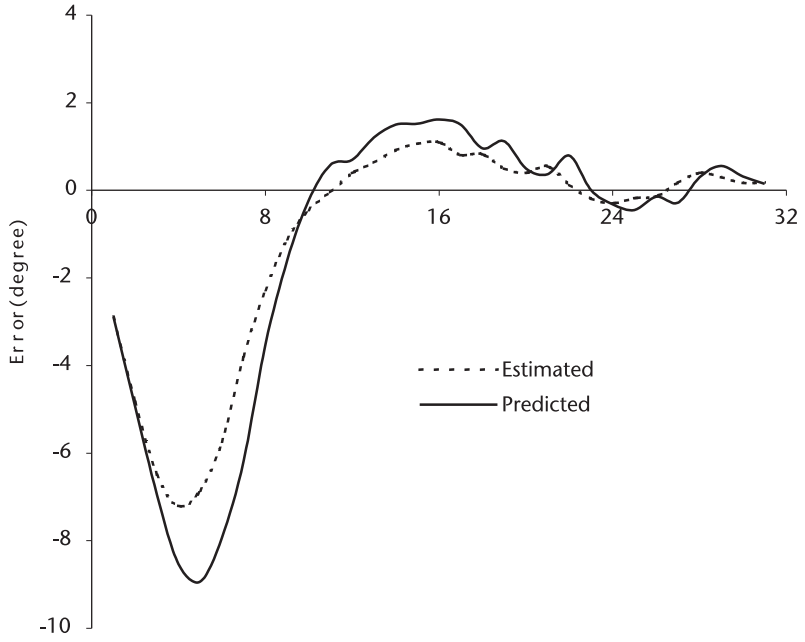


Figure 8.10 Errors in estimated and predicted azimuth.

We will modify the initialization by two-points measurement and inject $\mathbf{x}_1^+$ instead of $\mathbf{x}_0^+$ in the next program.

$$\mathbf{x}_1^+ = \begin{bmatrix} x_1 \\ \dot{x}_1 \\ y_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} x_1 \\ (x_1 - x_0/T) \\ y_1 \\ (y_1 - y_0)/T \end{bmatrix} \begin{matrix} \\ \dot{x}_1 \neq 0.00 \\ \\ \dot{y}_1 \neq 0.00 \end{matrix}$$

For the estimated error covariance matrix $\mathbf{P}_0^+$, a 4-by-4 matrix, all the elements are initialized a unity without any physical justification. After computation and an examination of data, we are pleasantly surprised the system had ever stabilized at all with such arbitrary initialization. In the next program we shall initialize using $\mathbf{P}_1^+$ instead of $\mathbf{P}_0^+$ with some physical meaning attached. We shall adopt modifications in both $\mathbf{x}_1^+$ and $\mathbf{P}_1^+$.

2. Some readers may feel the errors in range are too large. The remedy may be to shorten the transmitter pulsewidth. The consequence would be less energy radiated for target tracking. A pulse compression scheme may be entertained. See Chapter 4.
   If you feel that the errors in angle are too large, the size of the antenna should be increased or the carrier frequency should be increased. The antenna must be redesigned in either case. The consequence of the latter would be less energy return from target. See Chapter 7.
3. Some program algorithm may exhibit a divergent behavior—that is, the error in the estimate $\mathbf{x}_k^+$ or $\mathbf{x}_k^-$ fails to converge within one sigma boundary. A quick check is to see that semidefiniteness and symmetry of the covariance matrix are maintained. We should include test procedures to insure the unique properties of the error covariance matrices during the initial development phase.
   The word length of the computer and the round-off may cause a problem. Inversion of a matrix with large dimensions is often a troublesome source of broken symmetry and the birth of negative elements in $\mathbf{P}_k^+$ or $\mathbf{P}_k^-$. We shall study a better matrix inversion algorithm using the factorization method, or inversion-less algorithm in later sections.
4. In spite of all the precautions, some algorithms may wander around without any semblance of convergence. We should examine the state and measurement equations It is not uncommon that the dimension of Φ matrix is larger than ten-by-ten by including some functionality in a zealous pursuit of perfection that upsets the basic structure of the state equation. Reduce the dimension of the transition matrix Φ to be no larger than essential.
   Modeling a system requires a blend of science and art through experience. Let us get some experience through more examples.
5. In this section we have programmed the following:

   - FLT_XY.CPP: Generates a straight line flight path;
   - TRAJ_XY.CPP: Gaussian noise with zero mean and varianceof plus/minus 0.5 g is added to the straight-line trajectory to simulate the actual motion of the airliner;

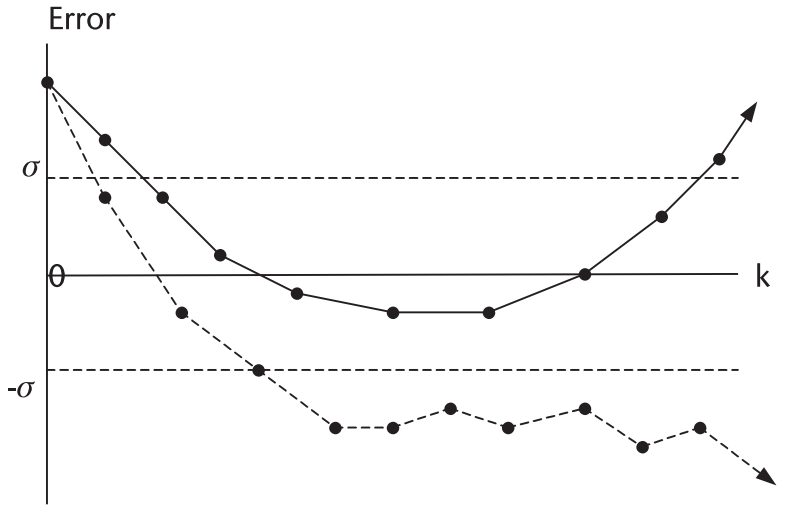


Figure 8.11 Examples of divergence and oscillatory behavior outside the sigma boundary.

- MEA_XY.CPP: The passenger airliner's position is tracked by a radar with 1.0-microsecond pulsewidth and antenna beamwidth of 2 degrees.
- KAL_XY.CPP: The Kalman filter algorithm is executed.
- MATRIX.H: This program generated seven data files:

  1. Pest.DAT: Estimated error covariance matrix;
  2. Ppre.DAT: Predicted error covariance matrix;
  3. K_gain.DAT: Kalman gain matrix;
  4. Xest.DAT: Estimated position of airliner;
  5. Xpre.DAT: Predicted position of airliner;
  6. Est_RAZ.DAT: Estimated position of airliner in range and azimuth;
  7. Pre_RAZ.DAT: Predicted position of airliner in range and azimuth;

- ERR_RAZ.DAT: Computes errors between the measured position and estimated and predicted position in range and azimuth.

8.4 Air Traffic Control Radar

In this section we study the tracking function of an air traffic control (ATC) radar. There are two radars in the ATC system: one for a shorter range and the other for a longer range. The relevant parameters of two radars are listed below. (Later models have improved some parameters).

ATC radars detect airborne targets in range and bearing angle. No elevation angle (altitude) of the target is measured. The altitude of the target aircraft is re-

Table 8.1

	Airport Surveillance Radar (ASR)	*Air Route Surveillance Radar (ARSR)*
Transmitter power	400 kW	4 MW
Maximum range	110 km (60 nm)	370 km (200 nm)
Antenna gain	34 dB	34 dB
Frequency	2.8 GHz	1.3 GHz
Pulse repetition freq	1200 pulse/sec	360 pulse/sec
Antenna scan rate	15 rev/min (90 deg/sec)	6 rev/min (36 deg/sec)
Data update	4 sec	10 sec
Antenna beamwidth	1.5 degrees	1.35 degrees
(-3 dB azimuth) Pulsewidth	0.8 ~ 0.9 μsec (0.83 μsec nominal)	2.0 μsec
Number of received pulses integrable	20	13

ported to the ground control center by a transponder onboard the aircraft when interrogated by a co-located beacon radar on the ground. A counterpart in the military air control is called the identification friend or foe (IFF).

In this second example of Kalman filter processing, the state equation will be written in the LOS coordinate system. We write the state equation (aircraft's position and motion) as follows:

$$\begin{cases} r_{k+1} = r_k + \dot{r}_k T \\ \dot{r}_{k+1} = \dot{r}_k + (\text{rng acc})T \\ \theta_{k+1} = \theta_k + \dot{\theta}_k T \\ \dot{\theta}_{k+1} = \dot{\theta}_k + (\text{az acc})T \end{cases} \tag{8.23}$$

Equation (8.23) is cast into a vector-matrix form.

$$\begin{bmatrix} r_{k+1} \\ \dot{r}_{k+1} \\ \theta_{k+1} \\ \dot{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_k \\ \dot{r}_k \\ \theta_k \\ \dot{\theta}_k \end{bmatrix} + \begin{bmatrix} 0 \\ u_1 \\ 0 \\ u_2 \end{bmatrix} \tag{8.24}$$

A more compact form of (8.24) is

$$\mathbf{x}_{k+1} = \Phi \mathbf{x}_k + \mathbf{w}_k$$

The covariance of $\mathbf{w}_k$, denoted by $\mathbf{Q}_k$, is given by

$$\begin{aligned} \mathbf{Q}_k &= E\{\mathbf{w}_k \mathbf{w}_k^T\} \\ &= E\left\{ \begin{bmatrix} 0 \\ u_1 \\ 0 \\ u_2 \end{bmatrix} [0 \; u_1 \; 0 \; u_2] \right\} = E\left\{ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & u_1^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2^2 \end{bmatrix} \right\} \end{aligned}$$

We must define u_1 and u_2. They are the product of range or angle acceleration and the sampling time T. As we have formulated the state equation these accelerations are considered to be small uncontrolled disturbances. (In the previous example in KAL_XY.CPP, we have taken the accelerations in x and y as definite components of Newtonian motion.)

Let us define the range acceleration more realistically. Let us assume that the range acceleration a_r consists of two parts as shown: a discreet and continuous probability distribution [7]. Further we assume that the aircraft experiences a disturbance of ±0.5g (±4.9 m/sec^2) 10% of time, zero acceleration 30% of time, and that the rest of time the acceleration disturbance is uniformly distributed between the two extremes. (See Figure 8.12.)

The variances of discrete and continuous probability density functions are respectively defined as follows:

$$\sigma_d^2 \equiv \sum_n (x_n - \eta)^2 \; P\{x = x_n\} \qquad \text{(discrete)}$$

$$\sigma_c^2 \equiv E\{(x - \eta)^2\} = \int (x - \eta)^2 \; f(x) \; dx \qquad \text{(continuous)}$$

The variance of the discrete distribution σ_d^2 is given by

$$\sigma_d^2 = \left(\frac{-g^2}{2}\right) P_1 + (0)P_2 + \left(\frac{g^2}{2}\right) P_1$$

and the variance of the continuous distribution σ_c^2 is given by

$$\sigma_c^2 = \frac{[1 - (P_2 + P_1)]}{g} \int_{-g/2}^{g/2} x^2 \; dx = \frac{g^2}{12}(1 - P_2 - 2P_1)$$

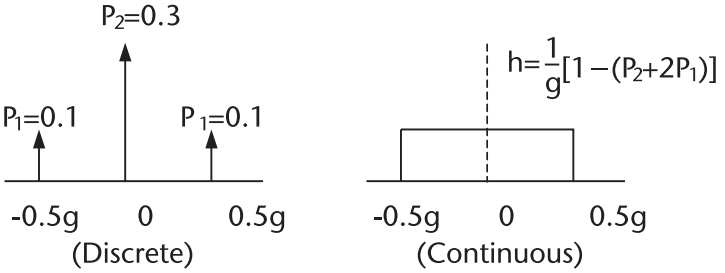


Figure 8.12 Probability density function of range acceleration, discrete and continuous.

The total variance is sum of the two

$$\sigma_t^2 = \frac{g^2}{12}(1 + 4P_1 - P_2) = 8.804 \ (\text{m/sec}^2)^2$$

The covariance matrix $\mathbf{Q}_k$ is given by

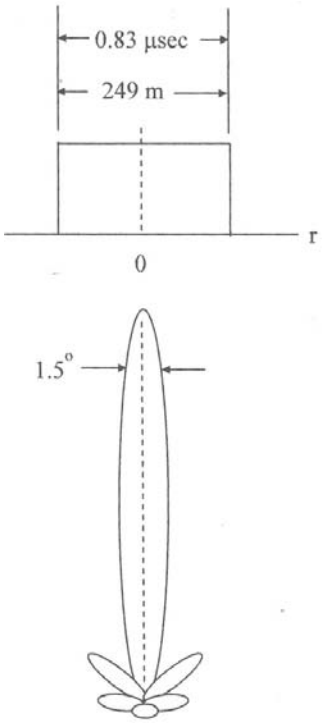
$$\mathbf{Q}_k = E\left\{ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & u_1^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_2^2 \end{bmatrix} \right\} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma^2 T^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{\sigma^2 T^2}{r_k^2} \end{bmatrix}$$

Note that element Q(3,3) is range measurement–dependent. The measurement in the LOS coordinate system is identical to that of KAL_XY.CPP.

$$\begin{bmatrix} r_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_k \\ \dot{r}_k \\ \theta_k \\ \dot{\theta}_k \end{bmatrix} + \begin{bmatrix} n_r \\ n_\theta \end{bmatrix}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$

where n_r and n_θ represent the measurement noise errors in range and angle, and they are assumed uniformly distributed with zero mean. We next compute the numerical values of variances of range and angle measurement error.



[ASR]

$$\sigma_r^2 = \frac{(249/2)^2}{12} = 1291.69 \ \text{m}^2$$

$$\sigma_r = 35.94 \ \text{m}$$

$$\sigma_\theta^2 = \frac{0.0262^2}{12} = 5.712e-5 \ \text{rad}^2$$

$$\sigma_\theta = 7.56e-3 \ \text{rad} = 0.43 \ \text{deg}$$

The measurement error covariance matrix $\mathbf{R}_k$ is given by

$$\mathbf{R}_k = \mathrm{E}\left\{\begin{bmatrix} n_r \\ n_\theta \end{bmatrix}\begin{bmatrix} n_r & n_\theta \end{bmatrix}\right\} = \mathrm{E}\left\{\begin{bmatrix} n_r^2 & n_r n_\theta \\ n_\theta n_r & n_\theta^2 \end{bmatrix}\right\}$$
$$= \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix} = \begin{bmatrix} 1291.69 & 0.00 \\ 0.00 & 5.712\mathrm{e}-5 \end{bmatrix}$$

The state and measurement equations and their error covariance matrices are summarized:

$$\mathbf{x}_{k+1} = \Phi\mathbf{x}_k + \Gamma\mathbf{w}_k \qquad (\text{or } \mathbf{x}_k = \Phi\mathbf{x}_{k-1} + \Gamma\mathbf{w}_{k-1})$$
$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$
$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{k}_k[\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-]$$
$$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{k}_k\mathbf{H}]\ \mathbf{P}_k^-[\mathbf{I} - \mathbf{k}_k\mathbf{H}]^{\mathrm{T}} + \mathbf{k}_k\mathbf{R}_k\mathbf{k}_k^{\mathrm{T}}$$
$$\mathbf{k}_k = \mathbf{P}_k^-\mathbf{H}^{\mathrm{T}}[\mathbf{H}\mathbf{P}_k^-\ \mathbf{H}^{\mathrm{T}} + \mathbf{R}_k]^{-1}$$
$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+\mathbf{\Phi}_{k-1}^{\mathrm{T}} + \mathbf{\Gamma}\mathbf{Q}_{k-1}\mathbf{\Gamma}^{\mathrm{T}} \quad (\text{you may drop } k-1 \text{ on } \mathbf{\Phi}_{k-1})$$

$$\mathbf{x}_{k+1}^- = \mathbf{\Phi}\mathbf{x}_k^+$$

$$\mathbf{Q}_k = \mathrm{E}\{\mathbf{w}_k\mathbf{w}_k^{\mathrm{T}}\} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \sigma^2\mathrm{T}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sigma^2\mathrm{T}^2}{r_k^2} \end{bmatrix}$$

$$\mathbf{R}_k = \mathrm{E}\{\mathbf{v}_k\ \mathbf{v}_k^{\mathrm{T}}\} = \begin{bmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$$

$$\mathbf{\Phi} = \begin{bmatrix} 1 & \mathrm{T} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \mathrm{T} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Compare the above equations and matrices with the corresponding equations and matrices of the previous example. We note that the different expressions for $\mathbf{Q}_k$ and $\mathbf{R}_k$. The ATC example in LOS coordinates system is much simpler: there is less computation load. The measurement covariance matrix $\mathbf{R}_k$ is time-invariant; no subscript k is needed.

The predicted and estimated error covariance matrices, $\mathbf{P}_k^-$ and $\mathbf{P}_k^+$, Kalman gain matrix $\mathbf{k}_k$, and the predicted and estimated state vector $\mathbf{x}_k^-$ and $\mathbf{x}_k^+$ are listed below, and a computation flow diagram is shown in Figure 8.16.

$$
\begin{aligned}
\mathbf{P}_k^- &= \mathbf{\Phi}\mathbf{P}_{k-1}^+\mathbf{\Phi}^T + \mathbf{Q}_{k-1} \\
\mathbf{k}_k &= \mathbf{P}_k^-\mathbf{H}^T[\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}_k]^{-1} \\
\mathbf{P}_k^+ &= [\mathbf{I} - \mathbf{k}_k\mathbf{H}]\ \mathbf{P}_k^-[\mathbf{I} - \mathbf{k}_k\mathbf{H}]^T + \mathbf{k}_k\mathbf{R}\mathbf{k}_k^T \\
\mathbf{x}_k^+ &= \mathbf{x}_k^- + \mathbf{k}_k[\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-] \\
\mathbf{x}_{k+1}^- &= \mathbf{\Phi}\mathbf{x}_k^+
\end{aligned}
$$

We plan to inject two initializations that are different from the previous example; instead of $\mathbf{x}_0^+$ we plan to inject $\mathbf{x}_1^+$.

$$
\mathbf{x}_1^+ = \begin{bmatrix} r_1 \\ \dot{r}_1 \\ \cdot^1 \\ {}^1 \end{bmatrix} = \begin{bmatrix} r_1 \\ (r_1 - r_0)/T \\ {}^1 \\ (\ {}^1 - \ {}^2)/T \end{bmatrix}
$$

Instead of $\mathbf{P}_0^+$ we plan to inject $\mathbf{P}_1^+$. Let us examine what $\mathbf{P}_1^+$ should be.

$$
\mathbf{P}_1^+ = E\{(\mathbf{x}_1 - \hat{\mathbf{x}}_1)(\mathbf{x}_1 - \hat{\mathbf{x}}_1)^T\} = E\left\{ \begin{bmatrix} \Delta r \\ \Delta \dot{r} \\ \Delta\theta \\ \Delta\dot{\theta} \end{bmatrix} [\Delta r \ \ \Delta \dot{r} \ \ \Delta\theta \ \ \Delta\dot{\theta}] \right\}
$$

$$
= \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix}_{k=1}
$$

$$
\begin{cases}
p_{00} = E\{n_{r1} \cdot n_{r1}^T\} = \sigma_r^2 \\
p_{01} = E\{n_{r1}[(n_{r1} - n_{r0})/T + u_1]^T\} \\
\qquad = E\{n_{r1}^2/T + n_{r1} \cdot u_1\} = \sigma_r^2 T \\
p_{02} = E\{n_{r1} \cdot n_{\theta 1}^T\} = 0 \\
p_{03} = E\{n_{r1}[(n_{\theta 1} - n_{\theta 0})/T + u_2]^T\} = 0
\end{cases}
$$

$$
\begin{cases}
p_{10} = p_{01} \\
p_{11} = E\{[(n_{r1} - n_{r0})/T + u_1][(n_{r1} - n_{r0})/T + u_1]^T\} \\
\qquad = E\{[(n_{r1} - n_{r0})/T]^2 + u_1^2\} = 2\sigma_r^2/T + \sigma_r^2 T^2 \\
p_{12} = E\{[(n_{r1} - n_{r0})/T + u_1][n_{\theta 1} + u_2]^T\} = 0 \\
p_{13} = E\{[(n_{r1} - n_{r0})/T + u_1][(n_{r1} - n_{r0})/T + u_1]^T\} = 0
\end{cases}
$$

$$\begin{cases} p_{20} = p_{02} \\ p_{21} = p_{12} \\ p_{22} = E\{n_{\theta 1} \cdot n_{\theta 1}^T\} = \sigma_\theta^2 \\ p_{23} = E\{n_{\theta 1}[(n_{\theta 1} - n_{\theta 0})/T + u_2]^T\} \\ \qquad E\{n_{\theta 1}^2/T - (n_{\theta 1} \cdot n_{\theta 0})/T\} = \sigma_\theta^2/T \end{cases}$$

$$\begin{cases} p_{30} = p_{03} = 0 \\ p_{31} = p_{13} = 0 \\ p_{32} = p_{23} \\ p_{33} = E\{[(n_{\theta 1} - n_{\theta 0})/T + u_2][(n_{\theta 1} - n_{\theta 0})/T + u_2]^T\} \\ \qquad = E\{(n_{\theta 1} - n_{\theta 0})^2/T^2 + u_2^2\} = 2\sigma_\theta^2/T^2 + \theta^2 T^2/r_1^2 \end{cases}$$

Therefore, the matrix form of $\mathbf{P}_1^+$ is

$$\mathbf{P}_1^+ = \begin{bmatrix} \sigma_r^2 & \sigma_r^2/T & 0 & 0 \\ \sigma_r^2/T & 2\sigma_r^2/T + \sigma_r^2 T^2 & 0 & 0 \\ 0 & 0 & \sigma_\theta^2 & \sigma_\theta^2/T \\ 0 & 0 & \sigma_\theta^2/T & 2\sigma_\theta^2/T^2 + \theta^2 T^2/r_1^2 \end{bmatrix}$$

The numerical value for $\mathbf{P}_1^+$ will be computed immediately after $\mathbf{z}_k$(k=1) is available. Next, we generate the trajectory of an airliner that approaches an airport for landing. (See Figure 8.13.)

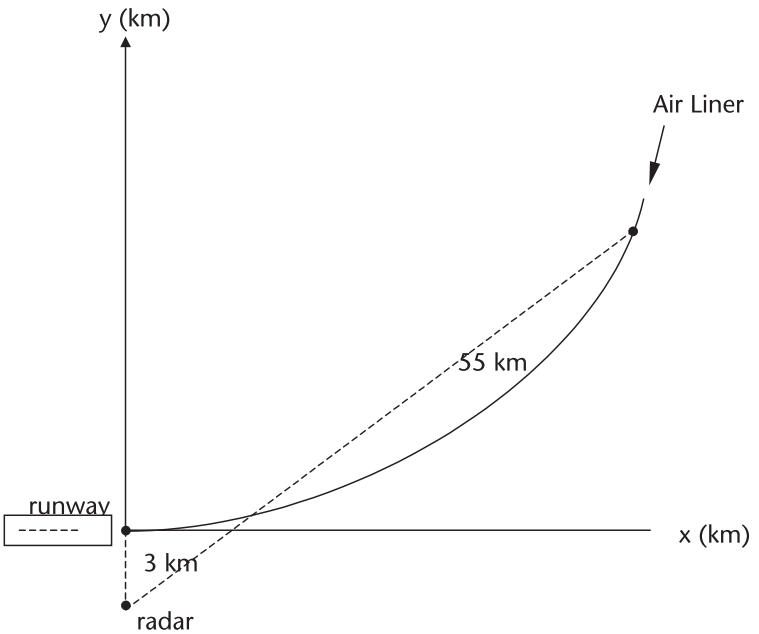


Figure 8.13 Flight trajectory, landing approach.

The tracking commences when the airliner is 55 km away at the cruising speed of approximately 270 knots. The pilot reduces the speed gradually to about 150 knots for the final approach. The antenna scan rate of the ASR is 90 degress/s, and the data rate is therefore 4 seconds. The trajectory is programmed in ATC_RAZ. CPP. Kalman filter processing is programmed in KAL_ATC.CPP with a header file MATRIX.H. The program follows the two-loops recursive diagram shown in Figure 8.16.

Since the complete data file is very voluminous, an abbreviated data file is created to record the results.

- ATC_(P).DAT: Predicted and estimated error covariance matrices have 16 elements, four-by-four, for every sampling, and there are 128 samplings. We record only two principal data, p(0,0) and p(2,2).
- ATC_(k).CPP: Kalman gain matrix has eight elements, four-by-two, for each sampling. We record only two, k(0,0) and k(2,1).
- ATC_(X).CPP: Predicted and estimated position vectors have four elements, four-by-one, for each sampling. We record only two principal data, x(0,0) and x(2,0).

The differences (errors) between the trajectory and the estimated or predicted state are computed in ATC_ERR.CPP. The errors are recorded ATC_ERR. DAT, and the results are shown in Figures 8.14 and 8.15.

A considerable dispersion of errors is observed. This is a price we pay for a simpler state equation in LOS coordinates. We have gained an invaluable experience

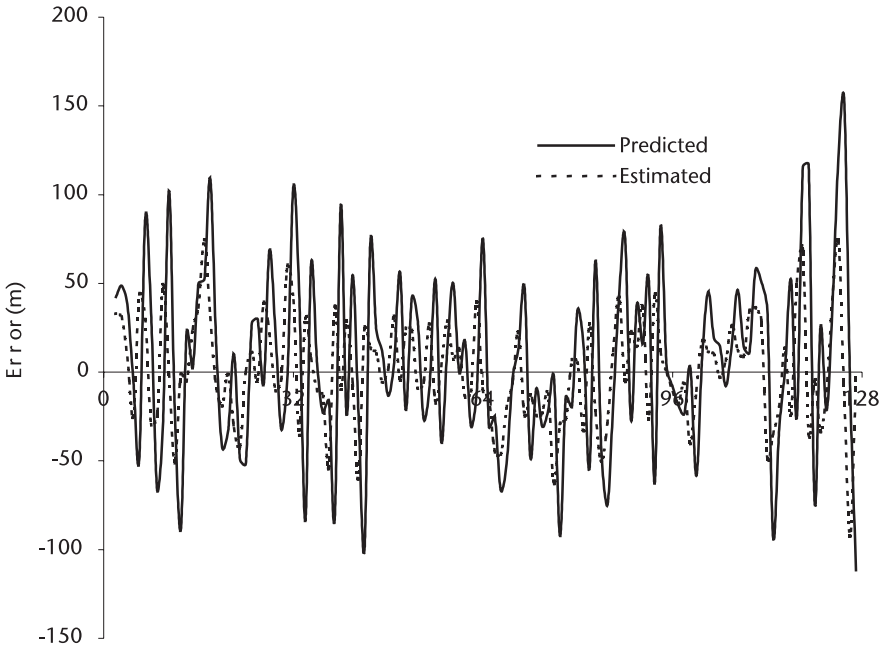


Figure 8.14 Predicted and estimated range error.

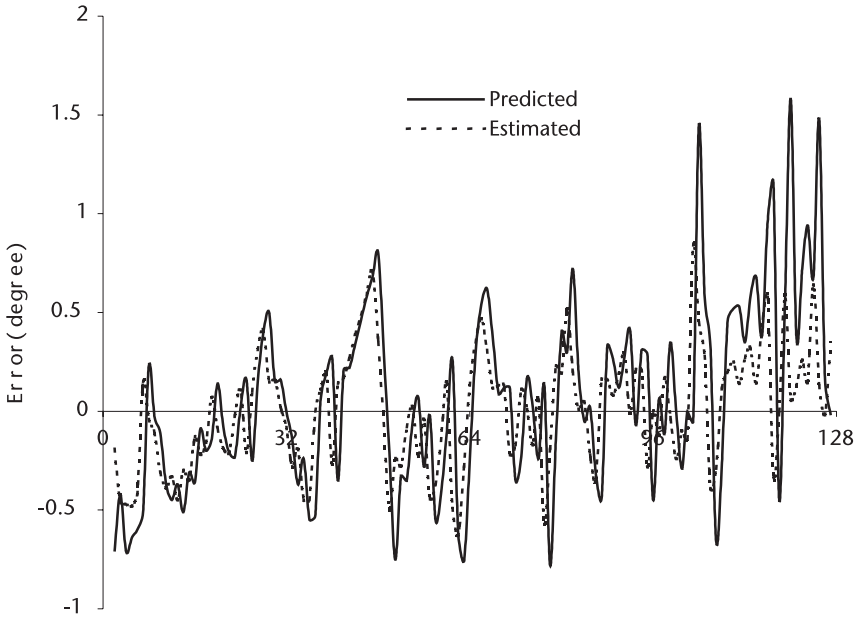


Figure 8.15 Predicted and estimated angle error.

in appreciation of the LOS versus CCS state equation and the concomitant simpler structures of $\mathbf{R}_k$ and $\mathbf{Q}_k$.

An ATC radar is not a precision-approach-control radar. In spite of its large error dispersion, the ATC radar serves the purpose of separating aircrafts in the air-space around the airport.

When we examine the following two situations we realize the root cause of errors in the present state equation.

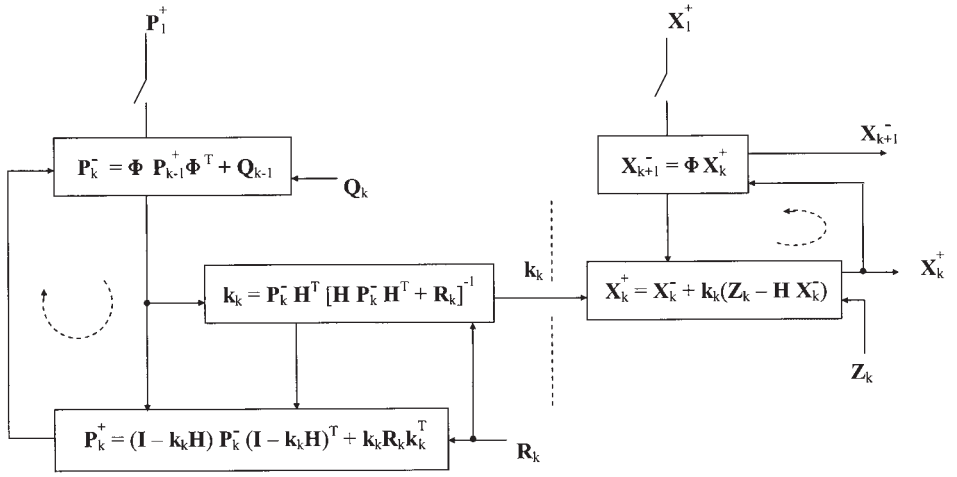
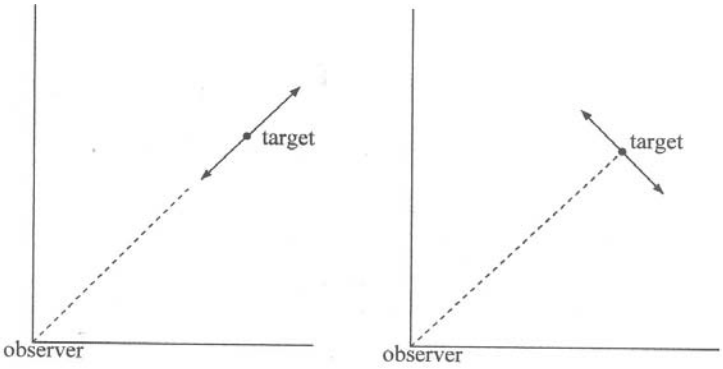


Figure 8.16 Flow diagram of Kalman recursive filter.



The first situation is where a target is radially approaching or receding from an observer (radar). The motion of the target is exactly what the observer measures. The second situation is where a target is in tangential motion; an observer declares that the target has zero range change even though the target may have a large velocity and/or acceleration.

Figures 8.14 and 8.15 show a relatively large error toward the end of tracking the airliner. The airliner is moving in tangentially with respect to the tracking radar. A relocation of radar site in Figure 8.13 may be recommended.

We should not, however, hesitate to adopt filter processing in LOS coordinates when processing speed is more important and accuracy is not. Nothing should discourage the use of the LOS system.

The complexity and computation load is largely dependent upon the dimensions of transition matrix $\mathbf{\Phi}$. In the present example $\mathbf{\Phi}$ is four-by-four. Later we shall design a state equation in LOS coordinates with nine-by-nine and show the results for comparison.

8.5 Air Defense Radar, Cartesian Coordinate System

In this section we design a Kalman filter for a short-range air defense radar. The radar tracks a hostile fighter-bomber on a ground attack mission. The trajectory is a typical "turn-dive-and-turn-climb" maneuver (Figure 8.17). The tracking radar is located at the origin of the Cartesian coordinate system.

We write the state equation (aircraft position and motion) in CCS coordinates (x,y,z) as follows.

$$\begin{cases} x_{k+1} = x_k + \dot{x}_k T + 1/2\ddot{x}_k T^2 \\ \dot{x}_{k+1} = \dot{x}_k + \ddot{x}_k T \\ \ddot{x}_{k+1} = \ddot{x}_k + n_{ax} \end{cases}$$

$$\begin{cases} y_{k+1} = y_k + \dot{y}_k T + 1/2\ddot{y}_k T^2 \\ \dot{y}_{k+1} = \dot{y}_k + \ddot{y}_k T \\ \ddot{y}_{k+1} = \ddot{y}_k + n_{ay} \end{cases}$$

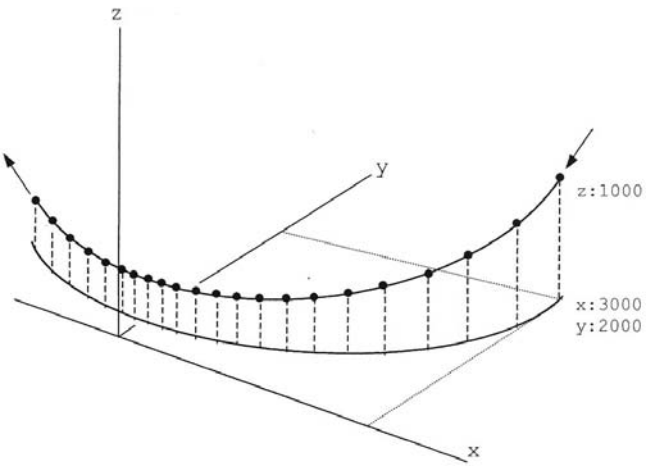


Figure 8.17 Target trajectory, turn-dive-and-turn-climb.

$$\begin{cases} z_{k+1} = z_k + \dot{z}_k T + 1/2\ddot{z}_k T^2 \\ \dot{z}_{k+1} = \dot{z}_k + \ddot{z}_k T \\ \ddot{z}_{k+1} = \ddot{z}_k + n_{az} \end{cases}$$

The above three simultaneous linear equations can be written in a vector-matrix form where n_{ax}, n_{ay}, and n_{az} are the random acceleration noise in x, y, and z axes.

$$\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \\ z \\ \dot{z} \\ \ddot{z} \end{bmatrix}_{k+1} = \left[\begin{array}{ccc:ccc:ccc} 1 & T & T^2/2 & & & & & & \\ 0 & 1 & T & & \mathbf{0} & & & \mathbf{0} & \\ 0 & 0 & 1 & & & & & & \\ \hdashline & & & 1 & T & T^2/2 & & & \\ & \mathbf{0} & & 0 & 1 & T & & \mathbf{0} & \\ & & & 0 & 0 & 1 & & & \\ \hdashline & & & & & & 1 & T & T^2/2 \\ & \mathbf{0} & & & \mathbf{0} & & 0 & 1 & T \\ & & & & & & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ y \\ \dot{y} \\ \ddot{y} \\ z \\ \dot{z} \\ \ddot{z} \end{bmatrix}_k + \begin{bmatrix} 0 \\ 0 \\ n_{ax} \\ 0 \\ 0 \\ n_{ay} \\ 0 \\ 0 \\ n_{az} \end{bmatrix}$$

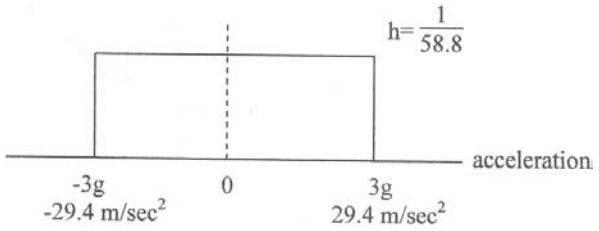
A more compact form is

$$\mathbf{x}_{k+1} = \Phi \mathbf{x}_k + \mathbf{w}_k \tag{8.25}$$

The random acceleration noise covariance matrix $\mathbf{Q}_k$ is given by

$$\mathbf{Q}_k = E\{\mathbf{w}_k \mathbf{w}_k^T\} = \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & \sigma_x^2 & & & \\ & & & 0 & & \\ & & & & 0 & \\ & & & & & \sigma_y^2 \\ & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ 0 \\ 0 \\ \sigma_z^2 \end{matrix}$$

The random acceleration noise is assumed equal for three axes: $n_{ax} = n_{ay} = n_{az}$, uncorrelated and uniformly distributed over ± 3.0 g.



The variance $\sigma^2 = \frac{(6g)^2}{12} = 288.12\ (m/sec^2)^2$. The measurement equation in LOS coordinates system is

$$\begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \\ \ddot{r} \\ \theta \\ \dot{\theta} \\ \ddot{\theta} \\ \varphi \\ \dot{\varphi} \\ \ddot{\varphi} \end{bmatrix} + \begin{bmatrix} \Delta r \\ \Delta\theta \\ \Delta\varphi \end{bmatrix}$$

The measurement equation in a compact matrix form is

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{8.26}$$

The measurement error covariance matrix $\mathbf{R}_k$ in LOS is given by

$$\mathbf{R}_k = \mathrm{E}\{\mathbf{v}_k \mathbf{v}_k^{\mathrm{T}}\} = \mathrm{E}\left\{\begin{bmatrix}\Delta r \\ \Delta\theta \\ \Delta\varphi\end{bmatrix}[\Delta r\ \Delta\theta\ \Delta\varphi]\right\} = \mathrm{E}\left\{\begin{bmatrix}(\Delta r)^2 & \Delta r\Delta\theta & \Delta r\Delta\varphi \\ \Delta\theta\Delta r & (\Delta\theta)^2 & \Delta\theta\Delta\varphi \\ \Delta\varphi\Delta r & \Delta\varphi\Delta\theta & (\Delta\varphi)^2\end{bmatrix}\right\}$$
$$= \begin{bmatrix}\sigma_r^2 & & \\ & \sigma_\theta^2 & \\ & & \sigma_\varphi^2\end{bmatrix} \tag{8.27}$$

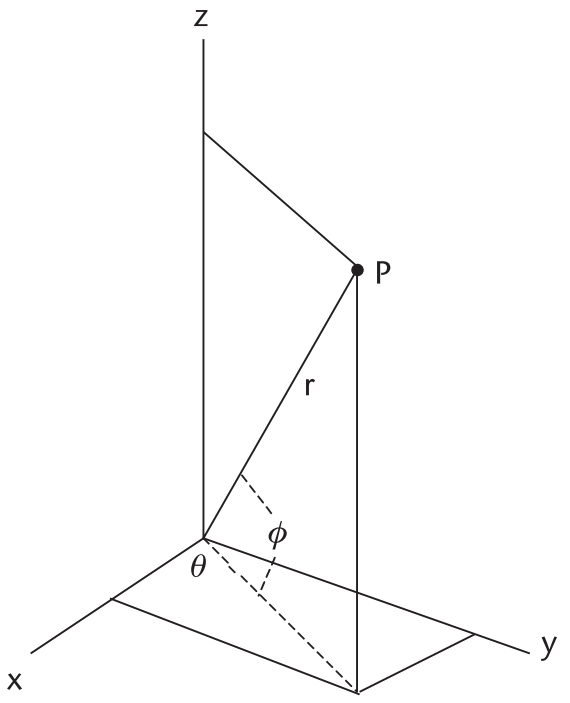
On the other hand, the error covariance matrix $\mathbf{R}_k$ in CCS is given by

$$\mathbf{R}_k = \mathrm{E}\{\mathbf{v}_k \mathbf{v}_k^{\mathrm{T}}\} = \mathrm{E}\left\{\begin{bmatrix}\Delta x \\ \Delta y \\ \Delta z\end{bmatrix}[\Delta x\ \Delta y\ \Delta z]\right\} = \mathrm{E}\left\{\begin{bmatrix}(\Delta x)^2 & \Delta x\Delta y & \Delta x\Delta z \\ \Delta y\Delta x & (\Delta y)^2 & \Delta y\Delta z \\ \Delta z\Delta x & \Delta z\Delta y & (\Delta y)^2\end{bmatrix}\right\}$$
$$= \begin{bmatrix}\sigma_x^2 & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_y\sigma_x & \sigma_y^2 & \sigma_y\sigma_z \\ \sigma_z\sigma_x & \sigma_z\sigma_y & \sigma_z^2\end{bmatrix} \tag{8.28}$$

Since the state equation is written in CCS coordinates and the measurement equation in LOS coordinates, we need a conversion formula between the two coordinates.

$$\mathbf{R}_k(\mathrm{LOS}) \leftrightarrow \mathbf{R}_k(\mathrm{CCS})$$

Figure 8.18 helps us to formulate the conversion between the Cartesian coordinates and the spherical coordinates.



$$r = (x^2 + y^2 + z^2)^{1/2}$$
$$\theta = \tan^{-1}\left(\frac{y}{x}\right) = \sin^{-1}\left[\frac{y}{(x^2 + y^2)^{1/2}}\right]$$
$$\varphi = \sin^{-1}\left(\frac{z}{r}\right)$$

Figure 8.18 Coordinate conversion, $\mathbf{R}_k(\mathrm{LOS}) \leftrightarrow \mathbf{R}_k(\mathrm{CCS})$.

The Jacobian transform matrices in (x,y,z) and (r, θ, ϕ) are defined by

$$\mathbf{J}_{xyz} \equiv \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \\ \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial z} \\ \frac{\partial \varphi}{\partial x} & \frac{\partial \varphi}{\partial y} & \frac{\partial \varphi}{\partial z} \end{bmatrix} = \begin{bmatrix} \frac{x}{r} & \frac{y}{r} & \frac{z}{r} \\ \frac{-y}{x^2+y^2} & \frac{x}{x^2+y^2} & 0 \\ \frac{-xz}{r^2(x^2+y^2)^{1/2}} & \frac{-yz}{r^2(x^2+y^2)^{1/2}} & \frac{(x^2+y^2)^{1/2}}{r^2} \end{bmatrix}$$

$$\mathbf{J}_{r\theta\varphi} \equiv \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} & \frac{\partial x}{\partial \varphi} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} & \frac{\partial y}{\partial \varphi} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial \theta} & \frac{\partial z}{\partial \varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi\cos\theta & -r\cos\varphi\sin\theta & -r\sin\varphi\cos\theta \\ \cos\varphi\sin\theta & r\cos\varphi\cos\theta & -r\sin\varphi\sin\theta \\ \sin\varphi & 0 & r\cos\varphi \end{bmatrix}$$

The two Jacobian matrices are related by

$$[\mathbf{J}_{xyz}] \cdot [\mathbf{J}_{r\theta\phi}] = [\mathbf{I}] \qquad \text{(an identify matrix)}$$

or

$$[\mathbf{J}_{xyz}] = [\mathbf{J}_{r\theta\phi}]^{-1}$$

The measurement error covariance matrix $\mathbf{R}_k$ in CCS is obtained by

$$\begin{aligned} \mathbf{R}_k^{CCS} &= [\mathbf{J}_{r\theta\varphi}]\,[\mathbf{R}_k^{LOS}]\,[\mathbf{J}_{r\theta\varphi}]^T \\ &= \begin{bmatrix} \cos\varphi\cos\theta & -r\cos\varphi\sin\theta & -r\sin\varphi\cos\theta \\ \cos\varphi\sin\theta & r\cos\varphi\cos\theta & -r\sin\varphi\sin\theta \\ \sin\varphi & 0 & r\cos\varphi \end{bmatrix} \begin{bmatrix} \sigma_r^2 & & \\ & \sigma_\theta^2 & \\ & & \sigma_\varphi^2 \end{bmatrix} \\ &\quad \times \begin{bmatrix} \cos\varphi\cos\theta & \cos\varphi\sin\theta & \sin\varphi \\ -r\cos\varphi\sin\theta & r\cos\varphi\cos\theta & 0 \\ -r\sin\varphi\sin\theta & -r\sin\varphi\sin\theta & r\cos\varphi \end{bmatrix} \\ &\Rightarrow \begin{bmatrix} \sigma_x^2 & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_y\sigma_x & \sigma_y^2 & \sigma_y\sigma_z \\ \sigma_z\sigma_x & \sigma_z\sigma_y & \sigma_z^2 \end{bmatrix} \end{aligned}$$

Multiplying the three matrices, we obtain

$$\begin{aligned} \sigma_x^2 &= \sigma_r^2\cos^2\varphi\cos^2\theta - \sigma_\theta^2 r^2\cos^2\varphi\sin^2\theta + \sigma_\varphi^2 r^2\sin^2\varphi\cos^2\theta \\ \sigma_y^2 &= \sigma_r^2\cos^2\varphi\cos^2\theta + \sigma_\theta^2 r^2\cos^2\varphi\cos^2\theta + \sigma_\varphi^2 r^2\sin^2\varphi\sin^2\theta \\ \sigma_z^2 &= \sigma_r^2\sin^2\varphi + \sigma_\varphi^2 r^2\cos^2\varphi \end{aligned}$$

$$\sigma_x\sigma_y = 1/2\ \sin(2\theta)\ [\sigma_r^2\ \cos^2\varphi - \sigma_\theta^2 r^2\ \cos\varphi + \sigma_\varphi^2 r^2\ \sin^2\varphi]$$
$$\sigma_x\sigma_z = 1/2\ \sin(2\varphi)\ \cos\theta\ [\sigma_r^2 - \sigma_\varphi^2 r^2]$$
$$\sigma_y\sigma_z = 1/2\ \sin(2\varphi)\ \cos\theta\ [\sigma_r^2 - \sigma_\varphi^2 r^2]$$

We have obtained $\mathbf{R}_k$ in CCS coordinates in terms of the measurements in LOS coordinates. Next we shall find the numerical values of the variances.

The air defense radar is assumed to have a transmitter pulsewidth of 1.0 ms, unmodulated. The antenna beamwidth is 1.0 degree in azimuth and 1.5 degrees in elevation.

The three variances are:

$$\sigma_r^2 = \frac{(30)^2}{12} = 75.0\ \text{m}^2, \qquad \sigma_r = 8.66\ \text{m}$$
$$\sigma_\theta^2 = \frac{(1.0)^2}{12} = 2.5385\text{e}-5\ \text{rad}^2, \quad \sigma_\theta = 0.2887\ \text{degrees}$$
$$\sigma_\varphi^2 = \frac{(1.5)^2}{12} = 5.7116\text{e}-5\ \text{rad}^2, \quad \sigma_\varphi = 0.4330\ \text{degrees}$$

Next we derive the expressions for the initializations of $\mathbf{x}_2^+$ and $\mathbf{P}_2^+$. We need three measurements to initialize both.

$$\mathbf{x}_2^+ = \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \\ -- \\ y \\ \dot{y} \\ \ddot{y} \\ -- \\ z \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} x_2 \\ (x_2 - x_1)/T \\ (x_2 - 2x_1 + x_0)/T^2 \\ \text{----------} \\ y_2 \\ (y_2 - y_1)/T \\ (y_2 - 2y_1 + y_0)/T^2 \\ \text{----------} \\ z_2 \\ (z_2 - z_1)/T \\ (z_2 - 2z_1 + z_0)/T^2 \end{bmatrix}$$

where

$$\begin{cases} x_i = (r_i\ \cos\ \varphi_i\ \cos\ \theta_i) & i=0,1,2 \\ y_i = (r_i\ \cos\ \varphi_i\ \sin\ \theta_i) & i=0,1,2 \\ z_i = (r_i\ \sin\ \varphi_i) & i=0,1,2 \end{cases}$$

The initialization of $\mathbf{P}_2^+$ will be obtained from an expectation of

$$\mathbf{P}_2^+ = E\{\tilde{\mathbf{x}}_2\tilde{\mathbf{x}}_2^T\} = E\{(\mathbf{x}_2 - \hat{\mathbf{x}}_2)(\mathbf{x}_2 - \hat{\mathbf{x}}_2)^T\}$$

where the estimated position vector $\hat{\mathbf{x}}_2$ is given by

$$\hat{\mathbf{x}}_2 = \begin{bmatrix} \hat{x}_2 \\ \hat{\dot{x}}_2 \\ \hat{\ddot{x}}_2 \\ -- \\ \hat{y}_2 \\ \hat{\dot{y}}_2 \\ \hat{\ddot{y}}_2 \\ -- \\ \hat{z}_2 \\ \hat{\dot{z}}_2 \\ \hat{\ddot{z}}_2 \end{bmatrix} = \begin{bmatrix} x_2 + n_{x2} \\ [(x_2 + n_{x2}) - (x_1 + n_{x1})]/T \\ [(x_2 + n_{x2}) - 2(x_1 + n_{x1}) + (x_0 + n_{x0})]/T \\ \hline (\text{same as above with } x_0,\ x_1, \text{ and} \\ \ x_2 \text{ are replaced by } y_0,\ y_1, \text{ and} \\ \ y_2) \\ \hline (\text{same as above with } y_0,\ y_1, \text{ and} \\ \ y_2 \text{ are replaced by } z_0,\ z_1, \text{ and} \\ \ z_2) \end{bmatrix}$$

Therefore, the error vector $\tilde{\mathbf{x}}_2$ is given by

$$\tilde{\mathbf{x}}_2 = \mathbf{x}_2 - \hat{\mathbf{x}}_2 = \begin{bmatrix} -n_{x2} \\ -(n_{x2} - n_{x1})/T \\ -(n_{x2} - 2n_{x1} + n_{x0})/T^2 \\ \hline \text{same as above with} \\ \text{subscript } {}_{x0,x1}, \text{ and } {}_{x2} \\ \text{are replaced by} \\ {}_{y0,y1}, \text{ and } {}_{y2} \\ \hline \text{same as above with} \\ \text{subscript } {}_{y0,y1}, \text{ and } {}_{y2} \\ \text{are replaced by} \\ {}_{z0,z1}, \text{ and } {}_{z2} \end{bmatrix}$$

The expectation operation $E\{(\mathbf{x}_2 - \hat{\mathbf{x}}_2)(\mathbf{x}_2 - \hat{\mathbf{x}}_2)^T\}$ will yield initialization of $\mathbf{P}_2^+$ as shown below after a lengthy but straightforward algebraic manipulation.

$$\mathbf{P}_2^+ = \left[\begin{array}{ccc|c|c} \sigma_x^2 & \sigma_x^2/T & \sigma_x^2/T^2 & & \\ \sigma_x^2/T & 2\sigma_x^2/T^2 & 3\sigma_x^2/T^3 & \mathbf{0} & \mathbf{0} \\ \sigma_x^2/T^2 & 3\sigma_x^2/T^3 & 6\sigma_x^2/T^4 & & \\ \hline & & & & \\ \mathbf{0} & & & (\text{same submatrix} & \mathbf{0} \\ & & & \text{block except } \sigma_x^2 & \\ & & & \text{is replaced by } \sigma_y^2) & \\ \hline & & & & (\text{same submatrix} \\ \mathbf{0} & & & \mathbf{0} & \text{block except } \sigma_y^2 \\ & & & & \text{is replaced by } \sigma_z^2) \end{array}\right]$$

The Kalman recursive equations for this program are summarized below.

$$\mathbf{P}_k^- = \mathbf{\Phi}\mathbf{P}_{k-1}^+\mathbf{\Phi}^T + \mathbf{Q} \quad \text{(no index on Q)} \tag{8.29}$$

$$\mathbf{k}_k = \mathbf{P}_k^-\mathbf{H}^T[\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}_k]^{-1} \tag{8.30}$$

$$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{k}_k\mathbf{H}]\ \mathbf{P}_k^-[\mathbf{I} - \mathbf{k}_k\mathbf{H}]^T + \mathbf{k}_k\mathbf{R}\mathbf{k}_k^T \tag{8.31}$$

$$\mathbf{x}_k^+ = \mathbf{x}_k^- + \mathbf{k}_k[\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-] \tag{8.32}$$

$$\mathbf{x}_{k+1}^- = \mathbf{\Phi}\mathbf{x}_k^+ \tag{8.33}$$

A flow diagram is shown in Figure 8.22. The program is written in KAL_ MIG. CPP. Five data files are created to record the results:

1. MIG_Ppre.DAT: Predicted error covariance matrix. Complete data is consisted of 81 elements per matrix, and there are 128 samplings. We have recorded the principal elements only.

   p(0,0) = [P_pre]_sub_xx

   p(3,3) = [P_pre]_sub_yy

   p(6,6) = [P_pre]_sub_zz

2. MIG_Pest.DAT: Estimated error covariance matrix. We have recorded the principal elements only.

   p(0,0) = [P_est]_sub_xx

   p(3,3) = [P_est]_sub_yy

   p(6,6) = [P_est]_sub_zz

3. MIG_K.DAT: Kalman gain matrix, nine-by-three matrix per sampling. Only the principal elements are recorded.

   k(0,0) = k_sub_xx

   k(3,1) = k_sub_yy

   k(6,2) = k_sub_zz

4. MIG_Xpre.DAT: Predicted position in CCS. The predicted position vector is nine-by-one. Only the principal elements are recorded.

   Xpre(0,0) = Xpre_sub_x

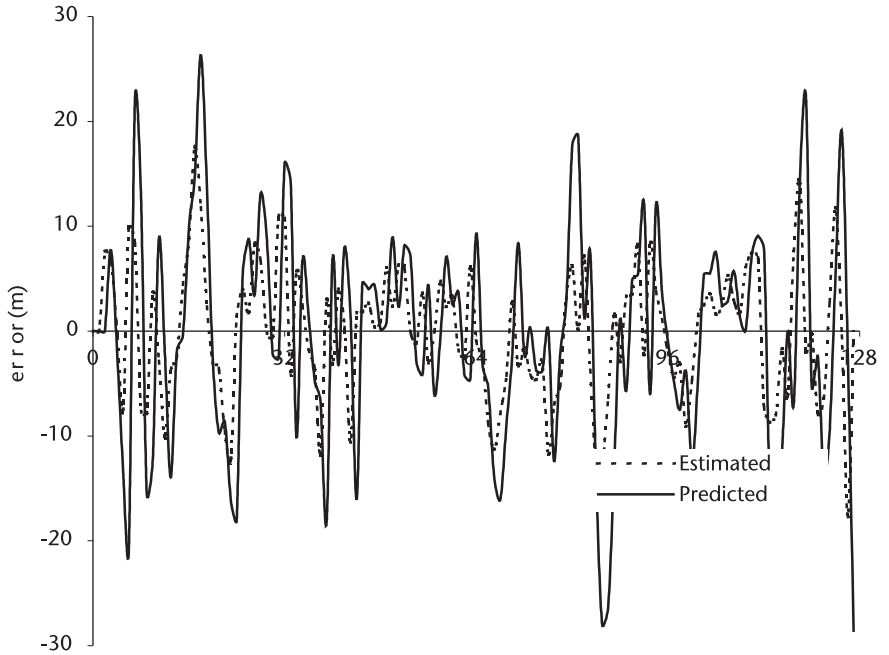


Figure 8.19 Estimated and predicted error in range.

$$Xpre(3,0) = Xpre\_sub\_y$$

$$Xpre(8,0) = Xpre\_sub\_z$$

5. MIG_Xest.DAT: Estimated positon in CCS. The estimated position vector is nine-by-one. The principal elements only

$$Xest(0,0) = Xest\_sub\_x$$

$$Xest(3,0) = Xest\_sub\_y$$

$$Xest(8,0) = Xest\_sub\_z$$

Finally the differences (errors) between the trajectory and the predicted or estimated positions are computed in ERR_MIG.CPP, and the results are shown in Figures 8.19–8.21. To some readers the results may appear that the predicted or estimated positions are not accurate enough. Two possible remedies are suggested: either increase the antenna size or the carrier frequency.

If this radar is employed for a countermunition guidance control and the munition is armed with a proximity fuse, the error in range is less important. We may increase the transmitter pulsewidth for higher transmit energy.

We have formulated the target state equation in CCS coordinates in using Newtonian dynamics with random acceleration noise n_{ax}, n_{ay}, and n_{az}. The target is assumed capable of a ±3 g maneuver. The matrix **Q** is found to be time-invariant.

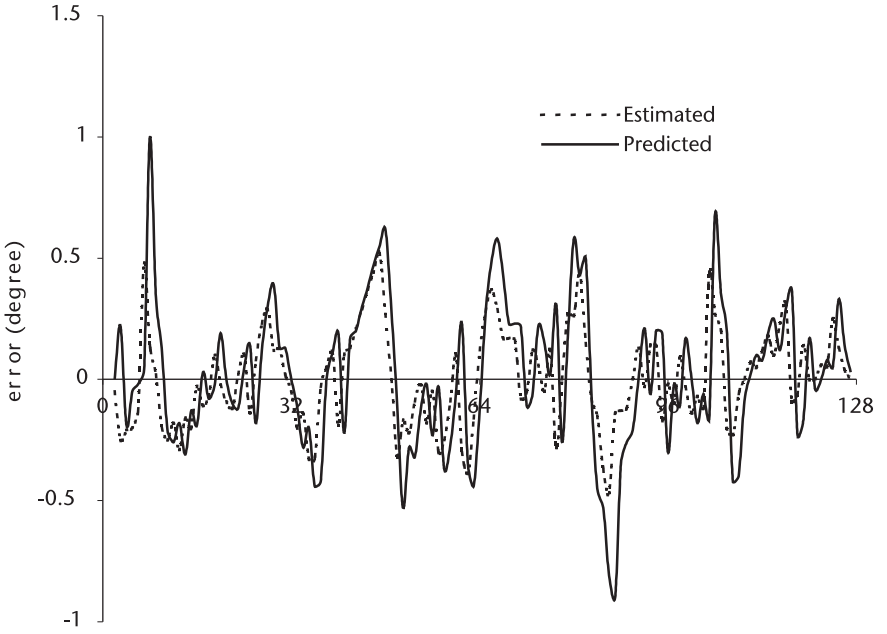


Figure 8.20 Estimated and predicted error in azimuth.

The matrix $\mathbf{R}_k$ is obtained through a Jacobian transform and found to be measurement-dependent.

The inversion of matrix $[\ \mathbf{H}\ \mathbf{P}_{\bar{k}}\ \mathbf{H}^T + \mathbf{R}_k\ ]^{-1}$ is executed after $[\mathbf{U}][\mathbf{D}][\mathbf{U}]^T$ factorization. The inversion through factorization has a superior numerical stability. It maintains the symmetry of error covariance matrices and prevents the birth of negative elements.

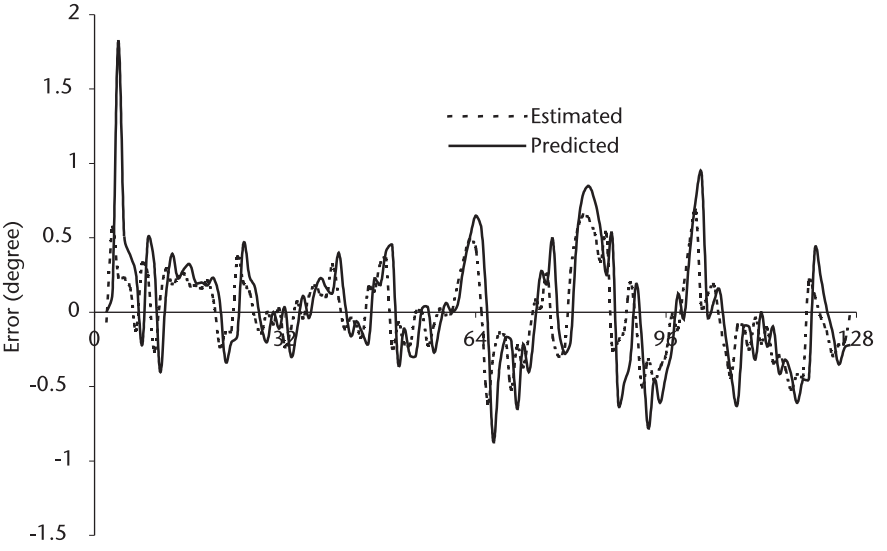


Figure 8.21 Estimated and predicted error in elevation.

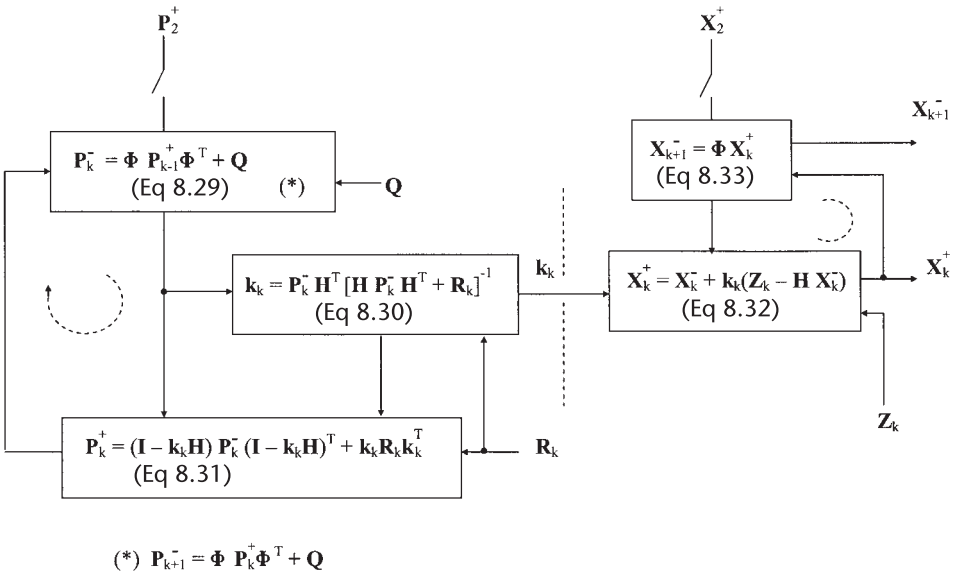


Figure 8.22 Flow diagram of Kalman recursive filter (KAL_MIG.CCP).

8.6 Air Defense Radar, LOS Coordinates

In this section we track the same fighter-bomber on a ground-strafing mission. In the previous section we wrote the state equation in CCS coordinates. For the present demonstration we write the state equation in LOS coordinates.

The state equations are

$$\begin{cases} r_{k+1} = r_k + \dot{r}_k T + 1/2 \ddot{r}_k T^2 \\ \dot{r}_{k+1} = \dot{r}_k + \ddot{r}_k T \\ \ddot{r}_{k+1} = \ddot{r}_k + n_{ar} \end{cases}$$

$$\begin{cases} \theta_{k+1} = \theta_k + \dot{\theta}_k T + 1/2 \ddot{\theta}_k T^2 \\ \dot{\theta}_{k+1} = \dot{\theta}_k + \ddot{\theta}_k T \\ \ddot{\theta}_{k+1} = \ddot{\theta}_k + n_{a\theta} \end{cases}$$

$$\begin{cases} \varphi_{k+1} = \varphi_k + \dot{\varphi}_k T + 1/2 \ddot{\varphi}_k T^2 \\ \dot{\varphi}_{k+1} = \dot{\varphi}_k + \ddot{\varphi}_k T \\ \ddot{\varphi}_{k+1} = \ddot{\varphi}_k + n_{a\varphi} \end{cases}$$

The state equation in a vector-matrix form is

$$\begin{bmatrix} r_{k+1} \\ \dot{r}_{k+1} \\ \ddot{r}_{k+1} \\ \theta_{k+1} \\ \dot{\theta}_{k+1} \\ \ddot{\theta}_{k+1} \\ \varphi_{k+1} \\ \dot{\varphi}_{k+1} \\ \ddot{\varphi}_{k+1} \end{bmatrix} = \left[\begin{array}{ccc:ccc:ccc} 1 & T & T^2/2 & & & & & & \\ 0 & 1 & T & & \mathbf{0} & & & \mathbf{0} & \\ 0 & 0 & 1 & & & & & & \\ \hdashline & & & 1 & T & T^2/2 & & & \\ & \mathbf{0} & & 0 & 1 & T & & \mathbf{0} & \\ & & & 0 & 0 & 1 & & & \\ \hdashline & & & & & & 1 & T & T^2/2 \\ & \mathbf{0} & & & \mathbf{0} & & 0 & 1 & T \\ & & & & & & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} r_k \\ \dot{r}_k \\ \ddot{r}_k \\ \theta_k \\ \dot{\theta}_k \\ \ddot{\theta}_k \\ \varphi_k \\ \dot{\varphi}_k \\ \ddot{\varphi}_k \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ n_{ar} \\ 0 \\ 0 \\ n_{a\theta} \\ 0 \\ 0 \\ n_{a\varphi} \end{bmatrix}$$

where n_{ar}, $n_{a\varphi}$, and $n_{a\varphi}$ are random acceleration noise in range, azimuth, and elevation angle, respectively. The state equation in a more compact vector-matrix form is

$$\mathbf{x}_{k+1} = \Phi\, \mathbf{x}_k + \mathbf{w}_k$$

and the corresponding measurement equation is

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k$$

$$\begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} r \\ \dot{r} \\ \ddot{r} \\ \theta \\ \dot{\theta} \\ \ddot{\theta} \\ \varphi \\ \dot{\varphi} \\ \ddot{\varphi} \end{bmatrix} + \begin{bmatrix} n_r \\ n_\theta \\ n_\varphi \end{bmatrix}$$

where n_r, n_θ, and n_φ are the measurement errors in range, azimuth, and elevation angle respectively. The measurement error covariance matrix $\mathbf{R}_k$ is given by

$$\mathbf{R}_k = E\{\mathbf{v}_k \mathbf{v}_k^T\} = E\left\{ \begin{bmatrix} n_r \\ n_\theta \\ n_\varphi \end{bmatrix} \begin{bmatrix} n_r & n_\theta & n_\varphi \end{bmatrix} \right\} = \begin{bmatrix} \sigma_r^2 & & \\ & \sigma_\theta^2 & \\ & & \sigma_\varphi^2 \end{bmatrix}$$

where $E\{n_i\, n_j\} = 0$, when $i \neq j$, uncorrelated noise.

The variance in range, azimuth, and elevation angle are taken to be identical to those in the previous program, KAL_MIG.CPP. They are

$$\sigma_r^2 = 75.0 \ \text{m}^2$$

$$\sigma_\theta^2 = 2.5386\text{e}{-5} \ \text{rad}^2$$

$$\sigma_\varphi^2 = 5.7116\text{e}{-5} \ \text{rad}^2$$

We note that matrix $\mathbf{R}_k$ is time-invariant. The elements of the corresponding matrix in CCS coordinates are lengthy trigonometric expressions through a Jacobian transform. We have lightened the computation loads in the present example. (Trigonometric computation is a lengthy sum of power series expansions.)

The state error covariance matrix $\mathbf{Q}_k$ is given by

$$\mathbf{Q}_k = \mathrm{E}\{\mathbf{w}_k\mathbf{w}_k^{\mathrm{T}}\} = \begin{bmatrix} 0 & 0 & 0 & & & & & & \\ 0 & 0 & 0 & \mathbf{0} & & & \mathbf{0} & & \\ 0 & 0 & \mathrm{E}\{n_{ar}^2\} & & & & & & \\ & & & 0 & 0 & 0 & & & \\ & \mathbf{0} & & 0 & 0 & 0 & \mathbf{0} & & \\ & & & 0 & 0 & \mathrm{E}\{n_{a\theta}^2\} & & & \\ & & & & & & 0 & 0 & 0 \\ & \mathbf{0} & & \mathbf{0} & & & 0 & 0 & 0 \\ & & & & & & 0 & 0 & \mathrm{E}\{n_{a\varphi}^2\} \end{bmatrix}$$

where $\mathrm{E}\{n_{ar}^2\} = \sigma_{ar}^2$, $\mathrm{E}\{n_{a\theta}^2\} = \sigma_{a\theta}^2$, $\mathrm{E}\{n_{a\varphi}^2\} = \sigma_{a\varphi}^2$.

In the previous program, KAL_MIG.CPP, we had $\sigma_{ax}^2 = \sigma_{ay}^2 = \sigma_{az}^2 = 288.12 \ \text{m}^2$. Therefore, in order to be equivalent, we assign

$$\sigma_{ar}^2 = \sqrt{3}\,\sigma_x^2$$

$$\sigma_{a\theta}^2 = \sqrt{2}\,\sigma_x^2/r_k^2$$

$$\sigma_{a\varphi}^2 = \sqrt{2}\,\sigma_x^2/r_k^2$$

We note that matrix $\mathbf{Q}_k$ is measurement-dependent whereas in KAL_MIG. CPP it was time-invariant. The time invariance and the measurement dependency of $\mathbf{R}_k$ and $\mathbf{Q}_k$ are reversed in LOS coordinates.

The recursive filter processing is programmed in KAL_F16.CPP. The target trajectory is identical to that of KAL_MIG.CPP. The algorithm flow diagram is also identical with the exception of $\mathbf{R}_k$ and $\mathbf{Q}_k$. The initialization of $\mathbf{x}_3^+$ and $\mathbf{P}_3^+$ follows the rules previously derived.

Five data files are created out of KAL_F16.CPP to record the filter performance.

1. F16_Ppre.DAT: Predicted error covariance matrix, this matrix has 81 elements per sampling instant. It is too voluminous to record the entire data for 128 samples. The principal elements only are recorded.

Ppre(0,0) = Ppre_rr (range)

Ppre(3,3) = Ppre_aa (azimuth)

Ppre(6,6) = Ppre_ee (elevation)

2. F16_Pest.DAT: Estimated error covariance matrix. The principal elements only.

Pest(0,0) = Pest_rr (range)

Pest(3,3) = Pest_aa (azimuth)

Pest(6,6) = Pest_ee (elevation)

3. F16_K.DAT: Kalman gain matrix. The gain matrix has nine-by-three per sampling. The principal elements only.

K(0,0) = K_rr (range)

K(3,1) = K_aa (azimuth)

K(6,2) = K_ee (elevation)

4. F16_Xpre.DAT: Predicted position vector. The vector has nine-by-one elements for each sampling instant. Three principal elements only.

Xpre(0,0) = Xpre_rng

Xpre(3,0) = Xpre_azd

Xpre(6,9) = Xpre_el

5. F16_Xest.DAT: Estimated position vector. The vector has nine-by-one, nine elements per sampling. Three principal elements only.

Xest(0,0) = Xest_rng

Xest(3,0) = Xest_azd

Xest(6,0) = Xest_eld

In order to evaluate performance of this filter the errors between the true trajectory and the predicted or estimated positions are computed in F16_ ERR1.CPP and F16_ERR2.CPP. The results are shown in Figures 8.23–8.25.

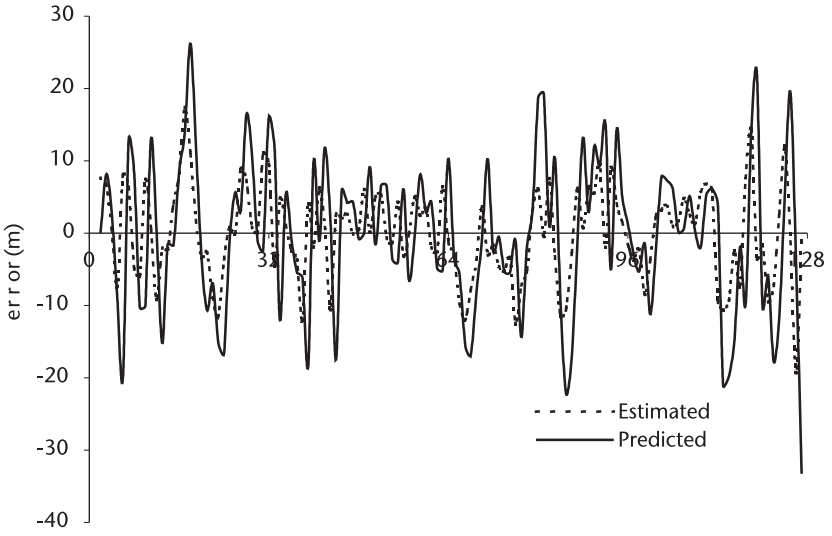


Figure 8.23 Errors in estimated and predicted range.

An examination of the figures indicates that the filter performance is comparable to KAL_MIG.CPP in CCS coordinates, even though the computation load is considerably lighter due to the structures of the $\mathbf{R}_k$ and $\mathbf{Q}_k$ matrices. This is not always true; we have to test both filters and choose accordingly.

For an air defense radar the primary importance is the accuracy of azimuth and elevation angles. The measures to improve the performance discussed previously apply in LOS coordinate processing as well.

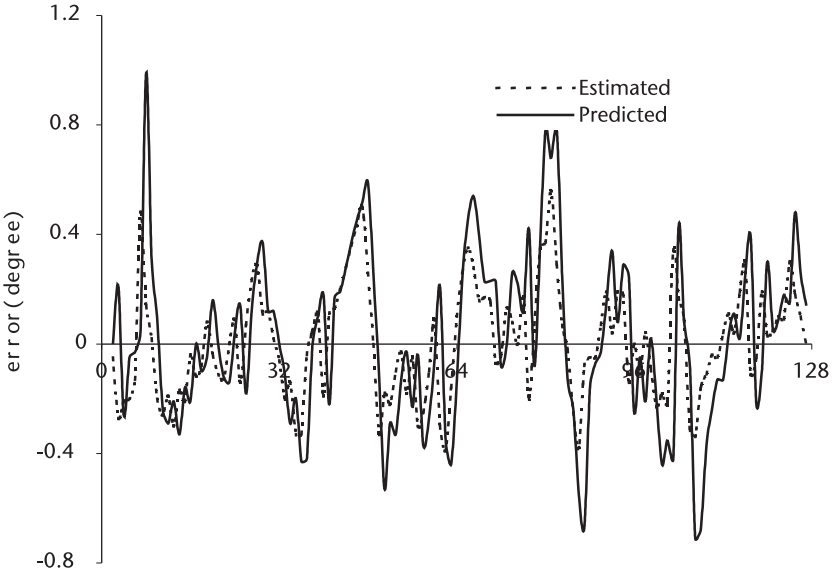


Figure 8.24 Errors in estimated and predicted azimuth angle.

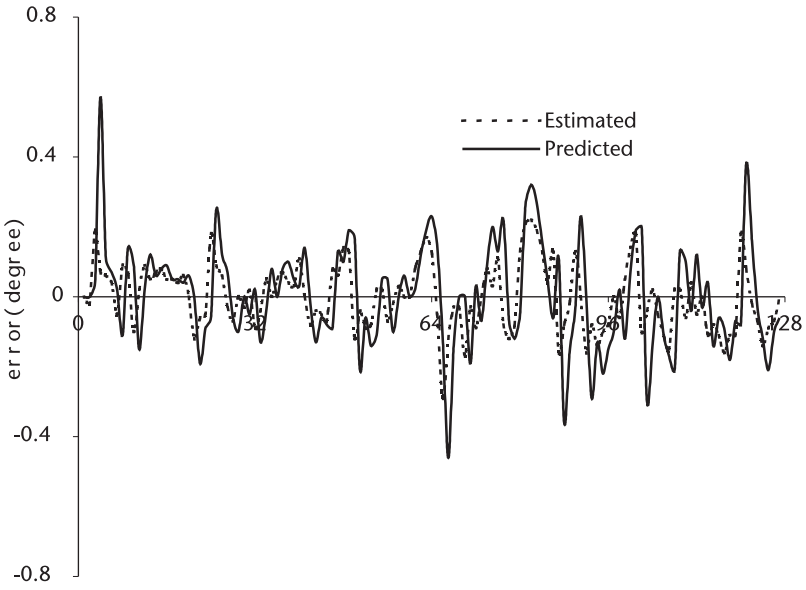


Figure 8.25 Errors in estimated and predicted elevation.

8.7 Kalman Filter without Matrix Inversion

For Kalman filter processing where matrix inversion is always required the round-off error can be a problem with small word-length machines. A class of Kalman filter processing with "square-root" algorithms [8, 9] has been developed for a better numerical behavior; however, it has not completely eliminated the problems.

Bierman [10] was the first to report a filter processing without matrix inversion through $\mathbf{UDU}^{\mathrm{T}}$ factorization for numerical stability. We study his inversionless algorithm in this section. We follow his development and appreciate the superiority of his approach.

Estimated or predicted error covariance matrix can be factorized (decom-posed) as follows, provided that the matrix is symmetrical and positive definite. We shall use an overhead caret (^) and an overhead tilde (~) for the estimated and predicted error covariance matrices respectively.

$$\hat{\mathbf{P}}_k = \hat{\mathbf{U}}_k \, \hat{\mathbf{D}}_k \, \hat{\mathbf{U}}_k^{\mathrm{T}} \tag{8.34}$$

$$\tilde{\mathbf{P}}_k = \tilde{\mathbf{U}}_k \, \tilde{\mathbf{D}}_k \, \tilde{\mathbf{U}}_k^{\mathrm{T}} \tag{8.35}$$

For example,

$$\begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} = \begin{bmatrix} 1 & u_{01} & u_{02} & u_{03} \\ 0 & 1 & u_{12} & u_{13} \\ 0 & 0 & 1 & u_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_{00} & & & \\ & d_{11} & & \\ & & d_{22} & \\ & & & d_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ u_{01} & 1 & 0 & 0 \\ u_{02} & u_{12} & 1 & 0 \\ u_{03} & u_{13} & u_{23} & 1 \end{bmatrix}$$

The $\mathbf{UDU}^{\mathrm{T}}$ factorization is programmed in Chapter 1. From previous analysis the estimated error covariance matrix update is given by

$$\begin{aligned}\hat{\mathbf{P}}_k &= [\mathbf{I} - \mathbf{k}_k\mathbf{H}]\,\widetilde{\mathbf{P}}_k \\ &= \widetilde{\mathbf{P}}_k - \mathbf{k}_k\mathbf{H}\,\widetilde{\mathbf{P}}_k \\ &= \widetilde{\mathbf{P}}_k - \widetilde{\mathbf{P}}_k\,\mathbf{H}^{\mathrm{T}}[\mathbf{H}\widetilde{\mathbf{P}}_k\mathbf{H}^{\mathrm{T}} + \mathbf{R}_k]^{-1}\mathbf{H}\widetilde{\mathbf{P}}_k\end{aligned}$$

If the bracketed term, without inversion, is a scalar, (we shall show this shortly) the above expression can be written as

$$\begin{aligned}\hat{\mathbf{P}}_k &= \widetilde{\mathbf{P}}_k - \widetilde{\mathbf{P}}_k\mathbf{H}^{\mathrm{T}}\left(\frac{1}{\mathrm{s}}\right)\mathbf{H}\,\tilde{\mathbf{P}}_k \\ &= \widetilde{\mathbf{U}}_k\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}} - \left(\frac{1}{\mathrm{s}}\right)(\widetilde{\mathbf{U}}_k\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}})\mathbf{H}^{\mathrm{T}}\,\mathbf{H}\,(\widetilde{\mathbf{U}}_k\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}}) \\ &= \widetilde{\mathbf{U}}_k[\widetilde{\mathbf{D}}_k - \left(\frac{1}{\mathrm{s}}\right)\,(\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})\,(\mathbf{H}\tilde{\mathbf{U}}_k\widetilde{\mathbf{D}}_k)]\widetilde{\mathbf{U}}_k^{\mathrm{T}} \\ &= \widetilde{\mathbf{U}}_k[\widetilde{\mathbf{D}}_k - \left(\frac{1}{\mathrm{s}}\right)(\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})(\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{\mathrm{T}}]\widetilde{\mathbf{U}}_k^{\mathrm{T}}\end{aligned} \tag{8.36}$$

Since $(\tilde{\mathrm{D}}_k\tilde{\mathrm{U}}_k^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})\,(\tilde{\mathrm{D}}_k\tilde{\mathrm{U}}_k^{\mathrm{T}}\,\mathbf{H}^{\mathrm{T}})^{\mathrm{T}}$ is symmetric and positive-definite the bracketed term is symmetric and positive-definite. Therefore we can apply $\mathbf{UDU}^{\mathrm{T}}$ factorization to the bracketed term.

$$\left[\widetilde{\mathbf{D}}_k - \left(\frac{1}{\mathrm{s}}\right)(\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})\,(\widetilde{\mathbf{D}}_k\widetilde{\mathbf{U}}_k^{\mathrm{T}}\mathbf{H}^{\mathrm{T}})^{\mathrm{T}}\right] \rightarrow \bar{\mathbf{U}}_k\bar{\mathbf{D}}_k\bar{\mathbf{U}}_k^{\mathrm{T}} \tag{8.37}$$

Then, the estimated error covariance matrix $\hat{\mathbf{P}}_k$ will be expressed as

$$\begin{aligned}\hat{\mathbf{P}}_k &= \widetilde{\mathbf{U}}_k(\bar{\mathbf{U}}_k\bar{\mathbf{D}}_k\bar{\mathbf{U}}_k^{\mathrm{T}})\widetilde{\mathbf{U}}_k^{\mathrm{T}} \\ &= (\widetilde{\mathbf{U}}_k\bar{\mathbf{U}}_k)\,\bar{\mathbf{D}}_k(\bar{\mathbf{U}}_k^{\mathrm{T}}\widetilde{\mathbf{U}}_k^{\mathrm{T}}) \\ &= (\widetilde{\mathbf{U}}_k\bar{\mathbf{U}}_k)\,\bar{\mathbf{D}}_k(\widetilde{\mathbf{U}}_k^{\mathrm{T}}\bar{\mathbf{U}}_k^{\mathrm{T}})^{\mathrm{T}} \\ &= \hat{\mathbf{U}}_k\hat{\mathbf{D}}_k\hat{\mathbf{U}}_k^{\mathrm{T}}\end{aligned} \tag{8.38}$$

where we use $\hat{\mathbf{U}}_k = \tilde{\mathbf{U}}_k\,\bar{\mathbf{U}}_k$ and $\hat{\mathbf{D}}_k = \bar{\mathbf{D}}_k$ for shorthand notation.

We have proved that the estimated error covariance matrix $\hat{\mathbf{P}}_k$ can be factorized as $\hat{\mathbf{U}}_k\,\hat{\mathbf{D}}_k\,\hat{\mathbf{U}}_k^{\mathrm{T}}$.

Next we prove that the Kalman gain matrix $\mathbf{k}_k$ can be computed without matrix inversion. The Kalman gain matrix is given by

$$\mathbf{k}_k = \tilde{\mathbf{P}}_k\,\mathbf{H}^{\mathrm{T}}\,[\,\mathbf{H}\tilde{\mathbf{P}}_k\,\mathbf{H}^{\mathrm{T}} + \mathbf{R}_k\,]^{-1}$$

Let

$$\widetilde{\mathbf{P}}_k = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix}_k \quad \mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{R}_k = \begin{bmatrix} r_{00} & r_{01} \\ r_{10} & r_{11} \end{bmatrix}_k$$

Then, $$[\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}_k] = \begin{bmatrix} p_{00} + r_{00} & p_{02} + r_{01} \\ p_{20} + r_{10} & p_{22} + r_{11} \end{bmatrix}$$

An inversion of the above matrix is given by

$$[\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}_k]^{-1} = \frac{1}{\Delta} \begin{bmatrix} p_{22} + r_{11} & -(p_{02} + r_{01}) \\ -(p_{20} + r_{10}) & p_{00} + r_{00} \end{bmatrix} \tag{8.39}$$

where $\Delta = (p_{00} + r_{00})(p_{22} + r_{11}) - (p_{02} + r_{01})(p_{20} + r_{10})$

When the predicted error covariance matrix $\tilde{P}_k$ is a block diagonal and the measurement error covariance $\mathbf{R}_k$ is also a diagonal; that is,

$$\widetilde{\mathbf{P}}_k = \begin{bmatrix} p_{00} & p_{01} & & \\ p_{10} & p_{11} & & \\ & & p_{22} & p_{23} \\ & & p_{32} & p_{33} \end{bmatrix}_k \quad \mathbf{R}_k = \begin{bmatrix} r_{00} & 0 \\ 0 & r_{11} \end{bmatrix}_k$$

Equation (8.39) can be written as

$$[\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}_k]^{-1} = \begin{bmatrix} \dfrac{1}{(p_{00} + r_{00})} & 0 \\ 0 & \dfrac{1}{(p_{22} + r_{11})} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{s_0} & 0 \\ 0 & \dfrac{1}{s_1} \end{bmatrix}$$

As we have seen above, the inverted matrix is a scalar division of the elements. Finally the gain matrix is given by

$$\begin{aligned} \mathbf{k}_k &= \widetilde{\mathbf{P}}_k \mathbf{H}^T [\mathbf{H}\widetilde{\mathbf{P}}_k \mathbf{H}^T + \mathbf{R}_k]^{-1} \\ &= \begin{bmatrix} p_{00} & p_{01} \\ p_{10} & p_{12} \\ p_{20} & p_{22} \\ p_{30} & p_{32} \end{bmatrix} \begin{bmatrix} 1/s_0 & 0 \\ 0 & 1/s_1 \end{bmatrix} = \begin{bmatrix} p_{00}/s_0 & p_{02}/s_1 \\ p_{10}/s_0 & p_{12}/s_1 \\ p_{20}/s_0 & p_{22}/s_1 \\ p_{30}/s_0 & p_{32}/s_1 \end{bmatrix} \end{aligned} \tag{8.40}$$

We have proven that gain matrix $\mathbf{k}_k$ can be obtained without matrix inversion. In general form, $\mathbf{k}_k$ is given by

$$\mathbf{k}_k = \begin{bmatrix} p_{00}/s_0 & p_{02}/s_1 & .... & p_{0m}/s_{m-1} \\ p_{10}/s_0 & p_{12}/s_1 & .... & p_{1m}/s_{m-1} \\ p_{20}/s_0 & p_{22}/s_1 & .... & p_{2m}/s_{m-1} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ p_{n0}/s_0 & p_{n2}/s_1 & .... & p_{nm}/s_{m-1} \end{bmatrix} \tag{8.41a}$$

where

$$s_j = \mathbf{H}_j \widetilde{\mathbf{P}}_k \mathbf{H}_k^T + r_{jj} \tag{8.41b}$$

$$p_{ij} = \mathbf{H}_{ij} \widetilde{\mathbf{P}}_k \mathbf{H}_{ij}^T + r_{jj} \tag{8.41c}$$

Next we shall prove that $\tilde{\mathbf{P}}_k$ is a block diagonal. When $\hat{\mathbf{P}}_k$ is a block diagonal and $\mathbf{Q}_k$ is also a diagonal, the predicted error covariance matrix $\tilde{\mathbf{P}}_k$ will remain block diagonal:

$$\begin{aligned} \widetilde{\mathbf{P}}_k &= \mathbf{\Phi} \hat{\mathbf{P}}_k \mathbf{\Phi}^T + \mathbf{Q}_k \\ \widetilde{\mathbf{U}}_k \widetilde{\mathbf{D}}_k \widetilde{\mathbf{U}}_k^T &= \mathbf{\Phi}[\hat{\mathbf{U}}_k \hat{\mathbf{D}}_k \hat{\mathbf{U}}_k^T] \mathbf{\Phi}^T + \mathbf{Q}_k \\ &= \mathbf{\Phi}(\widetilde{\mathbf{U}}_k \bar{\mathbf{D}}_k) \hat{\mathbf{D}}_k (\widetilde{\mathbf{U}}_k^T \bar{\mathbf{D}}_k) \mathbf{\Phi}^T + \mathbf{Q}_k \\ &= [(\mathbf{\Phi} \tilde{\mathbf{U}}_k \bar{\mathbf{D}}_k) \hat{\mathbf{D}}_k (\mathbf{\Phi} \tilde{\mathbf{U}}_k \bar{\mathbf{D}}_k)^T] + \mathbf{Q}_k \end{aligned}$$

Note that the second circular bracketed term on the right-hand side is a transpose of the first circular term. Therefore the square bracketed term is diagonal.

We have proved that the gain matrix $\mathbf{k}_k$ can be obtained without matrix inversion and that both error covariance matrices, $\tilde{\mathbf{P}}_k$ and $\hat{\mathbf{P}}_k$, can be factorized as $\mathbf{UDU}^T$. All we have to do is factorization of the two error covariances for every sampling instant.

Summarized:

$$\widetilde{\mathbf{P}}_k = \mathbf{\Phi}\ \hat{\mathbf{P}}_k \mathbf{\Phi}^T + \mathbf{Q}_k = \widetilde{\mathbf{U}}_k \widetilde{\mathbf{D}}_k \widetilde{\mathbf{U}}_k^T \tag{8.42}$$

$$\mathbf{k}_k = \tilde{\mathbf{P}}_k \mathbf{H}^T [\mathbf{H} \mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}_k]^{-1} \tag{8.43}$$

$$\begin{cases} s_j = \mathbf{H}_j \widetilde{\mathbf{P}}_k \mathbf{H}_k^T + r_{jj} & (8.44) \\ p_{ij} = \mathbf{H}_{ij} \widetilde{\mathbf{P}}_k \mathbf{H}_{ij}^T + r_{jj} & (8.45) \end{cases}$$

$$\begin{aligned} \widetilde{\mathbf{D}}_k &= \left(\frac{1}{s}\right) (\widetilde{\mathbf{D}}_k \widetilde{\mathbf{U}}_k^T \mathbf{H}^T)(\widetilde{\mathbf{D}}_k \widetilde{\mathbf{U}}_k^T \mathbf{H}^T)^T \\ &= \bar{\mathbf{U}}_k \bar{\mathbf{D}}_k \bar{\mathbf{U}}_k^T \end{aligned} \tag{8.46}$$

$$\hat{\mathbf{P}}_k = (\widetilde{\mathbf{U}}_k \bar{\mathbf{U}}_k) \bar{\mathbf{D}}_k (\widetilde{\mathbf{U}}_k^T \bar{\mathbf{U}}_k^T)^T = \hat{\mathbf{U}}_k \hat{\mathbf{D}}_k \hat{\mathbf{U}}_k^T \tag{8.47}$$

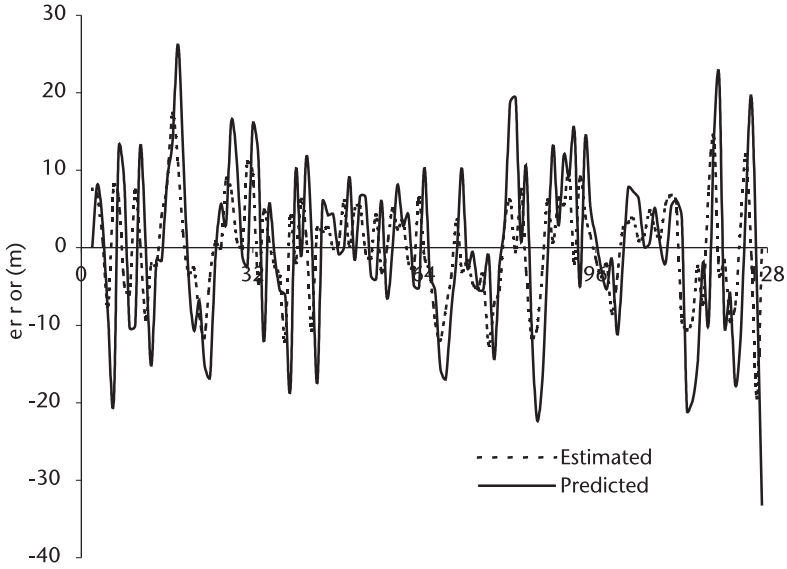


Figure 8.26 Errors in estimated and predicted range.

The estimated and predicted states are the same as before,

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k + \mathbf{k}_k[\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k] \tag{8.48}$$

$$\hat{\mathbf{x}}_{k+1} = \Phi\hat{\mathbf{x}}_k \tag{8.49}$$

A flow diagram of the Kalman filter without matrix inversion is shown in Figure 8.29. The predicted and estimated error covariances are factorized into $\mathbf{UDU}^T$ in the left loop. The loop on the right computes the estimated and predicted states. Two loops are connected by the Kalman gain $\mathbf{k}_k$.

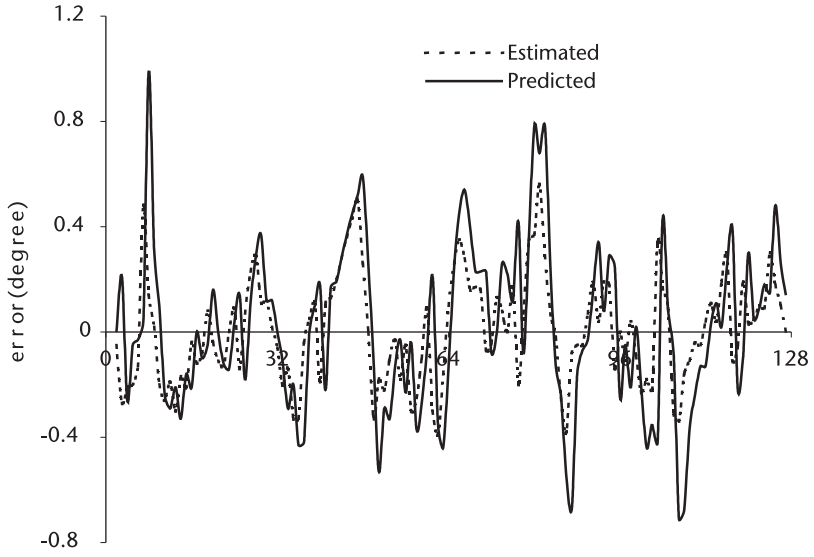


Figure 8.27 Errors in estimated and predicted range.

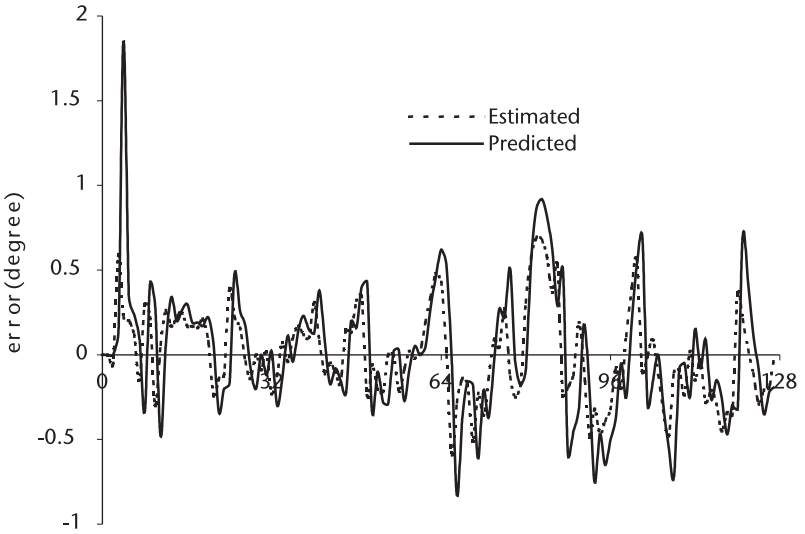


Figure 8.28 Errors in estimated and predicted azimuth.

The filter is programmed in KAL_NMI.CPP. "_NMI" stands for NoMatrixInversion. Five data files record the results:

1. NMI_Ppre.DAT: Predicted error covariance matrix; only principal elements are recorded.
2. NMI_Pest.DAT: Estimated error covariance matrix, the principal element only;

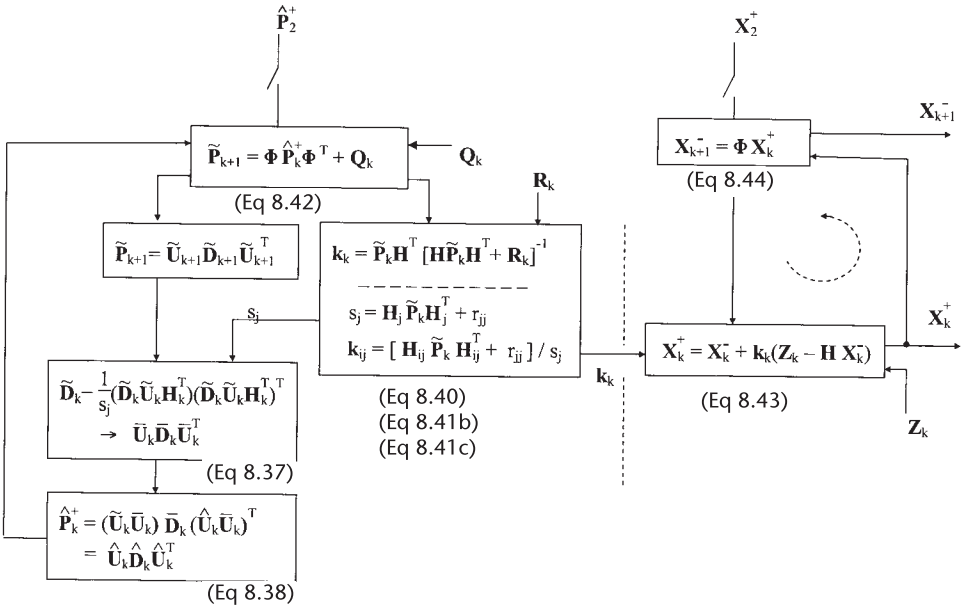


Figure 8.29 Errors in estimated and predicted elevation.

3. NMI_K.DAT: Kalman gain matrix; the principal;elements only;
4. NMI_Xpre.DAT: Predicted state vector; the principal elements only;
5. NMI_Xest.DAT: Estimated state vector, only the principal elements.

The errors between the trajectory and the predicted or estimated positions are computed in ERR_NMI.CPP, and the results are shown in Figures 8.26–8.28.

The difference between the no-matrix-inversion processing and KAL_F16. CPP is hardly discernable except for the elevation angle error. The $\mathbf{UDU}^{\mathrm{T}}$ factorization of the error covariance matrix is a powerful algorithm. Interested readers should consult Bierman [10] who has tabulated the computation load of several algorithms. The savings in computation through $\mathbf{UDU}^{\mathrm{T}}$ factorization and the inversion-free computation of gain matrix $\mathbf{k}_k$ is very substantial.

When the target state equation is written in CCS coordinates and the measurement equation in LOS coordinates as in KAL_F16.CPP, the measurement error matrix $\mathbf{R}_k$ is symmetric but not necessarily a diagonal. For such a case we apply $\mathbf{UDU}^{\mathrm{T}}$ factorization on $\mathbf{R}_k$ as follows so that decorrelated $\mathbf{R}k'$ becomes a diagonal.

$$\mathbf{R}_k = \mathrm{E}\{\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}\} = \begin{bmatrix} \sigma_x^2 & \sigma_x\sigma_y & \sigma_x\sigma_z \\ \sigma_y\sigma_x & \sigma_y^2 & \sigma_y\sigma_z \\ \sigma_z\sigma_x & \sigma_z\sigma_y & \sigma_z^2 \end{bmatrix} \rightarrow \mathbf{U}_k\mathbf{D}_k\mathbf{U}_k^{\mathrm{T}} \rightarrow \mathbf{D}_k$$

Then a new measurement vector $\mathbf{z}_k'$ and a new measurement matrix $\mathbf{H}_k'$ an be written as

$$\mathbf{z}_k' = \mathbf{U}_k^{-1}\,\mathbf{z}_k + \mathbf{v}_k'$$

$$\mathbf{H}' = \mathbf{U}_k^{-1}\,\mathbf{x}_k + \mathbf{v}_k'$$

The modified measurement error matrix $\mathbf{R}_k'$will be

$$\begin{aligned} \mathbf{R}_k' = \mathrm{E}\{\mathbf{v}_k'(\mathbf{v}_k')^{\mathrm{T}}\} &= \mathrm{E}\{(\mathbf{U}_k^{-1}\mathbf{v}_k)(\mathbf{U}_k^{-1}\mathbf{v}_k)^{\mathrm{T}}\} \\ &= \mathrm{E}\{\mathbf{U}_k^{-1}\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}(\mathbf{U}_k^{-1})^{\mathrm{T}}\} \\ &= \mathbf{U}_k^{-1}\mathrm{E}\{\mathbf{v}_k\mathbf{v}_k^{\mathrm{T}}\}(\mathbf{U}_k^{-1})^{\mathrm{T}} \\ &= \mathbf{U}_k^{-1}\mathbf{R}_k(\mathbf{U}_k^{-1})^{\mathrm{T}} \\ &= \mathbf{U}_k^{-1}(\mathbf{U}_k\mathbf{D}_k\mathbf{U}_k^{\mathrm{T}})(\mathbf{U}_k^{-1})^{\mathrm{T}} \\ &= (\mathbf{U}_k^{-1}\mathbf{U}_k)\mathbf{D}_k(\mathbf{U}_k^{-1}\mathbf{U}_k)^{\mathrm{T}} \\ &= \mathbf{D}_k \end{aligned}$$

which is a diagonal.

The importance of numerical accuracy and stability should be emphasized. Computers will always make errors in limited finite word-length operations. When the effect of round-off is ignored the error accumulates and we may experience a major blunder in a critical mission.

List of Programs

Program	*Features*
(1) FLT_XY.CPP	Generates a flight trajectory of a passenger airliner, an ideal straight line path
(2) TRAJ_XY.CPP	Generates a flight trajectory of a passenger
(3) STATIST.H	airliner perturbed by Gaussian acceleration noise; a header file checks statistics of Gaussian noise
(4) MEA_RAZ.CPP	Generates the measurement data by a ground radar in range and azimuth
(5) KAL_XY.CPP	The Kalman filter tracks the airliner, header
(6) MATRIX.H	file for matrix operations
(7) ERR_RAZ.CPP	Computes the errors between the trajectory and predicted/estimated positions of the airliner
(8) ATC_RAZ.CPP	Generates a flight trajectory of the airliner for landing approach
(9) KAL_ATC.CPP	An ATC radar tracks an
(10) MATRIX.H	airliner for landing approach
(11) ATC_ERR.CPP	Computes the errors between the landing approach trajectory and the predicted/estimated positions by an ATC radar
(12) FLT_XYZ.CPP	Generates a "turn-dive-and-turn-climb"
(13) FLT_XYZ.H	trajectory of a typical fighter-bomber on a ground attack maneuver in CCS coordinates
(14) XYZ_RAE.CPP	Converts CCS coordinates to LOS coordinates for later use.
(15) KAL_MIG.CPP	An air defense radar tracks an airborne target
(16) MATRIX,H	with ± 3g maneuver; Kalman filter in CCS
(17) HPHR_INV.H	coordinate system; a header file for matrix inversion
(18) ERR_MIG.CPP	Computes the errors between the target position and the predicted/estimated positions in CCS coordinates
(19) KAL_F16.CPP	The same airborne target is tracked but in
(20) MATRIX.H	LOS coordinates instead
(21) HPHR_INV.H	
(22) ERR_F16.CPP	Computes errors in KAL_F16.CPP in LOS coordinates
(23) KAL_NMI.CPP	The same airborne target is tracked but an
(24) MATRIX.H	inversion-free algorithm used
(25) UDU_FCTR.H	No-matrix-inversion; A header file for $\mathbf{UDU}^T$ factorization
(26) ERR_NMI.CPP	Computes the errors by NMI algorith

References

[1] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *Trans. ASME, Journal of Basic Engineering*, Mar. 1960.

[2] Kalman, R. E. and R. S. Bucy, "New Results in Linear Filtering and Prediction," *Trans. ASME, Journal of Basic Engineering*, 1961.

[3] Athens, M., *Gradient, Matrix and Matrix Calculations,* Lincoln Lab., MIT, Lexington, Mass. Nov. 1965 (AD 624426).

[4] Bucy, R. S. and P. D. Joseph, *Filtering for Stochastic Process with Application to Guidance,* New York, N.Y.: Interscience Publishers, 1968.

[5] Jazwinski, A. W., *Stochastic Processes and Filtering Theory,* New York., N.Y.: Academic Press, 1970.

[6] Gelb, A. ed., *Applied Optimal Estimation,* Cambridge, MA: The MIT Press, 1974.

[7] Singer, R. A., "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," *IEEE Trans. AES*, Vol. 6, No. 4, Jul. 1970.

[8] Carlson, N. A. "Fast Triangulation Estimation of the Square-Root Filter," *AIAA Journal*, Vol. 11, No. 5, Sept. 1973.

[9] Kaminski, P. G., A. E. Bryson, and J. F. Shumidt, "Discrete Square-Root Filtering: A survey of Current Theory," *IEEE Trans AC*, Vol. 16, No. 6, Dec. 1971.

[10] Bierman, G. J. *Factorization Methods for Discrete Sequential Estimation,* New York, N.Y.: Academic Press, 1977.

Selected Bibliography

Golub, G. H. and C. F. Van Loan, *Matrix Computations, 3rd ed.*, Baltimore, Maryland: The John Hopkins University Press, 1996.

CHAPTER 9

Monte Carlo Method and Function Integration

9.1 Introduction

The term Monte Carlo method or Monte Carlo technique stirs excitement in every engineer and scientist. In this chapter we present rudimentary principles involved in the Monte Carlo method or technique. The Monte Carlo technique is a branch of experimental mathematics, and it is extending into wider applications in such fields as nuclear physics, molecular chemistry, population demographic study, hydrographic analysis, corporate business planning, product quality control, and political opinion survey.

During the infancy period of Monte Carlo development, engineers were introduced to the "hit-or-miss" method analogy, which was greeted with a lukewarm reception. As the development progressed, many researchers in many diversified fields have recognized the great utility of this new experimental mathematics. Hundreds of research papers have been published in science, engineering, economics, and political science.

This chapter covers the very minimum basics of the Monte Carlo method through a few examples for the benefit of beginners without rigorous mathematical proof.

The Monte Carlo method may be classified by a set of particular techniques adopted to solve a problem on hand:

1. Hit-or-miss method;
2. Ordered sample method;
3. Sample mean method;
4. Importance sampling method;
5. Correlated sampling method;
6. Control variate method;
7. Stratified sampling method;
8. Antithetic variate method;
9. Some others.

We shall limit our study to the first four. For other advanced methods, see [1–11]. We summarize the mathematic principle involved and the limitation of each method through examples.

9.2 Hit-or-Miss Method

Suppose we draw an irregular curve on a unit square as shown in Figure 9.1, and wonder how to compute the area under the curve. The curved line is difficult or impossible to express by a mathematic equation.

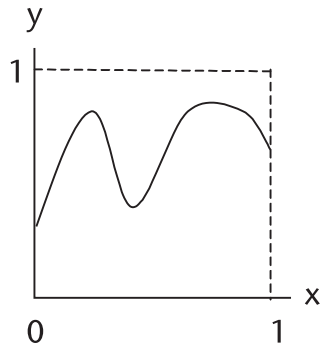


Figure 9.1 An irregular curve on unit square.

Let us be more mathematical in computing an area enclosed by curves. Two circles with different radii and centers have been drawn as shown in Figure 9.2.

The coordinates of the center o_1 and o_2 are given as well as two radii r_1 and r_2. Unlike the first example, we should be able to derive a mathematic expression for the area enclosed by the circles. However, it is very difficult to obtain an expression, or an extremely time-consuming task. (See Appendix 9A.)

The third example is a hypothetical situation where your company plans to market a new product. You know the production cost. You know the current bank interest rate. You would like to maximize the profit. Let us assume that the following data is available: the ordinate is the number of households, the abscissa the average household's income. If your product is a very luxurious item you would implement a higher profit margin and expect to sell fewer. On the other hand, if the product is a popular item you would put in place a lower price tag (a lower profit margin per item) but anticipate a larger sales volume. What would be an optimum price? (See Figure 9.3.)

Similar situations abound in our practical world. The three problems we depict in Figures 9.1–9.3 can be handled by the various methods mentioned above.

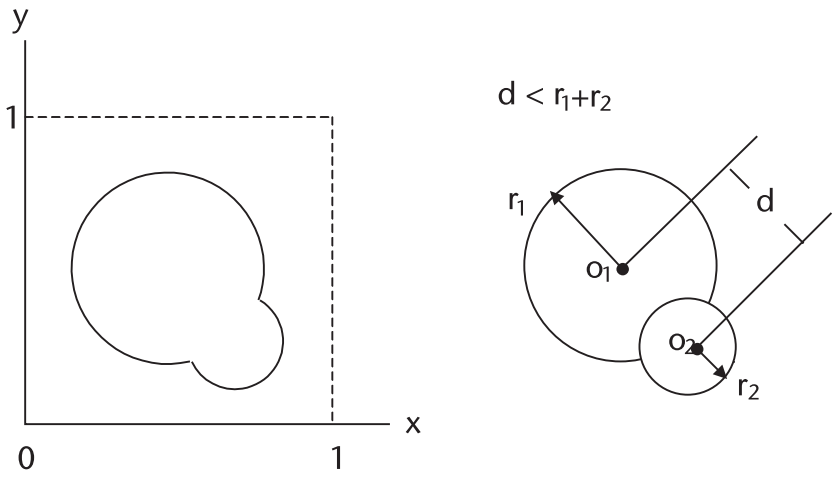


Figure 9.2 Two circles on a unit square.

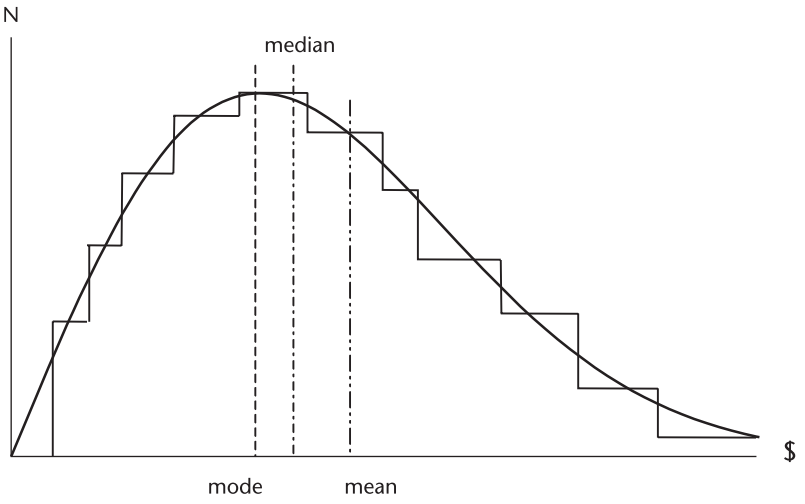


Figure 9.3 Household income versus number of households.

To be more mathematically tractable we choose a problem of computing (or estimating) the area under a unit Gaussian probability density function as shown in Figure 9.4. We have chosen this function because there is no closed-form expression to compute the area, even though we can find a table with nine significant digits.

We plan to compute, or using the more precise term, to estimate the area in the range $0 \le x \le 1$ by four different methods: the hit-or-miss, sample mean, ordered sample, and importance sampling methods.

We shall compare the results of the four methods, including the accuracy (the magnitude of error), ease or difficulty of algorithm (efficiency), and the number of samples required to obtain a statistically meaningful result (error bound and confidence level).

For the hit-or-miss method, we generate two sets of unit uniform random numbers $(x_0, x_1, x_2, \ldots, x_{N-1})$ and $(y_0, y_1, y_2, \ldots, y_{N-1})$, and pair them such that (x_0, y_0), (x_1, y_1), (x_2, y_2), $\ldots$, (x_{N-1}, y_{N-1}) represent the coordinate of points in the unit square. We shall have N points. We shall call a "hit" if (x_i, y_i) is on or under the Gaussian curve,

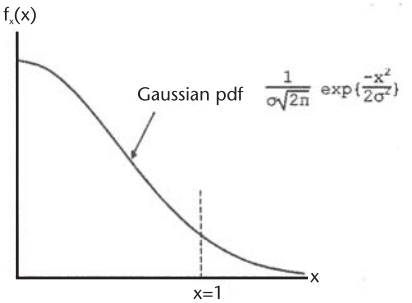


Figure 9.4 Gaussian probability density function.

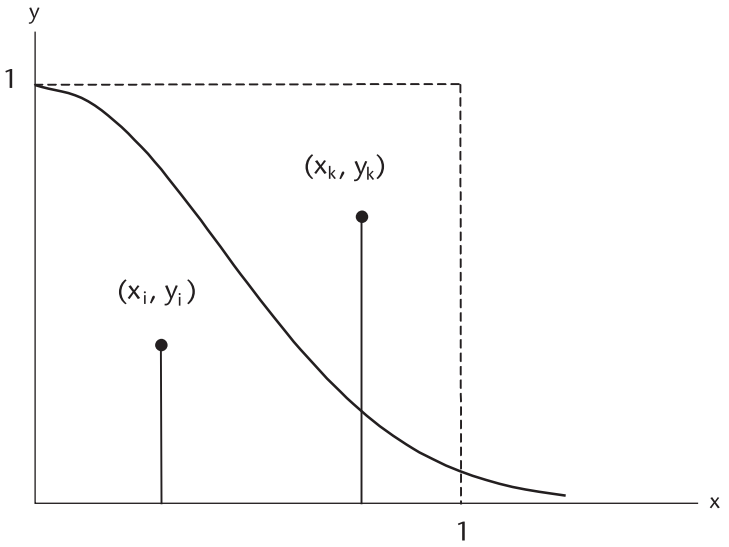


Figure 9.5 Hit-or-miss method on Gaussian pdf.

and call a "miss" if (x_k, y_k) is above the curve. (See Figure 9.5.) The area under the Gaussian probability density function (pdf), $0 \le x \le 1$, is estimated by

$$\text{area} = \frac{\text{number of hits}}{\text{total number of sample N}}$$

An estimate of the area is programmed in HIT_MISS.CPP, using 128 pairs of PRNs. The estimated area is 0.335938. The true area (or the answer given in the mathematic table) is 0.341345, an error of 0.005407 or 1.6%. Our intuition is that if we increase the number of samples the error will decrease. More about that later. The important key in the hit-or-miss method is that the random numbers be distributed uniformly in the range [1].

Digression to error function. The error function is defined in two different ways. Although they are essentially equivalent, the different numerical tables more often than not cause confusion. The error function is defined as follows:

$$1.\ \text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z \exp\{-t^2\}\, dt;$$

$$2.\ \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp\{-t^2/2\}\, dt.$$

Abramowitz and Stegun [4] have used the definition (1) and published an extensive numerical table. Other authors use the definition (2). Depending upon the problem on hand, one is more convenient than the other. We show how to convert the numerical tables given by two definitions.

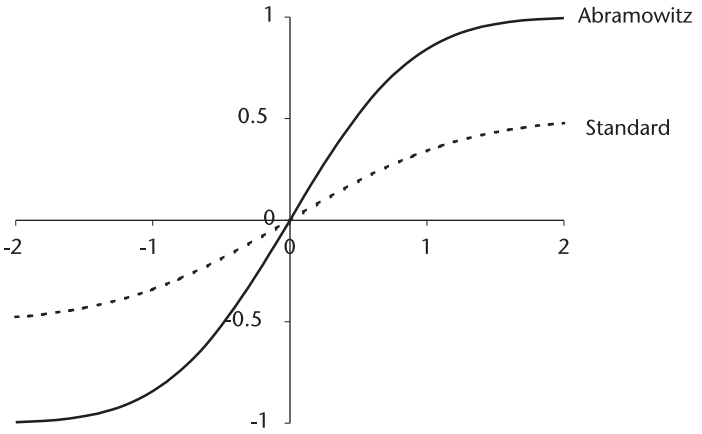


Figure 9.6 Two different definitions of error function.

$$\text{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_0^x \exp\{-x^2/2\}\, dx$$

$$\text{Let } x^2/2 = t^2, \quad x^2 = 2t^2, \quad x = \sqrt{2}\,t, \quad dx = \sqrt{2}\,dt$$

$$= \frac{1}{\sqrt{2\pi}} \int_0^{\sqrt{2}t} \exp\{-t^2\}\sqrt{2}\, dt$$

Substituting the definition of (1),

$$= \frac{1}{\sqrt{2\pi}} \left[\frac{2}{\sqrt{\pi}} \int_0^{\sqrt{2}t} \exp\{-t^2\}\, dt \right] \sqrt{2}\, \frac{\sqrt{\pi}}{2}$$

$$= \frac{1}{2} [\text{Abramowitz \& Stegun's table}]$$

The error functions defined by (1) and (2) are programmed in ERR_ABST. CPP and ERF_GAU.CPP, and shown in Figure 9.6.

9.3 Ordered Sample Method [3]

An integral can be estimated as the sum of trapezoids. A unit uniform random vector $\mathbf{U} = (u_0, u_1, u_2, \ldots, u_{N-1})$ is rearranged in an ascending order,

$$\mathbf{U}_{\text{ordered}} = u_{(0)} \le u_{(1)} \le u_{(2)} \le \ldots . \le u_{(N-1)}$$

and the ordinates are computed by

$$g[u_{(0)}],\ g[u_{(1)}], g[u_{(2)}], \ldots, g[u_{(N-1)}]$$

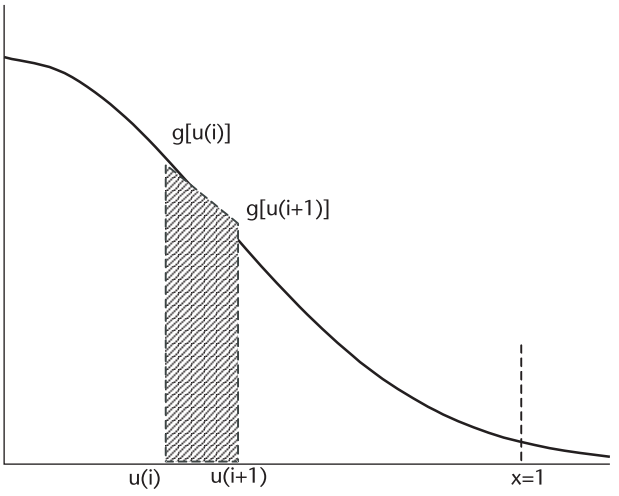


Figure 9.7 Area as sum of trapezoids.

An estimate of the integral is obtained by the sum of trapezoids as shown in Figure 9.7.

$$I = \tfrac{1}{2} \sum_{i=0}^{N-1} [g(u_i)+g(u_{i+1})]\ [u_{(i+1)}-u_{(i)}] \tag{9.1}$$

Equation (9.1) is programmed in ORDERED.CPP. The estimate is 0.338227. The correct answer is 0.341345, an error of 0.003118, or 0.91%.

9.4 Sample Mean Method

In the hit-or-miss method and the ordered sample method we have applied geometric interpretation to the integral. For the sample mean method we apply the probability concept to the integral. We shall interpret an integral as an expectation (or mean) operation as

$$I = \int_a^b g(x)\ dx = \int_a^b x f_x(x)\ dx = E\{x\} \tag{9.2}$$

We rewrite (9.2) as follows:

$$I = \int_a^b g(x)\ dx = \int_a^b \frac{g_x(x)}{f_x(x)} f_x(x)\, dx = E\left\{\frac{g_x(x)}{f_x(x)}\right\} \tag{9.3}$$

where the random variable x has a probability density function $f_x(x)$ and probability distribution function $F_x(x)$. They meet the axiom of the probability principle.

$$f_x(x) \geq 0 \tag{9.4}$$

$$\int_{-\infty}^{\infty} f_x(x)\,dx = F_x(x) = 1 \tag{9.5}$$

For the sample mean method we choose $f_x(x)=1$, a unit uniform probability density function. Then (9.3) becomes

$$I = E\{g_x(x)\} = \frac{1}{N}\frac{1}{\sqrt{2\pi}}\sum_{i=0}^{N-1}\exp\left\{\frac{-x_i^2}{2}\right\} \tag{9.6}$$

Equation (9.6) is programmed in SAMPMEAN.CPP. The integral I is 0.340730. The true value is 0.341345, and the error is 0.000615, or 0.18%.

9.5 Importance Sampling Method

For the importance sampling method we would like to entertain that $f_x(x)$ is different from a unit uniform probability density function as we have had in the sample mean method. The motivating idea is to concentrate the distribution of the random variables x in the close vicinity of $g_x(x)$ rather than spreading it all over the range. We repeat (9.3) below with ξ as an unbiased estimator of I.

$$I = \int_a^b g(x)\ dx = \int_a^b \frac{g_x(x)}{f_x(x)} f_x(x)\,dx = E\left\{\frac{g_x(x)}{f_x(x)}\right\} \approx \xi \tag{9.7}$$

The probability density function $f_x(x)$ is no longer a unit uniform function but instead is an arbitrary pdf that meets the axiom of (9.4) and (9.5). The question is what should $f_x(x)$ be in order to improve the accuracy of the integral.

The variance of (9.7) is given, by definition,

$$\begin{aligned} \text{var}\{\xi\} &= E\{\xi^2\} - E^2\{\xi\} \\ &= \int \frac{g^2(x)}{f^2(x)} f(x)\,dx - E^2\left\{\frac{g(x)}{f(x)}\right\} = \int \frac{g^2(x)}{f(x)}dx - I^2 \end{aligned} \tag{9.8}$$

We shall prove that var$\{\xi\}$ would be minimized when an arbitrary pdf $f_x(x)$ is proportional to the absolute magnitude of $g_x(x)$.

The Cauchy-Schwarz inequality states that,

$$\left[\int |u(x)v(x)|dx\right]^2 \le \int u^2(x)dx \cdot \int v^2(x)dx$$

Let

$$u(x) = \frac{g(x)}{\sqrt{f(x)}}, \quad v(x) = \sqrt{f(x)}$$

Then,

$$\left[\int |g(x)|dx\right]^2 \le \int \frac{g^2(x)}{f(x)}\,dx \cdot \int f(x)\,dx = \int \frac{g^2(x)}{f(x)}dx \tag{9.9}$$

From (9.8) and (9.9) it follows that

$$\text{var}\{\xi\} \le \left[\int |\,g(x)\,|\,dx\right]^2 - I^2 \tag{9.10}$$

We shall prove that the lower bound of var{ξ} of (9.10) is obtained when an arbitrary pdf f(x) is proportional to the magnitude of g(x).

$$f(x) = c\,|\,g(x)\,|$$

The extremum (in this case the minimum) of (9.10) can be found through the Lagrange operator:

$$L[\text{var}\{\xi\}] = \int \frac{g^2(x)}{f(x)}\,dx - \lambda^2\left[\int f(x)\,dx\right]^2 \tag{9.11}$$

The Lagrange multiplier λ is obtained when a partial derivative of (9.11) with respect to an arbitrary f(x) is taken and equated to zero:

$$\frac{\partial L}{\partial[f(x)]} = \frac{-g^2(x)}{f^2(x)} - \lambda^2 = 0$$

Thus,

$$\frac{g^2(x)}{f^2(x)} = -\lambda^2, \quad f(x) = \frac{1}{\lambda}|g(x)|, \quad \text{and} \quad f(x) = c\,|g(x)|$$

$$\int_a^b f(x)\,dx = c\int_a^b |g(x)|\,dx = 1, \quad c = \left[\int_a^b |g(x)|\,dx\right]^{-1}$$

We have proved that var{ξ} will be minimized when an arbitrary pdf f(x) is chosen to be proportional to the magnitude of g(x).

A disturbing question arises: the original function g(x) is assumed to be so complicated to defy an integration, or there is no closed-form analytic expression for integration (in our example of Gaussian pdf), and so is the proportional pdf f(x).

Don't despair, "not all is lost," says [7]. Hammersley and Handscomb proposed that the strict proportionality can be relaxed to similarity. In other words the pdf f(x) is similar in shape to g(x) yet we take a benefit of reduced variance.

For example, we take a cosine function as similar to Gaussian over the range of integration [0,1] as shown in Figure 9.8. The proposed f(x), being a probability density function, must meet the axiom of (9.4) and (9.5).

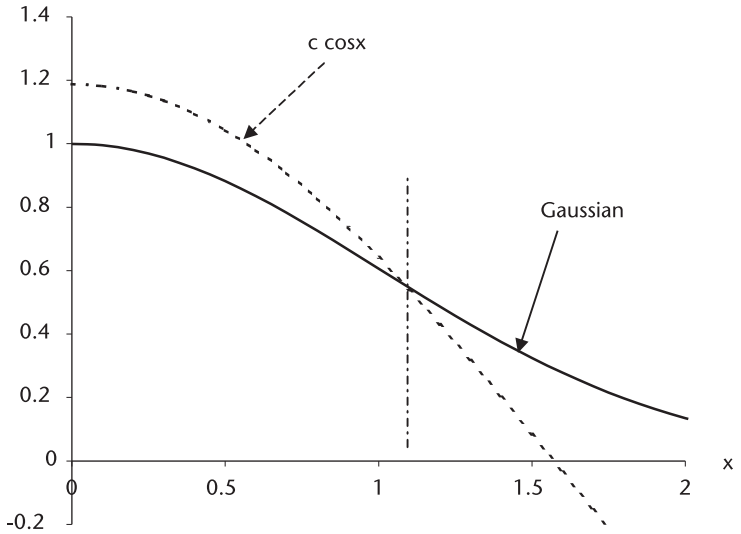


Figure 9.8 Similarity function, $\frac{1}{\sqrt{2}} \exp\left\{\frac{-x^2}{2}\right\} \sim c \cos x$

$$c\int_0^1 \cos x\, dx = c \sin(1) = 1, \quad c = \frac{1}{\sin(1)}.$$

Therefore,

$$I = \frac{1}{\sqrt{2\pi}}\int_0^1 \exp\left\{\frac{-x^2}{2}\right\} dx = \frac{1}{\sqrt{2\pi}}\int_0^1 \frac{\exp\left\{\frac{-x^2}{2}\right\}}{\left[\frac{\cos x}{\sin(1)}\right]} dx$$

$$= \frac{1}{N}\frac{1}{\sqrt{2\pi}} \sin(1) \sum_{i=0}^{N-1} \left[\exp\left\{\frac{-x_i^2}{2}\right\} / \cos x_i\right] \tag{9.12}$$

Equation (9.12) is programmed in IMPRTANC.CPP. The integral is 0.341507. The true value is 0.341345, an error of 0.000162, or 0.05%.

9.6 Observations and Remarks

1. Results of four different methods are tabulated below for comparison. The number of samples used is 128 for all cases.

Method	*Integral*	*Error*	*Percent*
Hit-or-Miss	0.335938	0.005408	1.6%
Ordered Sample	0.338227	0.003818	0.9%
Sample Mean	0.340730	0.000615	0.2%
Importance Sample	0.341507	0.000162	0.05%

2. When the integrand f(x) cannot be expressed in a mathematical equation, the hit-or-miss method is the only way available. The random variables must be uniformly distributed over the range.
3. When the integrand is mathematically expressible but the integration does not have a closed-form analytic expression, the other three methods can be employed.
4. The ordered sample method is a straightforward summation of trapezoids.
5. The sample mean method is a relatively simple computation of an expectation operation, the mean value.
6. The importance sampling method yields the best estimate of the integral among four methods, but with a heavy computation load. Through the importance sampling method we have learned the importance of variance reduction. All the methods (5–9) mentioned in the introduction are various strategies to reduce the variance efficiently.
7. The central questions in applying Monte Carlo method are:

    a. How many observations (or samples) would be adequate to ensure a statistically meaningful accuracy, and
    b. Conversely, given N observations how accurate is the estimate and what level of confidence.

    When the cost of collecting the samples is inexpensive (i.e., the generation of PRN), we should increase the number of samples N as large as we'd like. On the other hand, when the cost of collecting the samples is prohibitively expensive or the time to collect is limited as in a political opinion survey, we should like to economize on the number of samples.

Often we encounter a paucity of sampled data as in the flood data of the last 50 years or wind-gust data of the last 100 years. Under these constrained conditions we desire an estimated result with a certain accuracy (reliability) and an acceptable level of confidence.

We shall answer (7a, b) in the next section through an example, the Q-function.

9.7 Probability of False Alarm, Exponential Probability Density Function

Applications of the importance sampling method are numerous. We take up the case of computing the false-alarm rate in a radar system where the tail of the probability density function of noise or clutter has to be examined. When narrowband Gaussian noise passes through a square-law detector, the output noise is exponentially distributed (see Chapter 7). An example of the Q-function or the false-alarm probability is shown in Figure 9.9.

The problem is the reverse of Section 9.5. Here we are given the area Q, the false-alarm probability, and we would like to estimate the threshold. The threshold is analytically obtainable, as shown below, however, for a moment we pretend we

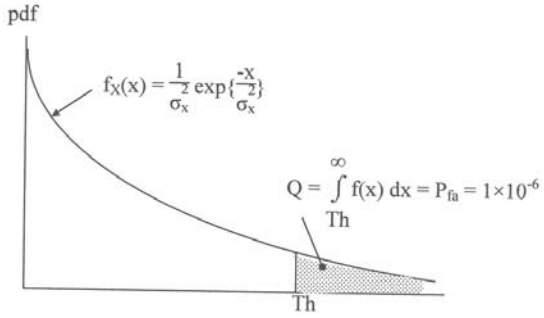


Figure 9.9 Q-function, the probability of false alarm.

do not know the analytic solution, and we resort to the Monte Carlo technique to estimate the threshold.

$$P_{fa} = 1.0E-6 = \int_{Th}^{\infty} \frac{1}{\sigma^2} \exp\left\{\frac{-x}{\sigma^2}\right\} dx = \exp\left\{\frac{-Th}{\sigma^2}\right\} \tag{9.13}$$

and,

$$\frac{Th}{\sigma^2} = 13.8155$$

Imagine we have employed the hit-or-miss method. If we distribute one million random samples, statistically one point may fall in the Q area, if at all. If we distribute 10 million random points, 10 points (maybe nine or 11 points) may fall in the shaded area. With 10 million sample points we achieve accuracy of 10%!

The tremendously large number of samples required and the poor estimate would discourage us from using the hit-or-miss method. Rescue comes from an application of the Importance Sampling method [10, 11]. We shall find a similarity function to the exponential pdf and use the technique of variance reduction in order to reduce the number of samples and establish a bound on the confidence level. (See Figure 9.10.)

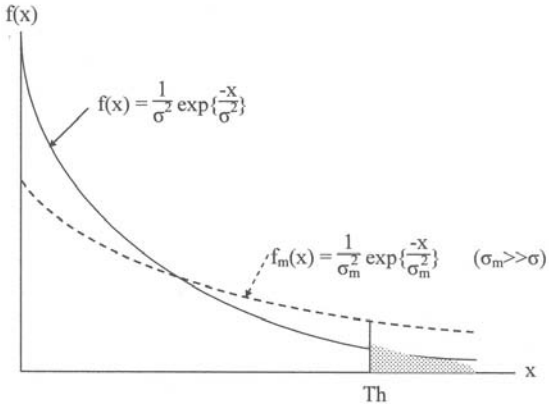


Figure 9.10 Exponential pdf and a similarity function.

The similarity function chosen is another exponential with a different variance, noise power σ_m^2, $\sigma_m^2 > \sigma^2$. A weighting function w(x) is introduced.

$$w(x) = \frac{f(x)}{f_m(x)} = \left(\frac{\sigma_m}{\sigma}\right)^2 \exp\left\{-\left[\frac{1}{\sigma^2} - \frac{1}{\sigma_m^2}\right] x\right\} \tag{9.14}$$

We digress to introduce the concept of the coefficient of dispersion (CD). The CD is defined as the ratio of the standard deviation to the first moment, the expectation, or mean. The CD is a measure of how certain an estimate is. The smaller the CD the higher the confidence that the estimate is closer to the true value. Figure 9.11 illustrates the concept of the CD.

We derive the mean and the second moment of the exponential pdf in terms of the weighting function w(x) and the similarity function $f_m(x)$.

$$E\{f(x)\} = \int_{Th}^{\infty} w(x) f_m(x)\, dx = \int_{Th}^{\infty} f(x)\, dx = \exp\left\{\frac{-Th}{\sigma^2}\right\} \tag{9.15}$$

$$\begin{aligned} E\{f^2(x)\} &= \int_{Th}^{\infty} w^2(x) f_m(x)\, dx = \int_{Th}^{\infty} \frac{f^2(x)}{f_m(x)}\, dx \\ &= \left(\frac{\sigma_m}{\sigma}\right)^2 \frac{1}{[2 - (\frac{\sigma}{\sigma_m})^2]} \exp\left\{-\left[\frac{2}{\sigma^2} - \frac{1}{\sigma_m^2}\right] Th\right\} \\ &\approx \left(\frac{\sigma_m}{\sigma}\right)^2 \frac{1}{2} \exp\left\{-\left[\frac{2}{\sigma^2} - \frac{1}{\sigma_m^2}\right] Th\right\} \end{aligned} \tag{9.16}$$

$$(\frac{\sigma}{\sigma_m})^2 <<< 1$$

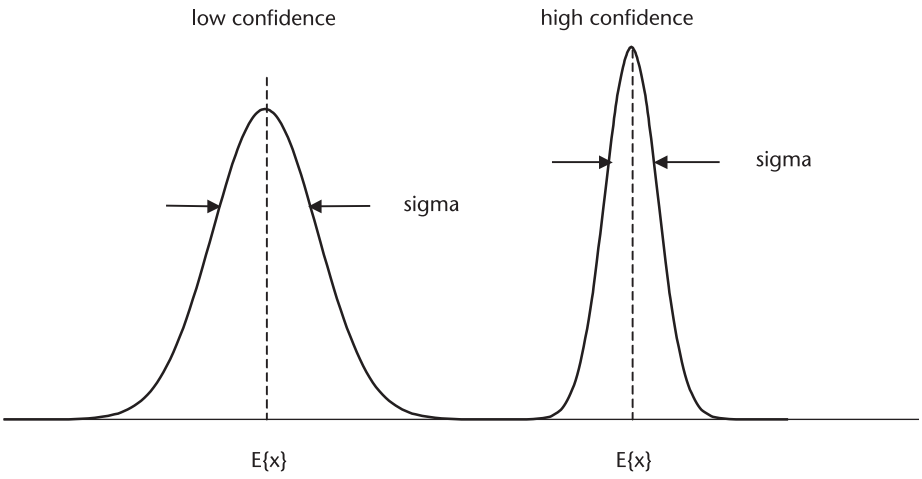


Figure 9.11 Mean, standard of deviation, and the CD.

The variance of f(x) is given by definition

$$\sigma_f^2 \equiv E\{f^2(x)\} - E^2\{f(x)\}$$

$$= \frac{1}{2}\left(\frac{\sigma_m}{\sigma}\right)^2 \exp\left\{-\left[\frac{2}{\sigma^2} - \frac{1}{\sigma_m^2}\right]\mathrm{Th}\right\} - \exp\left\{\frac{-2\mathrm{Th}}{\sigma^2}\right\}$$

Thus the CD is obtained as a ratio of

$$\mathrm{CD} = \frac{\sigma_f}{E\{x\}} = \frac{\sigma_x^2}{E^2\{x\}} = \frac{1}{2}\left(\frac{\sigma_m}{\sigma}\right)^2 \exp\left\{\frac{\mathrm{Th}}{\sigma^2}\right\} - 1 \tag{9.17}$$

The value of σ_m that would minimize CD is given by taking the partial derivative of (9.17) with respect to σ_m and setting the result equal to zero.

$$\frac{\partial}{\partial\sigma_m}\left[\frac{\sigma_x}{E\{x\}}\right] = \exp\left\{\frac{\mathrm{Th}}{\sigma^2}\right\}\left[\frac{\sigma_m}{\sigma^2} - \frac{\mathrm{Th}}{\sigma^2\sigma_m}\right] = 0 \tag{9.18}$$

therefore, $\sigma_m^2 = \mathrm{Th}$.

With the optimized value of $\sigma_m^2 = \mathrm{Th}$, the CD is obtained:

$$\mathrm{CD} = \frac{\sigma_x^2}{E^2\{x\}}\bigg|_{\sigma_m^2=\mathrm{Th}} = \frac{1}{2}\left(\frac{\mathrm{Th}}{\sigma^2}\right)\exp\{1\} - 1 \tag{9.19}$$

Substituting $\left(\frac{\mathrm{Th}}{\sigma^2}\right) = 13.8155$ from (9.13) into (9.19), we obtain the minimized CD.

$$\mathrm{CD} = \frac{\sigma_x^2}{E^2\{x\}} = \frac{1}{N}\left[\frac{1}{2}(13.8155\ \exp\{1\}\ - 1\right] = \frac{17.78}{N}$$

For a CD of 1%, the number of samples required is

$$\frac{1}{100} = \frac{17.78}{N}, \qquad N \approx 1{,}778$$

For a CD of 0.5%, the number of sample required is

$$\frac{5}{1000} = \frac{17.78}{N}, \qquad N \approx 3{,}556$$

For a CD of 0.1%,

$$\frac{1}{1000} = \frac{17.78}{N}, \qquad N \approx 17{,}780$$

The optimum choice of σ_m^2 for the similarity function, a modified exponential pdf, that minimizes the CD has reduced the number of samples by several orders of magnitude compared with the hit-or-miss method.

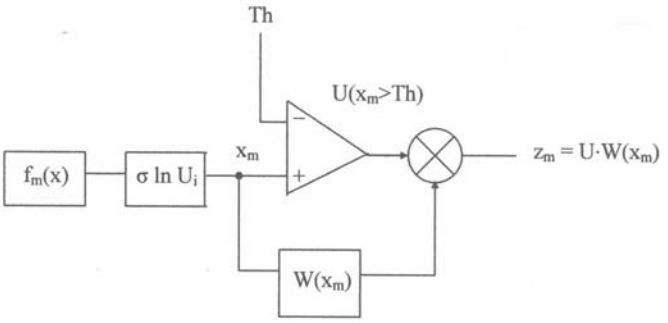


Figure 9.12 Flow diagram of Pfa_EXP.CPP.

The threshold Th for the false-alarm probability P_{fa} computed by the importance sampling method is programmed in Pfa_EXP.CPP with replication N=1,000, and an algorithm flow diagram is shown in Figure 9.12.

$$U_i = \text{unit uniform random variables}, i = 0, 1, 2, \ldots, N-1$$

$$f(x) = \frac{1}{\sigma^2} \exp\left\{\frac{-x}{\sigma^2}\right\}$$

$$f_m(x) = \frac{1}{\sigma_m^2} \exp\left\{\frac{-x}{\sigma_m^2}\right\}$$

$$w(x) = f(x)/f_m(x) = \left(\frac{\sigma_m}{\sigma}\right)^2 \exp\left\{-\left[\frac{1}{\sigma^2} - \frac{1}{\sigma_m^2}\right] x\right\}$$

$$w(x_m) = \exp\left\{-\left[\frac{1}{\sigma^2} - \frac{1}{\sigma_m^2}\right] x_m\right\}$$

$$E\{P_{fa}\} = \hat{P}_{fa} = \frac{1}{N}\sum_{i=1}^{N} z_m$$

The result of Pfa_EXP.CPP is shown in Figure 9.13. Although σ_m is optimized for Q = Pfa = 1.0E-6, Figure 9.13 shows no discernible error throughout the range of threshold. (Try N=500.)

9.8 Probability of False Alarm, Gaussian Probability Density Function

For the second example we use the Gaussian probability density function to compute the threshold for the Q-function, P_{fa}=1.0E-6, by the importance sampling

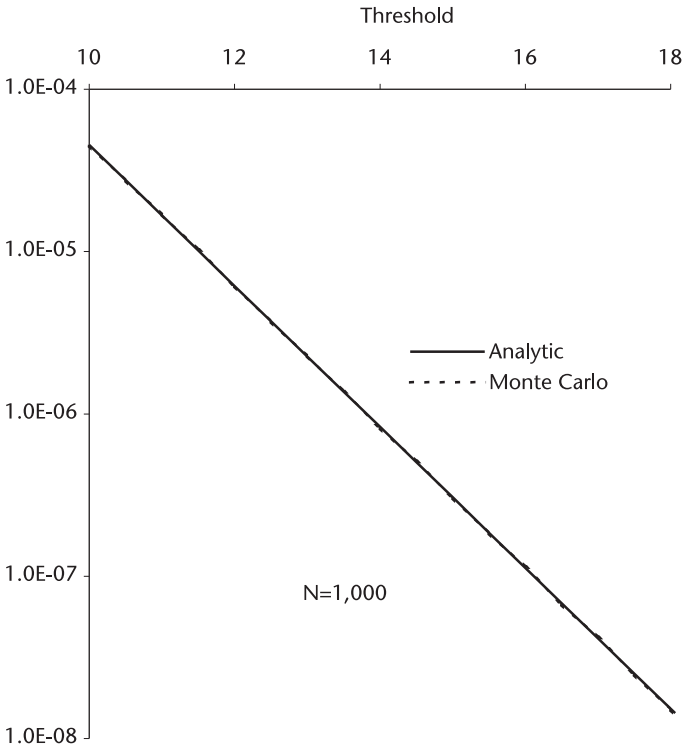


Figure 9.13 Threshold level versus false-alarm probability, exponential probability density function.

method. We derive an expression for the CD, minimize the dispersion, and establish the number of Monte Carlo replications required. (See Figure 9.14.)

Gaussian probability density function with zero mean and variance σ^2 is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-x^2}{2\sigma^2}\right\}$$

We modify f(x) by increasing the variance. This is the similarity function, not a strict proportional function that we have discussed.

$$f_m(x) = \frac{1}{\sigma_m\sqrt{2\pi}} \exp\left\{\frac{-x^2}{2\sigma_m^2}\right\}, \quad \sigma_m > \sigma$$

The procedure to obtain an expression for the CD is identical to that for an exponential pdf. First we derive the first moment (the expectation, or the mean) and the second moment of f(x) in terms of $f_m(x)$.

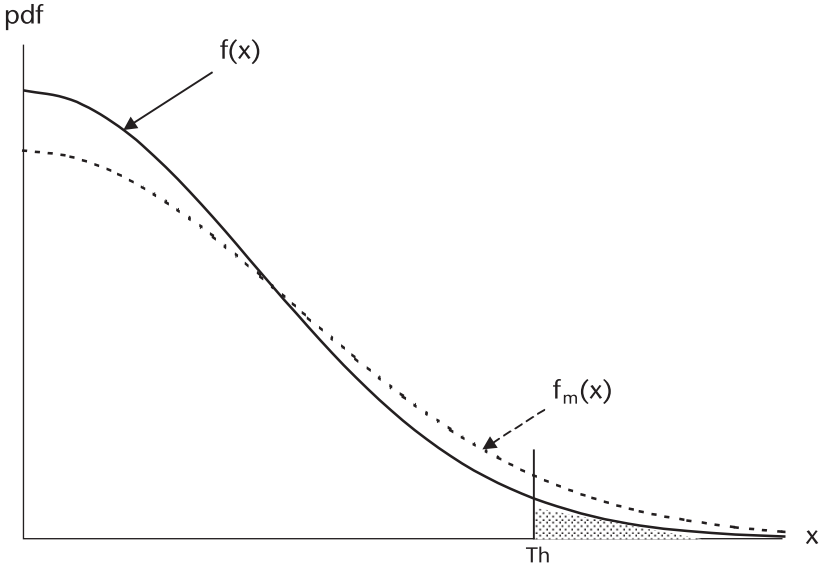


Figure 9.14 Gaussian probability density function and similarity function.

The weighting function w(x) is given by

$$w(x) = \frac{f(x)}{f_m(x)} = \frac{\sigma_m}{\sigma} \exp\left\{ -\left[\frac{1}{\sigma^2} - \frac{1}{\sigma_m^2} \right] \frac{x^2}{2} \right\}$$

The first moment of f(x) in terms of $f_m(x)$ is

$$E\{f(x)\} = \int_{Th}^{\infty} w(x)\, f_m(x)\, dx = \int_{Th}^{\infty} f(x)\, dx = \frac{1}{2} \text{erfc}\left(\frac{Th}{\sigma\sqrt{2}} \right)$$

where erfc(·) is the complementary error function. The second moment is given by

$$\begin{aligned} E\{f^2(x)\} &= \int_{Th}^{\infty} w^2(x) f_m(x)\, dx = \int_{Th}^{\infty} \frac{f^2(x)}{f_m(x)}\, dx \\ &= \frac{\sigma_m}{\sigma^2\sqrt{2\pi}} \int_{Th}^{\infty} \exp\left\{ -\left[\frac{2}{\sigma^2} - \frac{1}{\sigma_m^2} \right] \frac{x^2}{2} \right\} \\ &= \frac{1}{2} \frac{\sigma_m}{\sigma} \frac{1}{\left[2 - \frac{\sigma^2}{\sigma_m^2} \right]} \text{erfc}\left[\frac{Th}{\sigma\sqrt{2}} \left(2 - \frac{\sigma^2}{\sigma_m^2} \right)^{1/2} \right] \end{aligned}$$

The variance is by definition

$$\begin{aligned}\text{var}\{f(x)\} &\equiv E\{f^2(x)\} - E^{,2}\{f(x)\} \\ &= \frac{1}{2}\frac{\sigma_m}{\sigma}\frac{1}{\left[2-\frac{\sigma^2}{\sigma_m^2}\right]}\text{erfc}\left[\frac{Th}{\sigma\sqrt{2}}\left(2-\frac{\sigma^2}{\sigma_m^2}\right)^{1/2}\right] - \frac{1}{4}\text{erfc}\left(\frac{Th}{\sigma\sqrt{2}}\right)\end{aligned}$$

Finally, the CD is obtained as the ratio

$$\begin{aligned}CD &\equiv \frac{STD\{f(x)\}}{E\{f(x)\}} = \frac{\text{var}\{f(x)\}}{E^2\{f(x)\}} \\ &= \frac{2\sigma_m}{\sigma}\frac{1}{\left[2-\frac{\sigma^2}{\sigma_m^2}\right]}\text{erfc}\left[\frac{Th}{\sigma\sqrt{2}}\left(2-\frac{\sigma 2}{\sigma_m^2}\right)^{1/2}\right]/\text{erfc}^2\left(\frac{Th}{\sigma\sqrt{2}}\right)\end{aligned} \tag{9.20}$$

What should the threshold Th be to yield P_{fa}=1.0E−6?

$$Pfa = 1.0E-6 = \frac{1}{\sigma\sqrt{2\pi}}\int_{Th}^{\infty}\exp\left\{\frac{-x^2}{2\sigma^2}\right\}dx = \frac{1}{2}\text{erfc}\left(\frac{Th}{\sigma\sqrt{2}}\right)$$

We have written a program Q_THRES.CPP and found that the threshold is approximately 4.75. Readers may find a numerical table for the complementary error function.

By injecting Th ≈ 4.75 into (9.20), we plan to minimize the CD by finding an optimum value for σ_m. For an exponential pdf we have resorted to a partial derivative (maximum likelihood principle). For a Gaussian pdf, a partial derivative of (9.20) with respect of σ_m is very lengthy and complicated; therefore, a numerical iterative search is appropriate. An iterative numerical search is written in CD_GAU.CPP. The result is shown in Figure 9.15.

The CD reaches the minimum of 49.7 when

$$4.8<\sigma_m<4.9$$

with a relatively broad trough. Thus, the relationship between the CD and the number of Monte Carlo replications is as follows.

$$CD = \frac{STD\{f(x)\}}{E\{f(x)\}} = \frac{\text{var}\{f(x)\}}{E^2\{f(x)\}} = \frac{49.7}{N}$$

The number of Monte Carlo simulation samples required N and the CD is tabulated below.

N	$CD = \frac{\sigma}{E\{x\}}$
1,000	5.0%
5,000	1.0%
10,000	0.5%

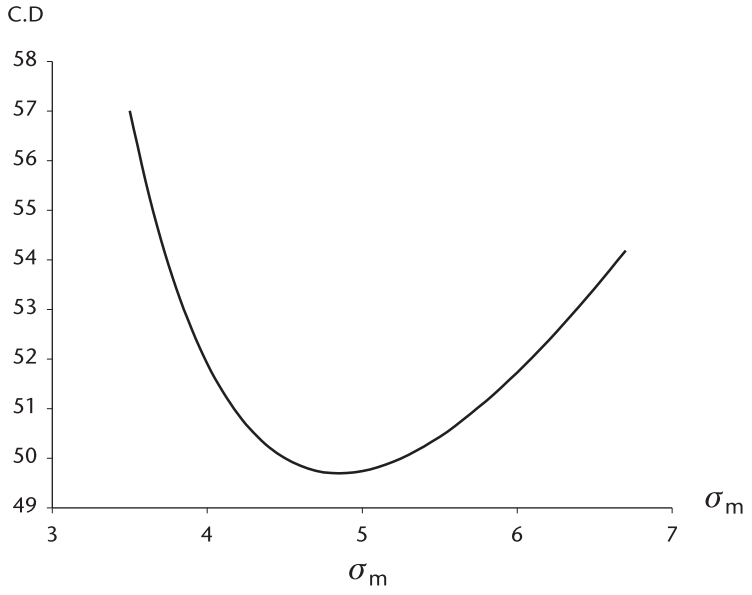


Figure 9.15 CD versus σ_m.

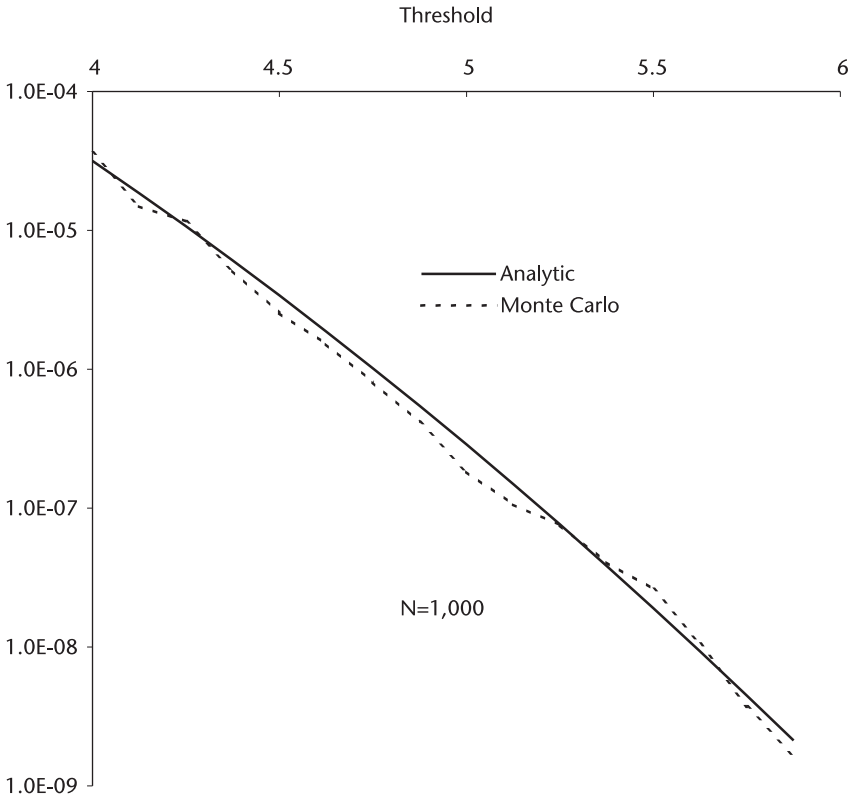


Figure 9.16 Threshold level versus false-alarm probability Gaussian probability density function.

The number of sample replications for the importance sampling method is three or four magnitudes less than the hit-or-miss method. The probability of false-alarm and the threshold is programmed in PFA_GAU.CPP, and the results are shown in Figure 9.16.

9.9 Integration of Functions

We have mentioned that the Monte Carlo method is a branch of experimental mathematics and that the experiment involves numerical integration of an unmanageably complex integrand. In the subsequent sections we analyze and program a few classic numerical integrations so that we learn when to employ the Monte Carlo technique.

The faint glow of glamour associated with the Monte Carlo goes out the window, as it should unless the problem on hand is extraordinary.

Numerical integration, sometimes called quadrature, is an obvious technique of summing the values of integrands with a sequence of abscissas in the given range. We have seen an example of quadrature in Section 9.3, the ordered sample method of summing the trapezoids. The quadrature approach, an accumulation of small slivers of areas works when the following are true:

1. The integrand does not have a singularity in a finite range of integration;
2. The integration range does not include positive or negative infinity.

There are some exceptions, however. We shall analyze and program the classic techniques of integration by the trapezoidal rule, Simpson's rule or an extended Simpson's rule. These are the workhorses when the integrand is moderately smooth and the sequence of abscissas is equally spaced. Next we shall program the Gaussian quadrature when the abscissas are not equally spaced in order to achieve integration of a higher order. We program a Gaussian-hyphenated-polynomial quadrature such as Gauss-Legendre, Gauss-Laguerre, Gauss-Chebyshev, and Gauss-Hermite. We have added simple cases of two- and three-dimensional integration.

9.9.1 Trapezoidal Rule

The ordinates of the interand are computed at a sequence of equally spaced abscissa to form trapezoids. The areas of the trapezoids are summed. The two end points of integration must be finite, as are the ordinates of the integrand; the integrand should not have singularity in the range.

$$\begin{aligned} I &= \int_a^b f(x)\, dx = h[{}^1\!/_2\, f_1 + f_2 + f_3 + \ldots + f_{N-1} + {}^1\!/_2\, f_N] \\ &= \frac{h}{2}[f_1 + 2f_2 + 2f_3 + \ldots 2f_{N-1} + f_N] \end{aligned} \tag{9.21}$$

where h is the width of equally spaced abscissa.

Equation (9.21) is programmed in TRAPZOID.CPP; f(x) is unit Gaussian probability density function with a=0 and b=1, N=128. The integral is 0.339438.

There is nothing to be excited about. We expect that the error will decrease as N is increased.

9.9.2 Simpson's Rule and Extended Simpson's Rule

The extended Simpson's rule is given by

$$\begin{aligned}\int_a^b f(x)\,dx &= h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{2}{3}f_2 + \ldots + \frac{2}{3}f_{N-2} + \frac{4}{3}f_{N-1} + \frac{1}{3}f_N\right] \\ &= \frac{h}{3}[f_0 + 4f_1 + 2f_2 + \ldots + 2f_{N-2} + 4f_{N-1} + f_N] \\ &= \frac{h}{3}\{f_0 + 4[f_1 + f_3 + f_5 + \ldots + f_{N-1}] \\ &+ 2[f_2 + f_4 + f_6 + \ldots + f_{N-2}] + f_N\}\end{aligned} \tag{9.22}$$

The 4/3 and 2/3 factor alternate throughout the interior of the summation. There is no mystery about the alternation. A simple three points Simpson's rule is applied to the successive nonoverlapping pairs of intervals.

$$\int_a^b f(x)\,dx = h\left[\frac{1}{3}f_0 + \frac{4}{3}f_1 + \frac{1}{3}f_2\right] \tag{9.23}$$

A program is written in SIMPSON.CPP for the unit Gaussian probability density function with a=0, b=1, and N=128. The integral is 0.341345, a surprisingly accurate answer matching the value in mathematical table with six significant digits. The error in the extended Simpson's rule is $1/N^4$. By reducing N we obtained the following results.

N	integral
64	0.34134474
32	0.34134474
16	0.34134477
8	0.34134540

The extended Simpson's rule works surprising well with small N when the integrand is relatively smooth. Simpson's three-point integration (9.23) can be extended to four-point integration.

$$\int_a^b f(x)\,dx = h\left[\frac{3}{8}f_0 + \frac{9}{8}f_1 + \frac{9}{8}f_2 + \frac{3}{8}f_3\right]$$

Equation (9.22) worked so well that we have added a new program SIMPSONX.CPP in the file. SIMPSON.CPP is tailored to the unit Gaussian probability density function. What if the integrand is different from the Gaussian pdf? We have to alter a few lines in the program, and this is a nuisance. We have inserted an arbitrary function name FUNC(x) in the new program for versatility. The declaration of FUNC(x) may be

$$f(x) = e^{-x}$$

$$f(x) = x \cdot e^{-x^2}$$

$$f(x) = \sqrt{1 + x^2}, \text{ etc.}$$

as long as the function f(x) is not singular in the range nor at the end points.

9.9.3 Gaussian Quadrature

So far the integrations have been restricted to a sequence of abscissas equally spaced in the range. If we remove the restriction, that is, densely spaced abscissas in a region where the integrand changes rapidly, sparsely where the integrand changes slowly, we can improve accuracy with an equal number of subintervals.

To introduce the idea of Gaussian quadrature we consider the following integration that involves certain polynomials:

$$\int_a^b w(x)P_n(x)P_m(x)dx = \begin{cases} 0 & n \neq m \\ h_n & n = m \end{cases}$$

A system of polynomials $P_n(x)$, in which degree of $P_n(x)$ is n, is called orthogonal in the interval [a,b] with respect to a weight function w(x), w(x)>0. When h_n is unity the polynomials are called othonormal:

$$\int_a^b w(x)P_n^2(x)\ dx = 1$$

There are many interesting orthogonal polynomials including Legendre, Chebyshev, Jacobi, Laguerre, and Hermite. When the limits [a,b] of integration are certain special cases, we select an appropriate polynomial; for example,

$$[a,b] = [-1,1] \quad \text{Legendre}$$

$$[a,b] = [-1,1] \quad \text{Chebyshev}$$

$$[a,b] = [0,\infty] \quad \text{Laguerre}$$

$$[a,b] = [-\infty,\infty] \quad \text{Hermite}$$

Integration of the following form can be evaluated by the Legendre polynomial.

$$\int_{-1}^{1} f(x)\ dx = \sum_{i=1}^{N} w_i f(x_i)$$

where w_i is the weight and x_i is the ith root of the Legendre polynomial of degree n. The integrand f(x) is evaluated at x_i.

Integration of the following form can be evaluated by Chebyshev polynomial.

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\ dx = \sum_{i=1}^{N} w_i\ f(x_i)$$

where w_i is the weight and x_i is the ith root of Chebyshev polynomial of degree n.

Integration of the following form can be obtained by Laguerre polynomials of degree n when the limits are [0, ∞].

$$\int_{0}^{\infty} e^{-x} f(x)\ dx = \sum_{i=1}^{N} w_i\ f(x_i)$$

where x_i is ith root of Laguerre polynomial of degree n, and w_i is the corresponding weight.

When the integrand is of the form of $\exp\{-x^2\} \cdot f(x)$ and the integration limits are [–∞, ∞], we employ a Hermite polynomial.

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)\ dx = \sum_{i=1}^{N} w_i\ f(x_i)$$

The key to a successful integration is how expeditiously we find the roots of the selected polynomial and the corresponding weights. We demonstrate a few of these quadrature techniques by examples.

9.9.3.1 Gauss-Legendre Quadrature

When a polynomial is Legendre we call the integration Gauss-Legendre quadrature.

$$\int_{a}^{b} f(x)\ dx = \sum_{i=1}^{N} w_i\ f(x_i)$$

An arbitrary limit [a,b] can be transformed to [–1, 1] by using a simple linear transformation.

$$y_i = \left(\frac{b-a}{2}\right) x_i + \left(\frac{b+a}{2}\right) \tag{9.24a}$$

$$\int_{a}^{b} f(x)dx = \left(\frac{b-a}{2}\right) \sum_{i=1}^{N} w_i\ f(x_i) \tag{9.24b}$$

Legendre polynomials have the following recurrence formula:

$$L_0(x) = 1$$

$$L_1(x) = x$$

$$L_2(x) = (1/2)\,(3x^2-1)$$

$$L_3(x) = (1/3)\,(5x^2-3x)$$

$$\begin{array}{cc} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{array}$$

$$L_n(x) = [(2n-1)/n]\; x \cdot L_{n-1}(x) - [(n-1)/n]\; L_{n-2}(x)$$

The derivatives and the weights are given by [11],

$$L'_n(x) = \frac{n[x\,L_n(x) - L_{n-1}(x)]}{(x^2-1)}, \quad w_i = \frac{2}{(1-x_i^2)[L'_n(x_i)]^2}$$

Figure (9.17) shows Legendre polynomials up to degree four. The weights are computed in LEGENDRE.CPP, up to degree ten, shown in Figure 9.18. Note the unequally spaced x[i], not like Simpsons & trapezoidal method.

[i]	x[i]	w[i]
1	0.013047	0.033336
2	0.067468	0.074726
3	0.160295	0.109543
4	0.283302	0.134633
5	0.425563	0.147762
6	0.574437	0.147762
7	0.716698	0.134633
8	0.839705	0.109543
9	0.932532	0.074726
10	0.986953	0.033336

Using the abscissas and weights obtained above, the integral of the unit Gaussian pdf with limits [0.0, 1.0] is computed in GAUS_LEG.CPP.

$$\frac{1}{\sqrt{2\pi}} \int_0^1 \exp\left\{\frac{-x^2}{2}\right\}\, dx = \frac{1}{\sqrt{2\pi}} \sum_{i=0}^{10} w_i\, f(x_i)$$

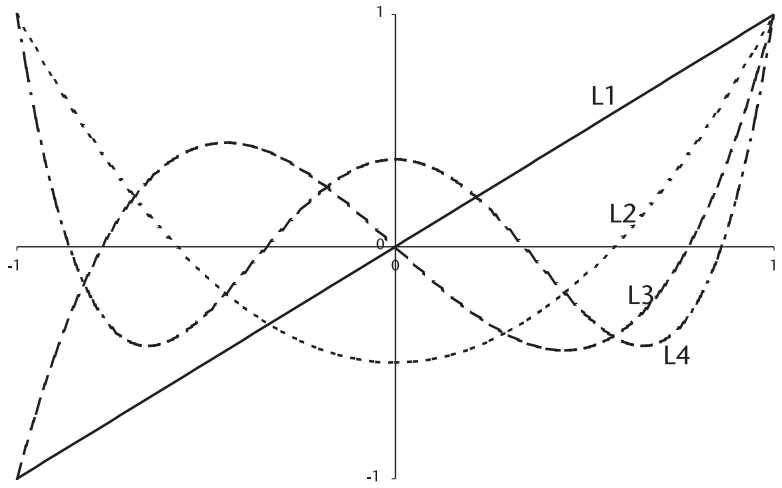


Figure 9.17 Legendre polynomials.

The answer is surprisingly accurate with 20 abscissas and weights.

When a change in integration limits is needed (i.e., from [0, 1] to [0, 3]) we use the transformation formula of (9.24). As an example the following integration is demonstrated in GAU_LEGX.CPP.

$$\frac{1}{\sqrt{2\pi}} \int_0^3 \exp\left\{\frac{-x^2}{2}\right\} dx$$

Check the result with a mathematic table. This is one-half of "three-sigma" we hear about often in quality control.

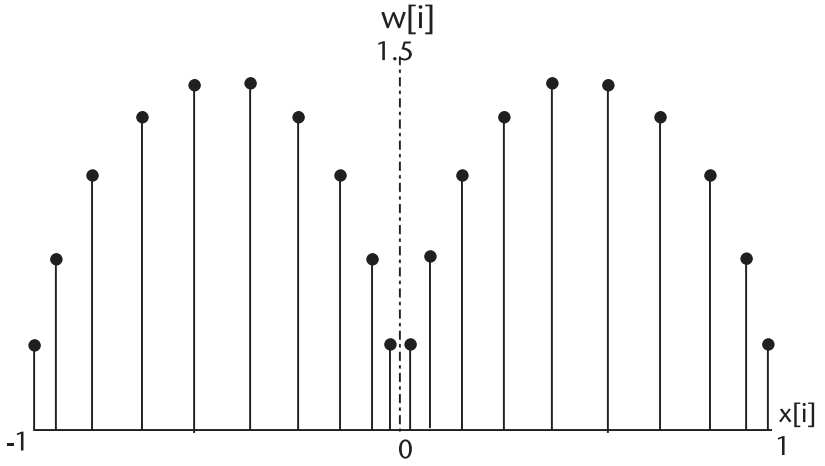


Figure 9.18 Abscissa and weight, Legendre polynomial n=20.

9.9.3.2 Gauss-Chebyshev Quadrature

The following integration will be evaluated by Gauss-Chebyshev quadrature.

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\,dx = \sum_{i=1}^{N} w_i\, f(x_i)$$

Chebyshev polynomials have the recurrence formula,

$$T_0(x) = 1$$
$$T_1(x) = 1$$
$$T_2(x) = 2x2 - 1$$
$$T_3(x) = 4x3 - 3x$$
$$\cdot \qquad \cdot$$
$$\cdot \qquad \cdot$$
$$T_{n+1}(x) = 2x\, T_n(x) - T_{n-1}(x)$$

Figure 9.19 shows Chebyshev polynomials up to the fourth degree. The polynomials are orthogonal in the range [–1, 1].

$$\int_{-1}^{1} \frac{T_n(x)\,T_m(x)}{\sqrt{1-x^2}}\,dx = \begin{cases} 0 & \text{when } n \neq m \\ \pi & \text{when } n = m \end{cases}$$

The abscissa x_i, the ith root of Chebyshev polynomials in [–1, 1] is given by,

$$x_i = \cos\left[\frac{\pi(i-1/2)}{n}\right]$$

and the corresponding weights are equal for all w_i, $w_i = \frac{\pi}{n}$. (See Figure 9.20.)

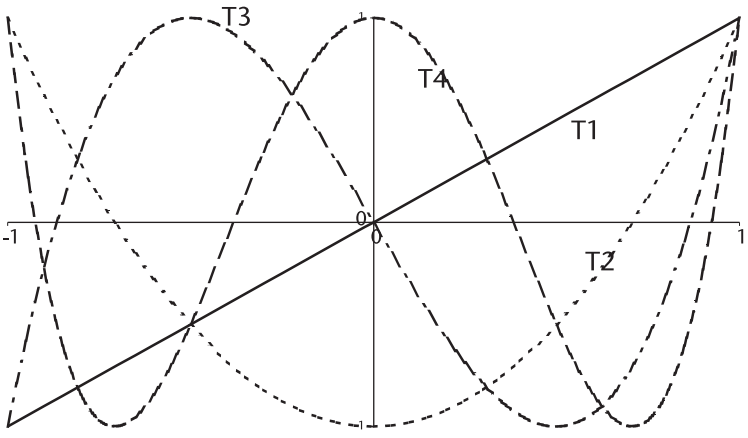


Figure 9.19 Chebyshev polynomials.

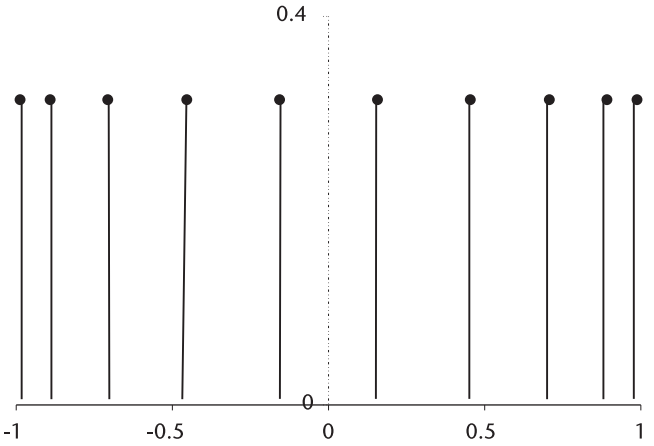


Figure 9.20 Weights of Chebyshev polynomial, n=10.

Two functions $f(x) = \frac{x^3}{\sqrt{1-x^2}}$, and $f(x) = \frac{\ln x}{\sqrt{1-x^2}}$ are integrated by Gauss-Chebyshev quadrature in [0, 1] in GAU_CHEB.CPP. The analytic answer is 2/3 and $-\pi/2 \ln(2)$. The first integrand has a singularity at x=1.0. The largest abscissa is 0.998630, a difference of 1.0–0.998630=0.001370. The second integrand has an additional singularity at x=0.0. The smallest abscissa is 0.052336, almost 38 times as large at x=1.0 of the first integrand. We expect a poor result for the second integral if we use the same degree, n=10. A suggestion may be an increase of n, or a better approach may be a change of the function ln x so that the additional singularity would be eliminated.

If we let $\ln x = -t$, $x = e^{-t}$, $dx = -e^{-t}\,dt$, the integrand will be transformed as shown below, and the singularity at x=0 is eliminated.

$$\int_{x=0}^{x=1} \frac{\ln x}{\sqrt{1-x^2}}dx = \int_{t=\infty}^{t=0} \frac{t\,e^{-t}}{\sqrt{1-e^{-2t}}}dt$$

An integration with limit [0, ∞] will be discussed in Gauss-Laguerre quadrature in the next section. When the limits are [0, 3] in the first integration, we would integrate in two parts. [0, 1] and [1, 3] and sum them.

9.9.3.3 Gauss-Laguerre Quadrature

The following integration can be evaluated by the Gauss-Laguerre quadrature.

$$\int_0^{\infty} e^{-x} f(x)\,dx = \sum_{i=1}^{N} w_i f(x_i)$$

Laguerre polynomials have the following recurrence formula,

$$L_0(x) = 1$$

$$L_1(x) = 1 - x$$

$$L_2(x) = {}^1\!/_2(x^2 - 4x + 2)$$

. .
. .
. .

$$L_{n+1} = \left(\frac{2n + 1 - x}{n + 1}\right) L_n(x) - \left(\frac{n}{n + 1}\right) L_{n-1}(x)$$

Laguerre polynomials up to five degrees are shown in Figure 9.21.
The abscissas and the corresponding weights are computed in LAGUE- RRE.CPP.

[i]	x[i]	w[i]
1	0.037793	3.084411E-01
2	0.729455	4.011199E-01
3	1.908343	2.180683E-01
4	3.401434	6.208746E-02
5	5.552496	9.501517E-03
6	8.330153	7.530084E-04
7	11.843786	2.825923E-05
8	16.279259	4.249314E-07
9	21.996586	1.839565E-09
10	29.920696	9.911827E-13

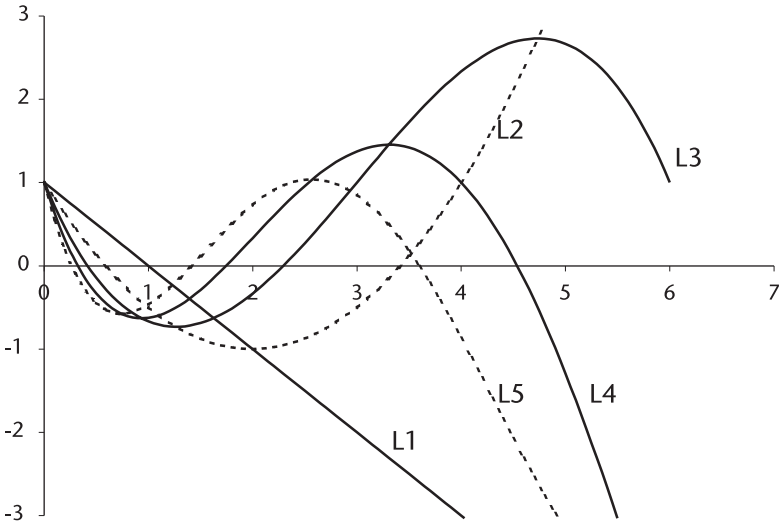


Figure 9.21 Laguerre polynomials, n=5.

A demonstration program is written in GAU_LAG.CPP for two functions:

$$\int_0^\infty e^{-x} x \sin(x)\, dx, \quad \int_0^\infty \frac{e^{-x}\, x}{\sqrt{1-e^{-2x}}} dx$$

The second integral is the one we discussed in the previous section while eliminating a singularity. Read GAU_LAG.CPP in the file.

9.9.3.4 Gauss-Hermite Quadrature

The following integration can be evaluated by Gauss-Hermite quadrature.

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)\, dx = \sum_{i=1}^{N} w_i f(x_i)$$

Hermite polynomials have recurrence formulas in two different forms; one is orthogonal polynomials given by [4], and the other orthonormal polynomials by [11].

Orthogonal	Orthonormal
$H_0(x) = 1$	$H_0(z) = 1/\pi^{1/4}$
$H_1(x) = 2x$	$H_1(z) = z\sqrt{2}/\pi^{1/4}$
$H_2(x) = 4x^2 - 2$	$H_2(z) = (z^2 - 1/2)\sqrt{2}/\pi^{1/4}$
$H_3(x) = 8x^3 - 12x$	$H_3(z) = (z^3 - 3z/2)(\sqrt{2}/\sqrt{3})\sqrt{2}/\pi^{1/4}$
.	.
.	.
.	.
.	.

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x) \quad H_{n+1}(z) = z\sqrt{\frac{2}{n+1}}H_n(z) - \sqrt{\frac{n}{n+1}}H_{n-1}(z)$$

The corresponding weights w_i are also given in two different forms:

$$w_i = \frac{2^{n-1} n! \sqrt{\pi}}{n^2 [H_{n-1}(x_i)]^2} \qquad w_i = \frac{2}{[H'(z_i)]^2}, \quad H'_n = \sqrt{2n} H_{n-1}$$

The orthonormal polynomials are preferred, since the weights given by Abramowitz and Stegun are very large and often overflow when degree n is moderately high due to the factorial n! involved.

The Hermite polynomial (orthonormal) are shown in Figure 9.22 for positive z only.

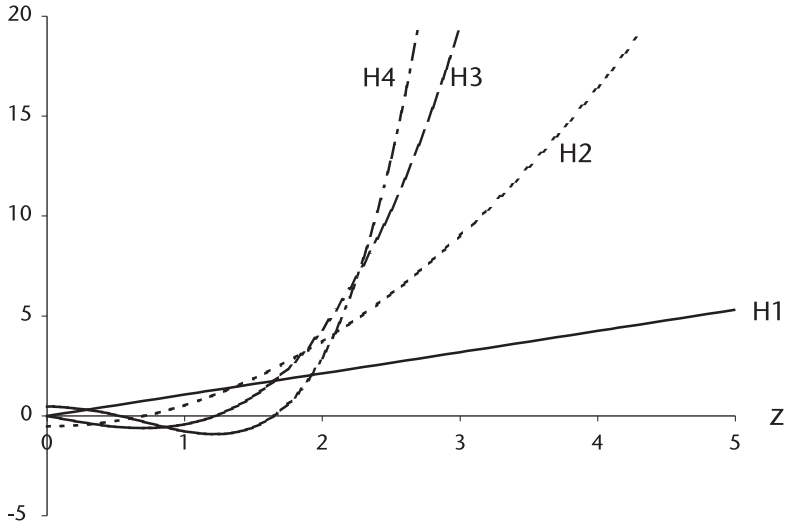


Figure 9.22 Hermite polynomials.

The abscissa and weight are computed in HERMITE.CPP, n=15.

[i]	z[i]	w[i]
1	4.499991	1.522476E-09
2	3.669950	1.059116E-06
3	2.967167	1.000044E-04
4	2.325732	2.778069E-03
5	1.719995	3.078003E-02
6	1.136116	1.584889E-01
7	0.565070	4.120287E-01
8	0.000000	5.641003E-01 (symmetry)
9	–0.565070	4.120287E-01
10	–1.136118	1.584889E-01
11	–1.719993	3.078003E-02
12	–2.325732	2.778069E-03
13	–2.967167	1.000044E-04
14	–3.669950	1.059116E-06
15	–4.499991	1.522476E-09

A demonstration program is written in GAU_HERM.CPP for the integrand $f(x)= e^{-x^2} x^2$ in the range $[0, +\infty]$. A header file HERMITR.H is created to compute the abscissas and weights for an arbitrary order of Hermite orthonormal polynomials.

9.10 Quadrature in Two Dimensions

One dimensional quadrature can be extended to two dimensions with careful declaration on the limits of inner and outer integrations. We demonstrate the techniques with three examples.

A two-dimensional quadrature would be evaluated in two steps:

$$I = \int_{x_1}^{x_2} \int_{y_1}^{y_2} f(x,y)dx\ dy$$

A temporary inner integral H(x) is evaluated first,

$$H(x) = \int_{y_1(x)}^{y_2(x)} f(x,y)\ dy$$

and, finally the outer integral

$$I = \int_{x_1}^{x_2} H(x)\ dx$$

The one-dimensional integral algorithm (trapezoidal, Simpson's extended rule, or Gauss-hyphenated-polynomial routine) is executed with an increment Δx on the outer integrand and many times, recursively, on the inner integrand over the specified limits of integration.

Example 1.

$$\int_{x_1=0}^{x_2=1} \int_{y_1=1+x^2}^{y_2=1+x} 2xy\ dx\ dy \tag{9.25}$$

Equation (9.25) is evaluated by two different algorithms: one by the extended Simpson's rule and other by Gauss-Legendre quadrature. For Simpson's extended rule we follow five steps described as follows.

1. Declare the integrand f(x,y)=2xy and the limits of inner integration, y_1 and y_2.
2. Apply Simpson's extended rule, one-dimensional routine, to the integration.
3. Declare three pointers; first pointer to f(x,y), second pointer to the lower limit $y_1(x)$, and third pointer to the upper limit $y_2(x)$.
4. Construct a prototype of 2D_Simpson algorithm; for example,

   Simpson_2D(a, b, y1, y2, FUNC_f(x,y), N)

   where "a" and "b" are unspecified general limits of outer integration, and N the number of subintervals of Simpson's rule.
5. Execute Simpson_2D algorithm after x_1 and x_2 specified.

The steps described above are written in 2D_SIMPS.CPP and the result is shown below.

Number of subintervals	Answer	Error
10	0.249967	3.328919E-05
12	0.249984	1.601875E-05
14	0.249991	8.657575E-06
16	0.249995	5.081296E-06
18	0.249997	3.218651E-06
20	0.249998	2.056360E-06
22	0.249999	1.415610E-06
24	0.249999	9.685755E-07
26	0.249999	7.301569E-07
28	0.249999	5.513430E-07
30	0.250000	4.023314E-07
32	0.250000	3.129244E-07
34	0.250000	2.682209E-07
36	0.250000	2.086163E-07
38	0.250000	1.341105E-07

As the number of subintervals increases the quadrature approaches the correct answer. How do we know the number of subintervals required at the outset of the program?

$$\int_{x_1}^{x_2}\int_{y_1}^{y_2} f(x,y)\,dx\,dy = \sum_{a}^{b}\sum_{i}^{N} f(x_i, y_i) + \text{remainder}$$

The remainder (error) is given by [4] (equation [25.9.4]) for Simpson's rule.

$$\text{remainder (error)} \leq \frac{Nh^5}{90} f^{(4)}(\xi)$$

where h=(b–a)/N, and $f^{(4)}(\xi)$ is the maximum value of fourth derivative of the integrand in [a, b]. Substituting $f^{(4)}(1) \approx 120$, we estimate the number of subintervals N for an error of 1.0E-6.

$$\text{error} \leq \frac{N}{90}\frac{(b-a)}{N^5} f^{(4)}(\xi)$$
$$= \frac{120}{90\ N^4}$$

and,

$$N \geq 34$$

Equation (9.25) is again evaluated by Gauss-Legendre quadrature, written in 2D_LEGDR.CPP. The program procedures are identical to 2D_SIMPS. CPP except that the step (2) is replaced by one-dimensional version demonstrated in Section 9.9.3.1, GAU_LEGX.CPP, with only ten x_i and w_i to obtain a correct answer.

Example 2.

$$\int_{x_1=0}^{x_2=\infty} \int_{y_1=0}^{y_2=x} e^{-(x+y)} dx\ dy \tag{9.26}$$

Equation (9.26) is evaluated in 2D_GAUSS.CPP using x_i and w_i given by Abromowitz and Stegun [4] (Table [25.4]) n=20. Only one-half of x_i and w_i are included in the program due to symmetry.

Example 3.

$$\int_{x_1=0}^{x_2=1} \int_{y_1=1+x^2}^{y_2=1+x} xy\ e^{-y^2} dx\ dy \tag{9.27}$$

Equation (9.27) is evaluated in 2D_GAUSS.CPP using the previous x_i and w_i. The result is surprisingly accurate with only n=10.

9.11 Quadrature in Three Dimensions

Integration in three variables x, y and z can be evaluated by extending the two-dimensional quadrature having two intermediate integrals G(x, y) and H(x) as shown.

$$I = \int_{x_1}^{x_2} \int_{y_1}^{y_2} \int_{z_1}^{z_2} f(x,y,z)\ dx\ dy\ dz$$

Two intermediate integrals G(x, y) and H(x) would be evaluated first,

$$G(x,y) = \int_{z1(x,y)}^{z2(x,y)} f(x,y,z)dz$$

$$H(x) = \int_{y1(x)}^{y2(x)} G(x,y)dx$$

and finally

$$I = \int_{x1}^{x2} H(x)\ dx$$

A basic one-dimensional quadrature is executed for the outermost integration, and the two intermediate integrals G(x, y) and H(x) are called recursively many times.

A rather simple integrand is taken up as a demonstration.

$$\int_{x1=0}^{x2=1} \int_{y1=0}^{y2=x} \int_{z1=0}^{z2=1+xy} 2xyz \, dx \, dy \, dz \tag{9.28}$$

Equation (9.28) is evaluated in 3D_LRGDR.CPP with ten x_i and w_i that we have generated in LEGENDRE.CPP previously. The CPP structure is similar to 2D_LEGDR.CPP or 2D_GAUSS.CPP with an additional intermediate integral.

The number of pointers is increased five, See 3D_LRGDR.CPP.

9.12 Concluding Remarks

I agonized over whether to expand this chapter or delete it entirely—and failed in both! I hope readers will be generous and gracious to accept these meager presentations on the Monte Carlo. Hammersly and Henderscomb [7] provide a rich trove of historical anecdotes and bibliography; Fishman [6] presents exhaustive mathematical derivations and algorithms; and Rubinstein [3] an excellent selection of materials for graduates or advanced undergraduates for one semester or two.

List of Programs

Program	*Features*
(1) HIT_MISS.CPP	Monte Carlo integral of irregular, complex f(x) that defies a mathematic expression, by the hit-or-miss method
(2) ERR_ABST.CPP	Computes the error function defined by Abramowitz and Stegun [4]
(3) ERR_GAUS.CPP	Computes the error function defined in standard textbooks
(4) ORDERED.CPP	Monte Carlo integral by ordered random numbers
(5) SAMPMEAN.CPP	Monte Carlo integral by sample mean
(6) IMPRTANC.CPP	Monte Carlo integral by importance sampling method
(7) PRN1KA.CPP	Generates 1,000 PRNs
(8) PRN1KB.CPP	Generates a second set of 1,000 PRNs
(9) CD_GAUFM.CPP	Computes the CD, Gaussion probability density function
(10) Q_THRES.CPP	Computes the threshold value, Q-function
(11) PFA_EXP.CPP	Monte Carlo integral of the probability of false alarm, exponential pdf
(12) PFA_GAU.CPP	Monte Carlo integral of the probability of false alarm, Gaussian pdf
(13) TRAPZOID.CPP	Computes an integral by the trapezoidal rule
(14) SIMPSON.CPP	Computes an integral by Simpson's rule
(15) SIMPSONX.CPP	Computes an integral by Simpson's extend rule
(16) LEGENDRE.CPP	Generates abscissas and weights of Legendre polynomials
(17) GAUS-LEG.CPP	Gauss-Legendre quadrature
(18) GAU_LEGX.CPP	Gauss-Legendre quadrature, extended
(19) GAU_CHEB.CPP	Gauss-Chebyshev quadrature
(20) LAGUERRE.CPP	Generates abscissas and weights of Laguerre polynomials
(21) GAUS_LAG.CPP	Gauss-Laguerre quadrature
(22) HERMITE.CPP	Generates abscissas and weights of Hermite polynomials
(23) HERMITE.H	A header file for Hermite polynomial abscissas and weights, to be attached to the main driver
(24) GAU_HERM.CPP	Gaussian-Hermite quadrature
(25) 2D_SIMPS.CPP	Two dimensional integration with Simpson's rule
(26) 2D_LEGDR.CPP	Two dimensional integration with Legendre polynomials
(27) 2D_GAUSS.CPP	Two dimensional integration with Gaussian abscissas and weights
(28) 2D_GAUSZ.CPP	Second demonstration of Gauss-quadrature
(29) 3D_LEGDR.CPP	Three-dimensional integration with Legendre polynomials

References

[1] Sobol, I. H., *A Primer for Monte Carlo Method*, Boca Raton, FL: CRC Press, 1994.

[2] Kalos, M. H. and P.A. Whitlock, *Monte Carlo Methods*, Vol. 1, New York, NY: John Wiley & Sons, 1986.

[3] Rubinstein, R.Y., *Simulation and the Monte Carlo Method*, New York, NY: John Wiley & Sons, 1981.

[4] Abramowitz, M., and I. A., Stegun, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, National Bureau of Standards, Series 55, 1964.

[5] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, New York, NY: McGraw-Hill, 1965.

[6] Fishman, F. S., *Monte Carlo, Concept, Algorithms and Applications*, New York, NY: Springer-Verlag, 1996.

[7] Hammersley, J. M. and D. C. Hendscomb, *Monte Carlo Methods*, New York, NY: Chapman & Hall, 1964.

[8] Brennan, L. E. and Reed, I.S. "A Recursive Method of Computing Q Function," *IEEE Trans. AES*, Vol. 11, Apr. 1965.

[9] Hansen, V. G. *Importance Sampling in Computer Simulation of Signal Processor Computers and Electric Engineering*, Vol. 1, New York, NY: Pergamon Press, 1974.

[10] Mitchell, R. L., Importance Sampling Applied to Simulation of False Alarm Statistics, *IEEE Trans. AES*, Vol. 17, No.1, Jan. 1981.

[11] Press, W. H., et al., *Numerical Recipes in C*, second ed., Cambridge, U.K.: Cambridge Univ. Press, 1995.

Appendix 9A

The area occupied by the overlapped two circles is:

$$A = [\pi r_1^2 + \pi r_2^2] - 4\left\{\pi r_1^2\left(\frac{\theta}{360}\right) - \frac{1}{2}\left[\left(\frac{r_1 - r_2 + d}{2}\right) r_1 \sin\theta\right]\right\}$$

where

$$\cos\theta = \frac{1}{r_2}\left(\frac{r_1 + r_2 - d}{2}\right).$$

CHAPTER 10

Constant False Alarm Rate (CFAR) Processing

10.1 Introduction

One of most important signal processing objectives in target detection in locally varying homogeneous noise or clutter is to maintain the false-alarm rate as a constant. Constant false-alarm rate (CFAR) processing is one of maybe half a dozen indispensable types of signal processing for the successful operation of a radar system.

The classic theory of target detection assumes that the noise is distributed as Gaussian with unknown power. When an antenna sweeps the surveillance sectors, the radar receives noise and clutter returns, and they may not be distributed as Gaussian; clutter may be distributed as lognormal, Weibull, gamma or K distribution, or the like. We shall investigate a few CFAR processing methods in Gaussian noise and Weibull clutter.

10.2 Cell-Averaged CFAR (CA-CFAR)

Finn and Johnson [1] have reported a CFAR processing method for use when the target signal received is embedded in Gaussian noise of unknown power level. The probability density function of noise is Gaussian; the unknown is the locally varying noise power. We follow their analysis.

When the input to a square-law detector is narrowband Gaussian noise with zero mean and unknown variance (noise power), the output is distributed as exponential with unknown variance [2–4]. (See Figure 10.1.)

$$f_x(x) \rightarrow \boxed{y = x^2} \rightarrow f_y(y)$$

$$f_x(x) = \frac{1}{\sigma_x\sqrt{2\pi}} \exp\left\{\frac{-x^2}{2\sigma_x^2}\right\}$$

$$f_y(y) = \frac{1}{2\sigma_x^2} \exp\left\{\frac{-y}{2\sigma_x^2}\right\} = \frac{1}{\sigma_y^2} \exp\left\{\frac{-y}{\sigma_y^2}\right\}, \qquad (2\sigma_x^2 = \sigma_y^2)$$

We note that the mean of the exponential is $E\{y\}=\sigma_y$, and the average of the random variable y is equal to the standard deviation of y. In radar literature we call $f_y(y)$ the probability density function of the square-law detector output when the input is a narrowband Gaussian noise, G(mean=0, var=σ_x^2).

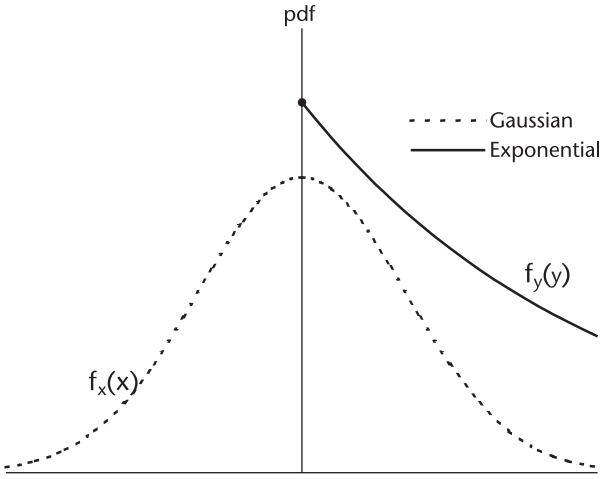
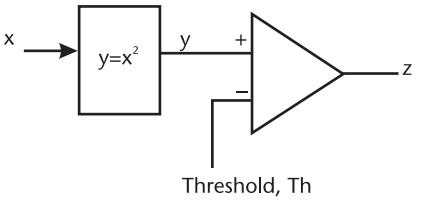


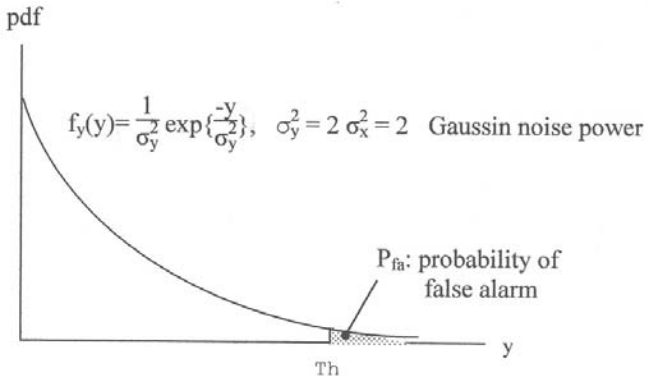
Figure 10.1 Probability density functions: Gaussian and exponential.

Consider that we have cascaded a comparator with a threshold voltage, Th, after the square-law detector, as shown.

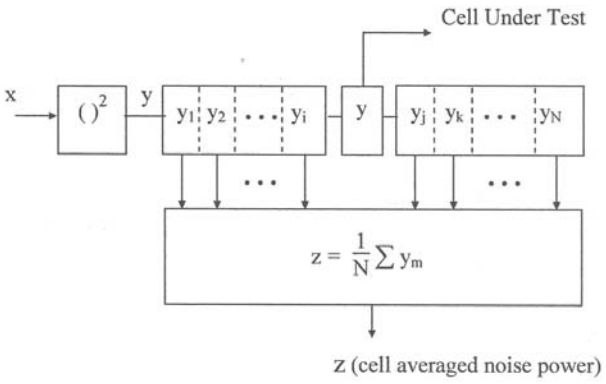


The probability of false alarm P_{fa} is given by

$$P_{fa} = \int_{Th}^{\infty} \frac{1}{\sigma_y^2} \exp\left\{\frac{-y}{\sigma_y^2}\right\} dy = \exp\{-Th\} \tag{10.1}$$



The noise or clutter power at the input to the square-law detector varies as the antenna observes different features of the surface or volume as it sweeps. The following network (or signal processing) would estimate the locally varying noise power.



We call $y_1, y_2, \ldots, y_i, y_j, \ldots, y_N$ the reference cells. They are grouped into two reference windows, and a test cell Y is inserted between the windows in order to detect a target as $y_1, y_2, \ldots, y_i, y_j, \ldots, y_N$ slide through the reference cells, as shown in Figure 10.2.

Mathematically then, the false alarm probability is defined as

$$P_{fa} = \int_{z=0}^{\infty} \left[\int_{Th}^{\infty} f_y(y_o)\, dy \right] f_z(z)\, dz \tag{10.2}$$

where $f_y(y_o)$ is the pdf of the test cell when it is filled with noise

$$f_y(y_o) = \frac{1}{\sigma_y^2} \exp\left\{ \frac{-y}{\sigma_y^2} \right\} \tag{10.3}$$

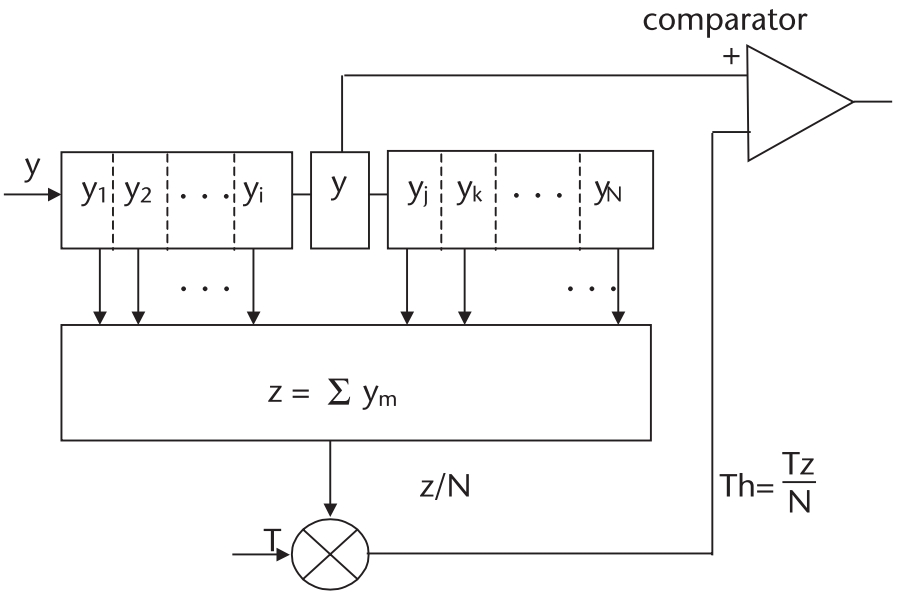


Figure 10.2 CA-CFAR circuits or processing.

and $f_z(z)$ is the pdf of the sum of surrounding reference cells. In order to evaluate P_{fa} we have to know the exression for $f_z(z)$. When the random variables y_k, k=1, 2, 3, . . . , N, are distributed exponentially, the sum $z = \Sigma y_k$ is distributed as a Gamma function [5].

$$f_z(z) = \frac{1}{\Gamma(N)} \left(\frac{1}{\sigma}\right)^N z^{N-1} \exp\left\{\frac{-z}{\sigma}\right\} \tag{10.4}$$

Substituting (10.3) and (10.4) into (10.2), the false alarm probability is obtained.

$$\begin{aligned} P_{fa} &= \int_0^\infty \left[\int_{Tz/N}^\infty \frac{1}{\sigma^2} \exp\left\{\frac{-y}{\sigma^2}\right\} dy \right] \frac{1}{(N-1)!} \left(\frac{1}{\sigma}\right)^N z^{N-1} \exp\left\{\frac{-z}{\sigma}\right\} dz \\ &= \frac{1}{\sigma^N (N-1)!} \int_0^\infty z^{N-1} \exp\left\{-\left[\frac{1+T/N}{\sigma}\right] z\right\} dz \\ &= \frac{1}{[1+T/N]^N} \end{aligned} \tag{10.5}$$

The result shows that the false-alarm probability is independent of noise power σ^2, and therefore, the circuit or the processing of Figure 10.2. is CFAR. Since the local noise power is estimated by averaging noise in the cells, we call it cell-averaged CFAR (CA-CFAR).

Solving for T in (10.5), the threshold multiplication factor is obtained.

$$T = N\left[\exp\left\{\frac{-\ln P_{fa}}{N}\right\} - 1\right] \tag{10.6}$$

Equation (10.5) is programmed in CA_MULTI.CPP and the results are shown below.

N	$T(P_{fa}$=1.0E-6)	$T(P_{fa}$=1.0E-5)
4	122.4911	67.1312
6	54.0000	34.8775
8	36.9873	25.7357
10	29.8107	21.6228
12	25.9473	19.3219
.	.	.
.	.	.
20	19.9052	15.5656
30	17.5468	14.0340
.	.	.
.	.	.
.	.	.

100	14.8154	12.2018
.	.	.
.	.	.
Infinity	13.8155	11.5128

The threshold multiplier T for N = infinity is obtained from (10.1) with σ_y^2=1.

The expression for the probability of detection P_d is identical to (10.2) except that the inner integral, the bracketed term, represents the pdf of a target return plus noise in the test cell Y.

$$P_d = \int_0^{\infty} \left[\int_{Tz/N}^{\infty} f_y(y_1)dy \right] f_z(z)\,dz \tag{10.7}$$

The probability density functions of Marcum's target model and Swerling's target models 1, 2, 3, and 4 are given in Chapter 7, repeated below for Swerling's target with a single return.

$$\text{Swerling target model 1 \& 2: } f_y(y_1) = \frac{1}{1+\overline{y}} \exp\left\{\frac{-y}{1+\overline{y}}\right\} \tag{10.8}$$

$$\text{Swerling target model 3 \& 4: } f_y(y_1) = \frac{y}{1+\overline{y}} \exp\left\{\frac{-y}{1+\overline{y}}\right\} \tag{10.9}$$

where

$\overline{y}$ = Average SNR in power;
y = Instantaneous SNR in power.

Substitution of (10.8) and (10.4) into (10.7) yields the detection probability of Swerling's targets 1 and 2.

$$P_d = \int_0^{\infty} \left[\int_{Tz/N}^{\infty} \frac{1}{(1+SNR)} \exp\left\{\frac{-y}{1+SNR}\right\} dy \right] \frac{1}{\Gamma(N)} \left(\frac{1}{\sigma}\right)^N z^{N-1} \exp\left\{\frac{-z}{\sigma}\right\} dz$$

$$= \frac{1}{\left[1 + \frac{T}{N(1+SNR)}\right]^N} \qquad N = \text{finite} \tag{10.10}$$

When the number of reference cells is infinity, N=∞, the noise statistics are perfectly known, and P_d is given by

$$P_d = \exp\left\{\frac{-T_\infty}{(1+SNR)}\right\}, \quad N = \infty \tag{10.11}$$

We call the above detection probability the no-CFAR probability. The reduction in P_d incurred by using a finite number of reference cells is the difference between (10.10) and (10.11).

$$\text{CA-CFAR}_{(\text{loss})} = P_d(N = \infty) - P_d(N = \text{finite})$$

Equation (10.10) is programmed in PdCACFAR.CPP as the signal-to-noise ratio (in power) is incremented by 1 dB, and the number of reference cells N is an independent parameter; N=8, 16, 32 and 64. Equation (10.11) is programmed in PdNoCFAR.CPP. The results of the two programs are shown in Figure 10.3. The loss of the CA-CFAR with a finite number of reference cells is indicated. The CA-CFAR loss can be computed in another way. The loss is obtained by the ratio of two SNRs of (10.10) and (10.11).

$$\text{SNR}_{(N=\text{finite})} = \frac{1}{N}\left[\frac{T_N}{(P_d)^{-1/N} - 1} - 1\right] - 1 \tag{10.12}$$

$$\text{SNR}_{(N=\infty)} = \frac{-T_\infty}{\text{Ln } P_d} - 1 \tag{10.13}$$

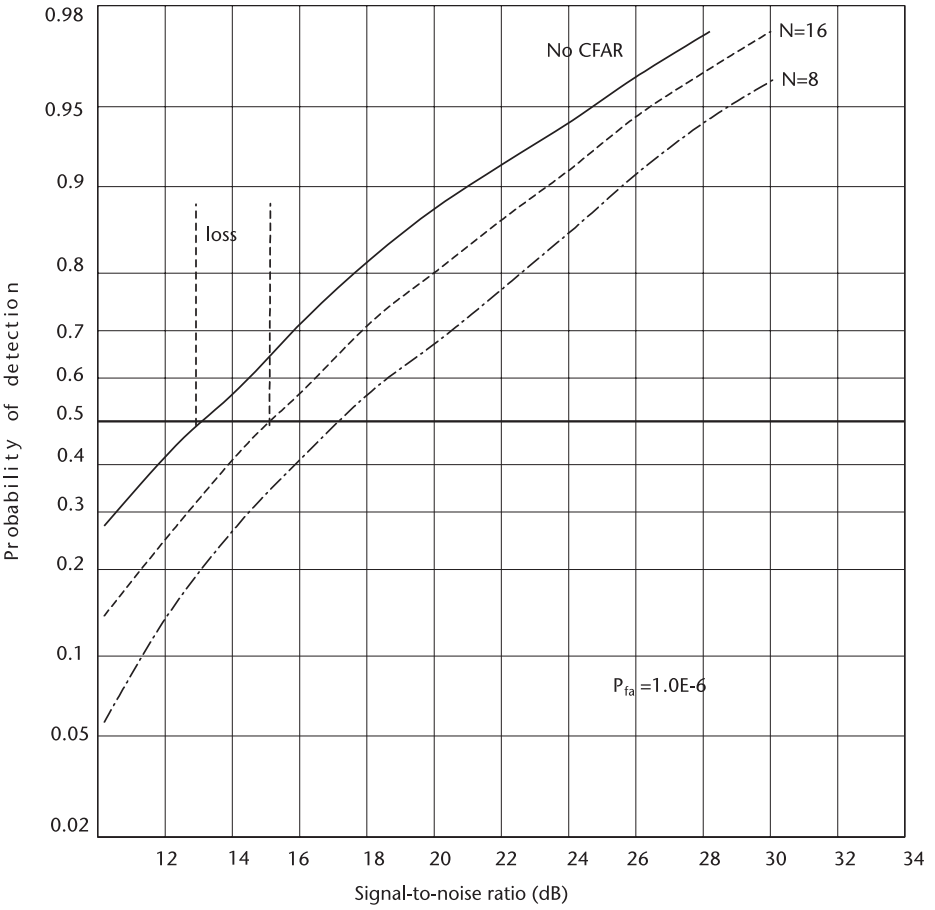


Figure 10.3 Probability of detection, CA-CFAR. (For accurate reading, refer to the corresponding CPP and DAT file.)

$$\text{loss}_{(\text{dB})} = 10 \log \frac{[(10.12)]}{[(10.13)]} \approx 10 \log \left(\frac{T_N}{T_\infty} \right) \tag{10.14}$$

Equation (10.14), the exact expression, is programmed in CA_LOSS.CPP, shown in Figure 10.4. The approximation is accurate within 0.01 dB when $N \geq 8$ and $P_d \geq 0.5$.

We observe in Figure 10.4 that the loss decreases as the number of reference cells increases; the larger the N, the smaller the loss. However, we cannot increase the number of reference cells as much as we wish, since there is a finite number of range bins. A compromise must be struck. There is collateral damage in increasing the number of reference cells, so-called clutter-edge-widening. We shall discuss this problem after we present a simulation program.

We propose, for a demonstration, the following clutter-level profile across the range, as shown in Figure 10.5. The proposed clutter profile is not very realistic; however, through this demonstration we shall see the response of CA-CFAR processing when the clutter levels change abruptly.

The clutter-edge response is clearly visible. The width of the response is approximately one-half of the reference window. The target return in these regions would be lost. The choice for the number of reference cells must be a compromise between a lower loss and the width of edge response.

The effect that is most detrimental to CA-CFAR processing, as depicted in the figures is when multiple targets reside in the reference window in addition to a

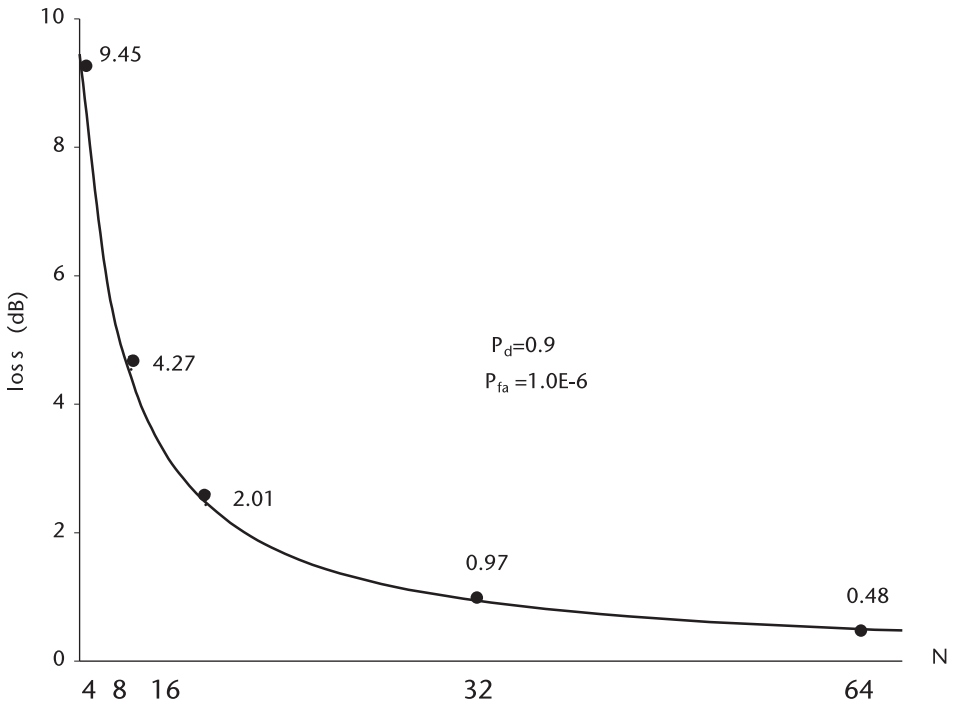


Figure 10.4 CA-CFAR Loss versus number of reference cells.

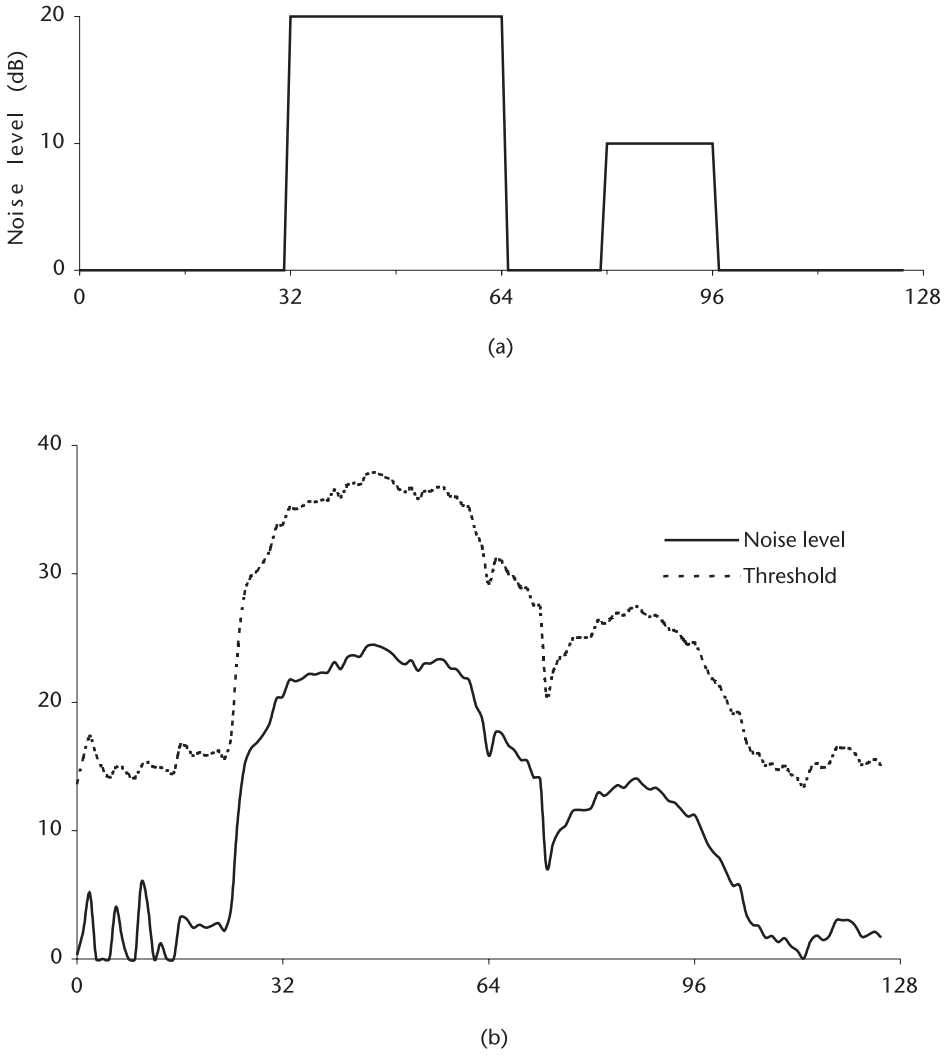


Figure 10.5 CA-CFAR, noise and threshold level N=16, P_{fa} =1.0E-6.

target in the test cell. These spurious targets will raise the noise level and threshold erroneously higher. The target in the test cell may be undetected.

Some papers propose so-called greatest-of- or smallest-of- CFAR processing, or GO-CFAR or SO-CFAR, which choose between the left half or right half of the reference window, whichever is greater or smaller. Both processing methods have some shortcomings. The best remedy appears to be a censoring technique to censor the highest one, two or three clutter samples in the reference window (they might be targets) from the estimate of clutter power level. We shall investigate one of these censoring techniques in the next section.

In a practical design of CA-CFAR processing we add a guard cell at the front and rear of the test cell as shown in Figure 10.6 since a strong target return at the test cell with a sloping skirt may spill over to the adjacent cells to raise the estimate of clutter level.

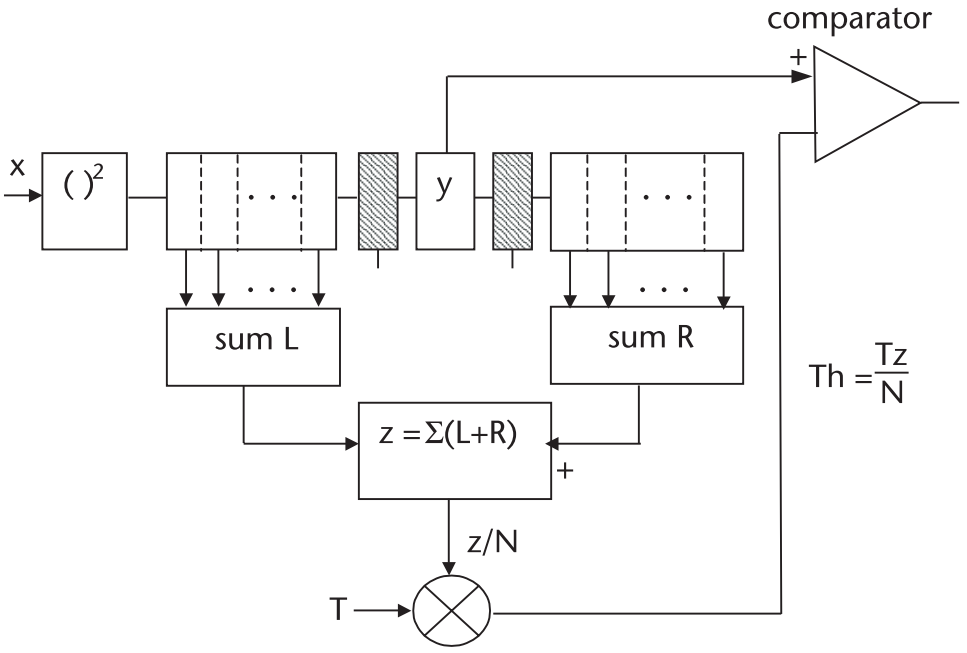


Figure 10.6 CA-CFAR processing with guard cells.

10.3 Order-Statistics CFAR, OS-CFAR

A CFAR processing method based on the theory of order statistics is reported by Rohling [6]. The motivation of his investigation is to handle multiple-target situations by censoring the spurious targets in the reference window. An attendant benefit is a narrower width of the clutter-edge response. A OS-CFAR processing is schematically shown in Figure 10.7.

We start with a rank-ordering operation on the clutter (or noise) stored in the reference window. We discuss the rank-order operations in Chapters 2 and 9. We rearrange the clutter samples in ascending order of the magnitude, considering that the samples may include spurious targets,

$$y_{(1)} \leq y_{(2)} \leq y_{(3)} \leq \cdots \leq y_{(i)} \leq y_{(j)} \leq \cdots \leq y_{(N)}$$

where $y_{(1)}$ is the lowest noise level, $y_{(i)}$ and $y_{(j)}$ are intermediate levels, and $y_{(N)}$ is the highest level. After the rank order we plan to censor one, two, three, or more of the highest samples and pick the kth sample level to control the threshold.

The false alarm probability P_{fa} is given as in CA-CFAR by,

$$P_{fa} = \int_{z=0}^{\infty} \left[\int_{y=Tz}^{\infty} f_y(y_o)\, dy \right] f_z(z)\, dz \tag{10.15}$$

The bracketed term is obtained directly,

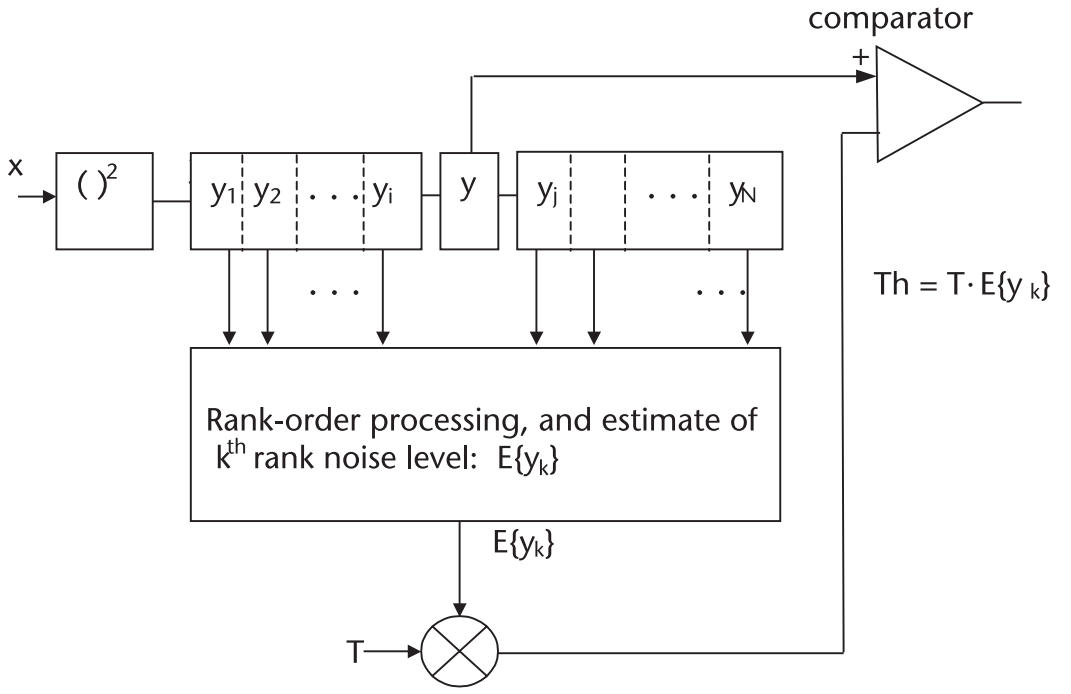


Figure 10.7 OS-CFAR processing (no guard cells necessary).

$$\int_{y=Tz}^{\infty} \frac{1}{\sigma^2} \exp\left\{\frac{-y}{\sigma^2}\right\} dy = \exp\left\{\frac{-Tz}{\sigma^2}\right\} \tag{10.16}$$

The probability density function of $f_z(z)$ is given by [7, 8], with that kth ranked cell as a parameter.

$$f_z(z) = \frac{k}{\sigma}\binom{N}{k}\left[1 - \exp\left\{\frac{-z}{\sigma}\right\}\right]^{k-1}\left[\exp\left\{\frac{-z}{\sigma}\right\}\right]^{N-k+1} \tag{10.17}$$

where

$$\binom{N}{k} = \frac{N!}{k!(N-k)!}$$

The false-alarm probability P_{fa} is obtained by substituting (10.16) and (10.17) into (10.15).

$$P_{fa} = \frac{k}{\sigma}\binom{N}{k}\int_{z=0}^{\infty} \exp\left\{\frac{-Tz}{\sigma^2}\right\}\left[1 - \exp\left\{\frac{-z}{\sigma}\right\}\right]^{k-1}\left[\exp\left\{\frac{-z}{\sigma}\right\}\right]^{N-k+1} dz \tag{10.18}$$

Two successive changes in variables yield an expression for the probability of false alarm.

$$\frac{z}{\sigma} = x, \quad dz = \sigma\, dx \quad \text{and} \quad e^{-x} = t, \quad dx = \frac{-dt}{t}$$

$$P_{fa} = k\binom{N}{k}\int_{t=0}^{1} t^{(T+N-k)}(1-t)^{k-1}dt \qquad \text{See [9]}$$

$$= \frac{N!}{(N-k)!}\,\frac{\Gamma(T+N-k+1)}{\Gamma(T+N+1)} \tag{10.19}$$

where $\Gamma(\cdot)$ is Gamma function and T is a threshold multiplier, not the threshold. Equation (10.19) is solved for T in OS_MULTI.CPP with P_{fa}=1.0E-6.

[k]	N=12	N=16	N=24	N=28
4	319.609	442.734	688.281	810.781
5	146.895	206.758	325.977	385.391
6	83.955	120.410	192.832	228.945
7	54.155	79.463	129.473	154.355
8	37.534	56.626	94.072	112.686
9	27.146	42.427	72.080	86.787
10	20.061	32.883	57.329	69.390
11	14.838	26.067	46.851	57.046
12	10.585	20.955	39.065	47.886
13		16.952	33.074	40.847
14		13.690	28.333	35.286
15		10.895	24.485	30.784
16		8.294	21.298	27.068
17			18.607	23.934
18			16.293	21.283
19			14.274	18.981
20			12.477	16.967
21			10.848	15.179
22			9.335	13.578
23			7.874	12.122
24			6.342	10.782
25				9.526
26				8.321
27				7.122
28				5.824

The threshold is the product of T and the estimated noise level in the kth cell.

$$Th = T \cdot E\{y_{(k)}\} \tag{10.20}$$

The expectation of the kth ranked cell is given by [7, 8],

$$E\{y_{(k)}\} = \sum_{i=N-k+1}^{N}\frac{1}{i} = \sum_{j=1}^{k}\frac{1}{N-k+j} \tag{10.21}$$

Equations (10.20) and (10.21) are programmed in OS_THRES.CPP and the result is shown in Figure 10.8.

The broad minimum trough for Th suggests that we may censor two, three, or four highest noise samples in the reference window, depending upon the processing loss we are willing to accept.

The loss for OS-CFAR processing is given by,

$$\text{loss(dB)} = 10 \ \log\left[\frac{\text{Th-OS(N \& k specified)}}{\text{Th-CA}(N=\infty)}\right] \tag{10.22}$$

The denominator Th-CA(N=∞) = 13.8155 for P_{fa}=1.0E-6 as mentioned in the previous section. Equation (10.22) is computed in OS_LOSS.CPP and the result is shown in Figure 10.9. The loss for CA_CFAR is added to the figure in order to show the relative merit of OS-CFAR processing. We observe a broad minimum trough for all N.

The probability of detection P_d is given by replacing the inner integral in (10.15) with the probability density function of noise plus Swerling target models 1 and 2, (10.22).

$$P_d = \int_{z=0}^{\infty}\left[\int_{y=Tz}^{\infty} f_y(y_1)\ dy\right] f_z(z)\ dz \tag{10.23}$$

$$f_y(y_1) = \frac{1}{1+\bar{y}}\exp\left\{\frac{-y}{1+\bar{y}}\right\} \tag{10.24}$$

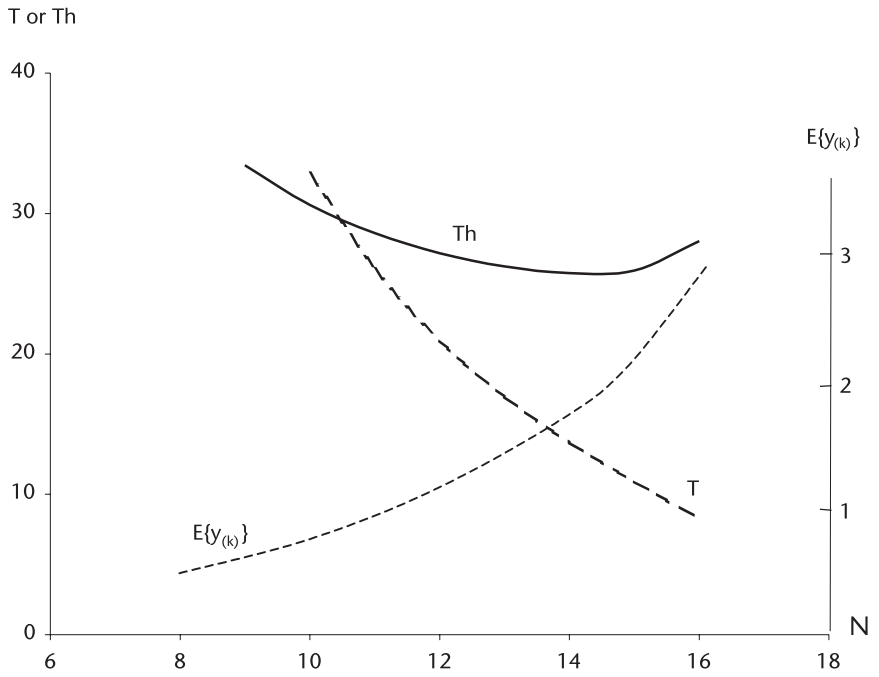


Figure 10.8 Threshold Th versus estimate of kth cell $E\{y_{(k)}\}$.

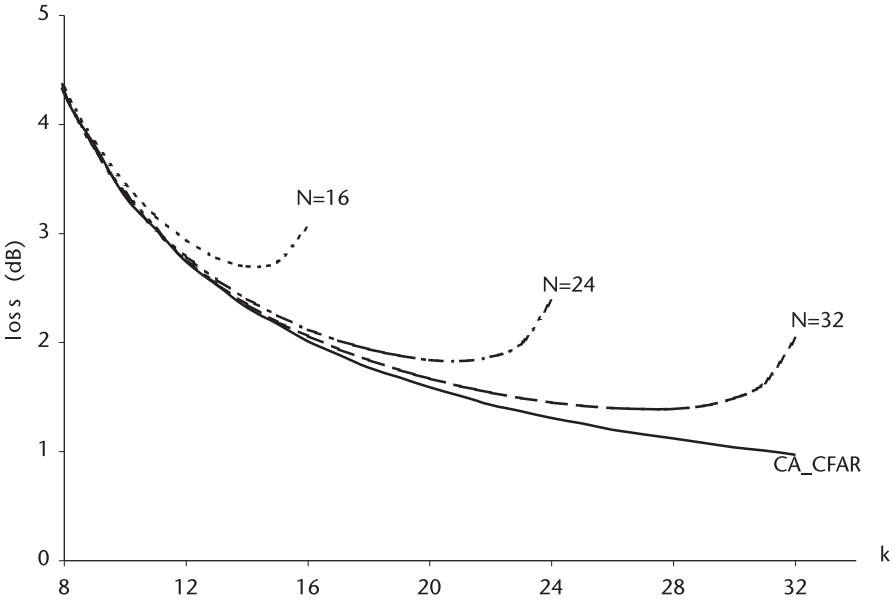


Figure 10.9 OS-CFAR Loss versus number of reference cells.

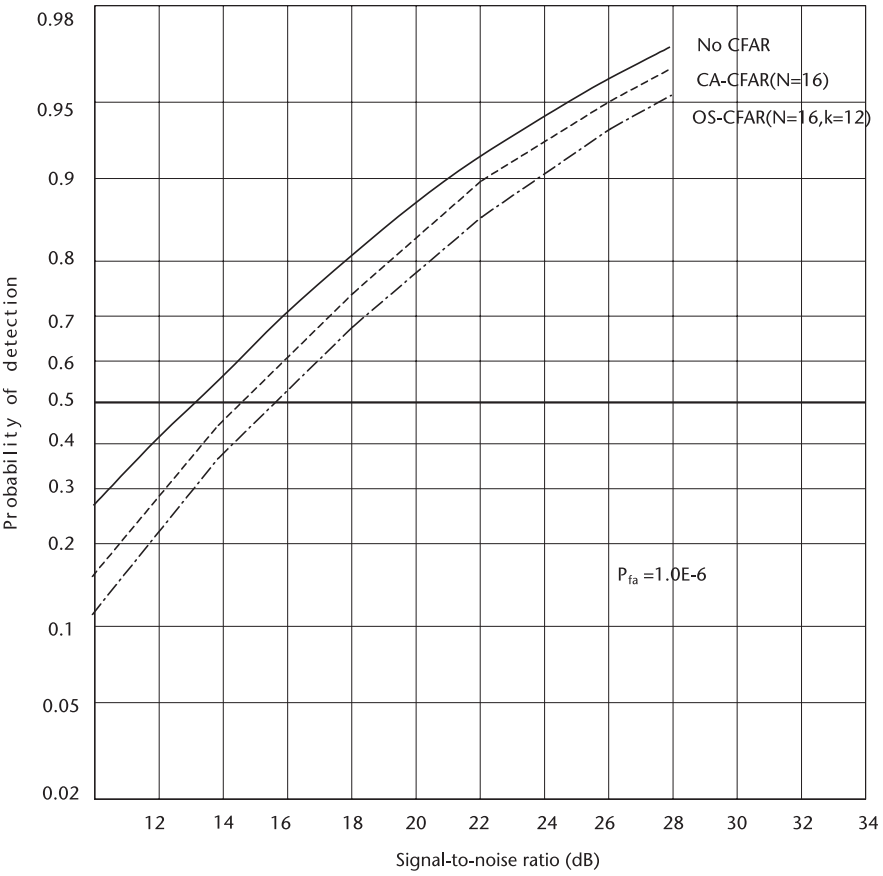


Figure 10.10 Detection probability OS-CFAR, N=16, k=12.

Substitution of (10.24) and (10.17) into (10.23) yields the detection probability as a function of SNR with two parameters N and k.

$$P_d = \frac{N!}{(N-k)!} \frac{\Gamma(DT + N - K + 1)}{\Gamma(DT + N + 1)} \tag{10.25}$$

where

$$D = \frac{1}{1 + SNR}$$

Equation (10.25) is programmed in Pd_OS_NK.CPP, and the result is shown in Figure (10.10). The lines for no-CFAR and CA-CFAR with N=16 are added to the graph for comparison. Note that the signal-to-noise ratio required for OS-CFAR is approximately 1dB higher than for CA-CFAR. This is a small price to pay for the inherent protection from spurious intrusion of up to four targets. We do not need the guard cells.

A narrower width of clutter-edge response is demonstrated for OS-CFAR. CPP.

Figure 10.11 shows that the clutter-edge response is much faster than that of CA-CFAR. In the region between two clutter banks the threshold is approximately 3 dB lower in OS-CFAR than in CA-CFAR.

So far we have investigated CFAR processing when clutter or noise is distributed as Gaussian, passed through a square-law detector. A large number of research

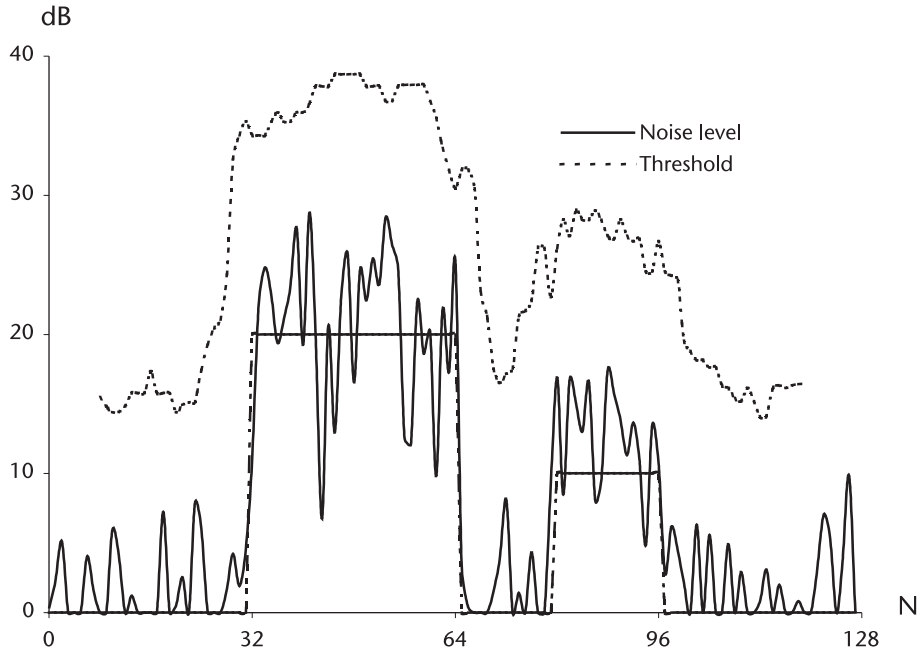


Figure 10.11 Clutter-edge response OS-CFAR, N=16, k=12.

papers indicate that the clutter returns are distributed as lognormal or Weibull. In the next sections we study CFAR processing in Weibull- distributed clutter.

10.4 Weibull Clutter

Before we investigate an optimal CFAR processing against Weibull clutter, we digress to review the statistics of a Weibull random variable. As radar resolution capability improves (as demonstrated by such factors as an ultra-narrow transmit pulse width and a wider signal bandwidth) it has been observed that the surface clutter returns become distributed as Weibull.

The Weibull probability density function is first reported by Swedish engineer Walodi Weibull in his study [10] on the strength of materials and the fatigue analysis. Later Boothe [11] and Sekine [12] and many others observed that clutter returns are indeed distributed as Weibull random variables.

10.4.1 Weibull Probability Density Function

The Weibull probability density function is a two-parameter function. It is defined variously in the literature, however, we adhere to the following notations throughout our discussion.

$$f_w(x) = \frac{c}{b}\left(\frac{x}{b}\right)^{c-1} \exp\left\{-\left(\frac{x}{b}\right)^{c}\right\} \tag{10.26}$$

where

c: The shape parameter, $c > 0$;
b: The scale parameter, $b > 0$;
x: Weibull variates, $x \geq 0$.

The scale parameter b of the Weibulll has an equivalent meaning attached to σ of a Gaussian or Rayleigh probability density function.

$$\text{Gaussian} \quad f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{\frac{-(x-m)^2}{2\sigma^2}\right\} \qquad -\infty \leq x \leq \infty$$

$$\text{Rayleigh} \quad f(x) = \frac{1}{\sigma^2} \exp\left\{\frac{-x^2}{2\sigma^2}\right\} \qquad x \geq 0$$

The Gaussian and Rayleigh pdf do not have a shape parameter; their shape is always the same. They have an equivalent scale parameter σ, that we call it the standard deviation. As σ changes the Gaussian or Rayleigh pdf gets sharper or flatter, but the shape is unchanged. The m of the Gaussian, the mean or expectation, is a location designator. Gaussian, Rayleigh, and many other probability density functions are one-parameter functions.

When the shape parameter c of Weibull is unity or less, the Weibull pdf is an exponential; when c=2 the Weibull is Rayleigh; and when c>2 Weibull resemble a Gaussian. The Weibull is very versatile function. See Chapter 2.

The probability distribution function (or the cumulative function) is obtained directly by integration of the density function.

$$\begin{aligned} F(x) &= \int_0^x \left(\frac{c}{b}\right)\left(\frac{x}{b}\right)^{c-1} \exp\left\{-\left(\frac{x}{b}\right)^c\right\} dx \\ &= 1 - \exp\left\{-\left(\frac{x}{b}\right)^c\right\} \end{aligned} \tag{10.27}$$

The first, second and rth moments of Weibull random variables are,

$$E\{x\} = \int_0^\infty x\, f(x)\, dx = b\Gamma\left(1+\frac{1}{c}\right) \tag{10.28}$$

$$E\{x^2\} = \int_0^\infty x^2\, f(x)\, dx = b^2\Gamma\left(1+\frac{2}{c}\right)$$

$$\vdots$$

$$E\{x^r\} = \int_0^\infty x^r f(x)\, dx = b^r\Gamma\left(1+\frac{r}{c}\right)$$

The variance is given as, by definition,

$$\begin{aligned} var\{x\} = \sigma_x^2 &= E\{x^2\} - E^2\{x\} \\ &= b^2\left[\Gamma\left(1+\frac{2}{c}\right) - \Gamma^2\left(1+\frac{1}{c}\right)\right] \end{aligned} \tag{10.29}$$

The mode is equal to the maximum value of the pdf; x_{mod} is the most likely value of the random variable, and its probability interpretation is that the sampled values in the range $x_{mod} \pm \Delta x$ have the highest probability of occurrence in $\pm\Delta x$. The mode is determined by

$$\begin{aligned} \frac{\partial}{\partial x}[f(x)] &= 0 \\ x_{mod} &= b\left[\frac{c-1}{c}\right]^{1/c} \end{aligned} \tag{10.30}$$

When $0 < c \leq 1$, the Weibull pdf is an exponential; there is no mode. The median x_{med} is obtained from (10.27) by letting F(x)=1/2.

$$\frac{1}{2} = 1 - \exp\left\{-\left(\frac{x}{b}\right)^c\right\}$$
$$x_{med} = b[\text{Ln } 2]^{1/c} \tag{10.31}$$

The x_{med} exists for all c, unlike the mode.
The quantiles of x, q=1/4, q=1/2, q=3/4 is obtained similarly.

$$x_q = b\left[\ln\left(\frac{1}{1-1/q}\right)\right]^{1/c} \tag{10.32}$$
$$x_{1/4} = b\ [\ln(4/3)]^{1/c}$$
$$x_{1/2} = b\ [\ln(2)]^{1/c}$$
$$x_{3/4} = b\ [\ln(4)]^{1/c}$$

Summarizing the characteristics of the Weibull probability density function we observe the following:

1. When b, the scale parameter, is held constant and the shape parameter c varies, the shape of the Weibull pdf changes from an exponential to Rayleigh to a Gaussian slightly skewed to right.
2. When c, the shape parameter, is held constant and the scale parameter b varies, the Weibull pdf gets sharper or flatter but no change in shape.

Last, we add the term CD to our review. The CD is defined as the ratio of the standard deviation to the first moment, the expectation:

$$CD = \frac{\text{standard of deviation}}{\text{first moment}} = \frac{\sigma_x}{E\{x\}} = \frac{\sigma_x^2}{E^2\{x\}} \tag{10.33}$$

The coefficient of dispersion was a key analytic tool in determining the number of Monte Carlo replications required in Chapter 9. The CD is a measure of how accurate an estimate is, Figure 10.12 illustrates the point.

The systems A and B have an equal means, E{x}, but different standard deviations. System A has a smaller standard deviation than B. Therefore, an estimate from system A has a higher accuracy than B. The systems C and D have an equal standard deviation but different first moment, E{x}. System C has a better estimate than D.

10.4.2 Weibull Clutter After a Square-Law Detector

When random variables pass through a square-law detector, the input probability density function transforms to a different pdf. We have seen the transformation of narrowband Gaussian noise to an exponential pdf.

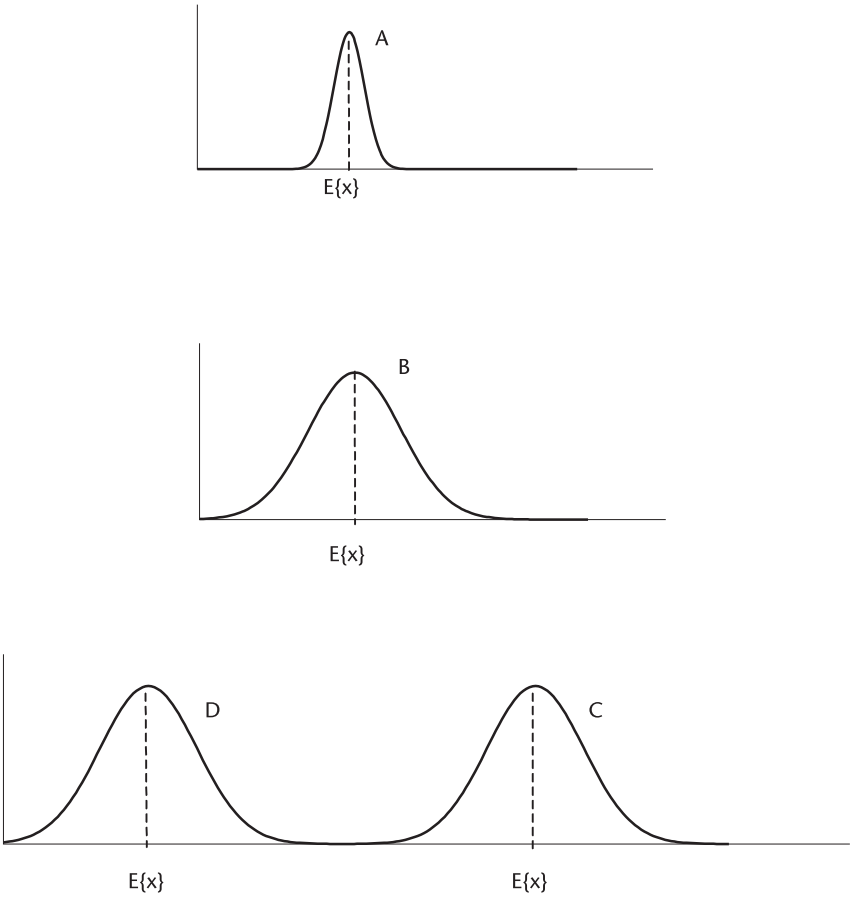
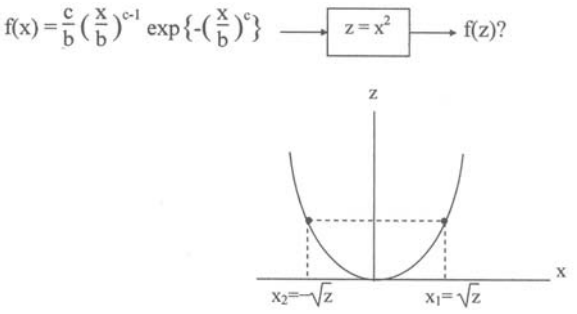


Figure 10.12 The coefficient of dispersion.

We shall determine the output pdf of a square-law detector when the input is a Weibull random variable.



The transformation of the pdf is given by [2].

$$
\begin{aligned}
f(z) &= \frac{f(x_1)}{|g'(x_1)|} + \frac{f(x_2)}{|g'(x_2)|}, \quad g'(x) = \frac{d}{dx} f(x) \\
&= \frac{1}{2\sqrt{z}} \frac{c}{b} \left(\frac{x_1}{b}\right)^{c-1} \exp\left\{-\left(\frac{x_1}{b}\right)^{c}\right\}\Big|_{x1=\sqrt{z}} \\
&= \frac{1}{2\sqrt{z}} \frac{c}{b} \left(\frac{\sqrt{z}}{b}\right)^{c-1} \exp\left\{-\left(\frac{\sqrt{z}}{b}\right)^{c}\right\} \\
&= \left(\frac{c/2}{b^2}\right) \left(\frac{z}{b^2}\right)^{c/2-1} \exp\left\{-\left(\frac{z}{b^2}\right)^{c/2}\right\}
\end{aligned}
\tag{10.34}
$$

The result indicates that the output pdf is another Weibull. The shape parameter c is halved, and the scale parameter b is squared. An interpretation of the result is that the input Weibull random variables is in amplitude and the output variable is in power; b is voltage, b^2 is power.

We have reviewed the characteristics of Weibull random variables, such as the probability density function f(x), probability distribution (cumulative) function F(x), the moments, the variance, mode and median, and quantile. We have reviewed the CD in regard to the accuracy of any estimate, given by a ratio of $\sigma_x/E\{x\}$.

We shall proceed to study CFAR processing in Weibull clutter.

10.5 Weber-Haykin CFAR (WH-CFAR)

Weber and Haykin have developed a CFAR processing method in Weibull clutter [13]. Their development is an extension of Rohling's OS-CFAR. The OS-CFAR originally reported applies when the input noise (or clutter) is a Gaussian with zero mean, an exponential after a square-law detector. The unknown to be estimated is noise power σ^2. For Weibull clutter we have two parameters to be estimated, the shape parameter c (or c/2) and the scale parameter b (or b^2).

We shall follow their analysis and present numerical results since they have not reported any. We start with the probability density function after a square-law detector.

$$
f(x) = \left(\frac{c/2}{b^2}\right) \left(\frac{x}{b^2}\right)^{c/2-1} \exp\left\{-\left(\frac{x}{b^2}\right)^{c/2}\right\} \tag{10.35}
$$

$$
F(x) = 1 - \exp\left\{-\left(\frac{x}{b^2}\right)^{c/2}\right\} \tag{10.36}
$$

From (10.36),

$$
x = b^2\{\ln\,[1 - F(x)]\}^{\,2/c} \tag{10.37}
$$

The false-alarm probability and the threshold are obtained by

$$P_{fa} = 1 - F(x)|_{x=Th} = \exp\left\{-\left(\frac{Th}{b^2}\right)^{c/2}\right\} \tag{10.38}$$

Thus,

$$Th = b^2[-\ln P_{fa}]^{2/c} \tag{10.39}$$

In order to compute the threshold Th of (10.39) we must know b and c or estimate them simultaneously. This is a task of two-parameters estimation.

Before we proceed further, we digress to demonstrate how sensitive the false probability P_{fa} is to an error in estimating c. In OS-CFAR processing the false alarm probability is given by (10.19). Rohling's noise pdf out of a square-law detector is an exponential. Weibull clutter out of a square-law detector is another Weibull with c halved. Equation (10.39) is modified accordingly.

$$P_{fa} = \frac{N!}{(N-k)!}\,\frac{\Gamma(T^{c/2} + N - k + 1)}{\Gamma(T^{c/2} + N + 1)} \tag{10.40}$$

Equation (10.40) is programmed in Pfa_OS_C.CPP, and the result is shown in Figure 10.13. The figure indicates that when c changes from 2.0 to 1.0, the false alarm probability changes almost four orders of magnitude. A large variation of P_{fa}

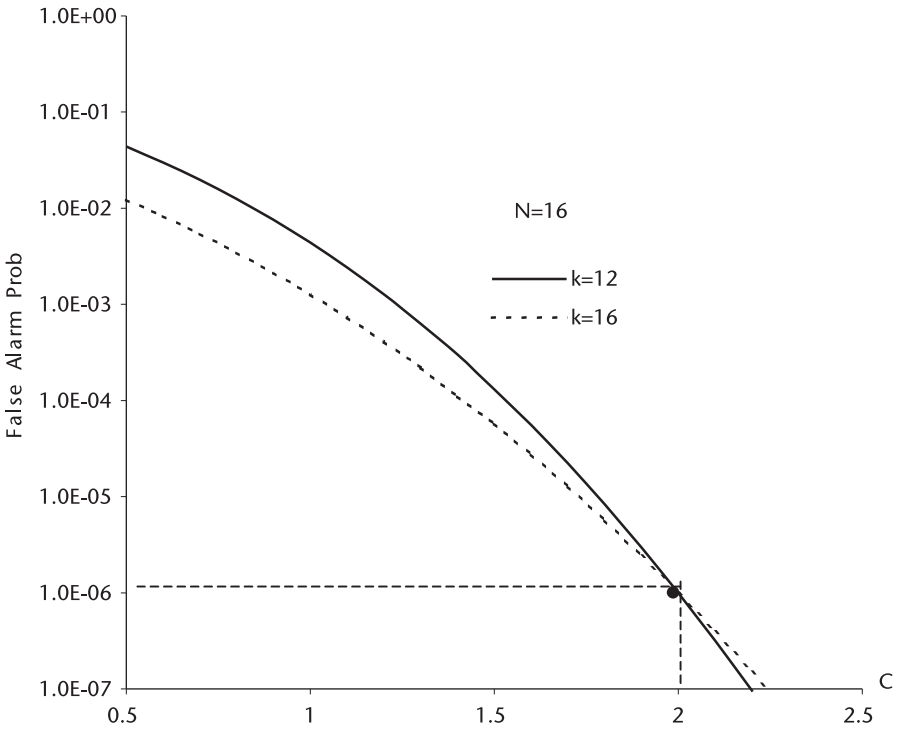


Figure 10.13 False alarm probability versus shape parameter c.

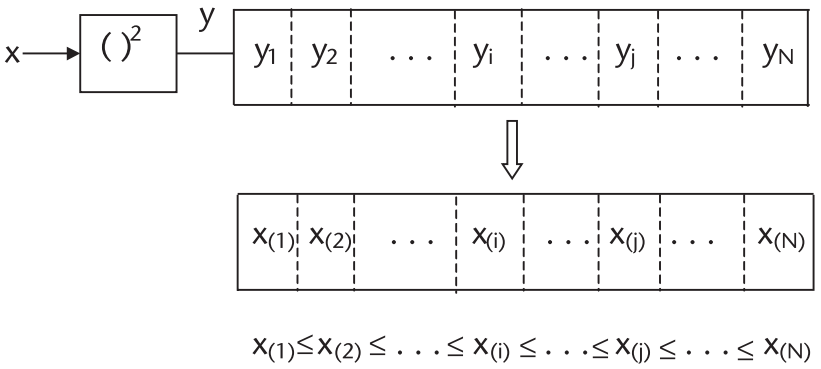


Figure 10.14 Weibull clutters are rank-ordered.

with respect to a small change in the shape parameter c is critical to all CFAR processing in Weibull clutter. Finn and Johnson [1] have also mentioned a large variation of P_{fa} with respect to a small change in σ_x of Gaussian noise. The importance of an accurate estimate of two parameters is emphasized here.

Back to the task of estimating the two parameters c and b. Suppose we have rank-ordered the Weibull clutter samples in the reference window as shown in Figure 10.14.

We take two clutters $x_{(i)}$ and $x_{(j)}$, i<j, out of the rank-ordered set and take the ratio of $x_{(i)}$ to $x_{(j)}$ according to (10.37). The scale parameter b will be cancelled. See Figure 10.15.

$$\frac{x_{(i)}}{x_{(j)}} = \frac{\{-\ln[1 - F(x_i)]\}^{2/c}}{\{-\ln[1 - F(x_j)]\}^{2/c}} = \left(\frac{k_i}{k_j}\right)^{2/c} \tag{10.41}$$

The identity at the right most is for a notational convenience as in Weber and Haykin. Taking the natural logarithm of both sides we obtain an estimate of c.

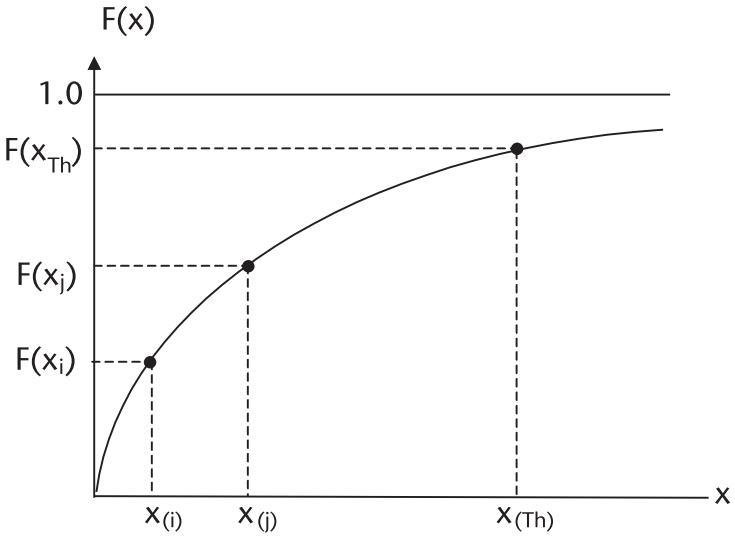


Figure 10.15 Selection of two rank-ordered clutters, and their cumulative values.

$$\ln\left[\frac{x_{(i)}}{x_{(j)}}\right] = \frac{2}{c}\ln\left(\frac{k_i}{k_j}\right)$$

$$\hat{c} = \frac{2\ln(k_i/k_j)}{\ln(x_i/x_j)}$$

$$= 2\frac{\ln\{-\ln[1-F(x_i)]\} - \ln\{\ln[1-F(x_j)]\}}{\ln(x_i) - \ln(x_j)} \tag{10.42}$$

Substitution of $\hat{c}$ into (10.37) yields an estimate of the scale parameter b.

$$\hat{b}^2 = \frac{x_i}{\{-\ln[1-F(x_i)]\}^{2/c}} = x_i(k_i)^{-2/\hat{c}} \tag{10.43}$$

and the estimate of the threshold Th is obtained from (10.39) by substituting the estimates of both b and c, after sliding $x_{(j)}$ to x_{Th} in Figure 10.15:

$$\begin{aligned}\hat{Th} &= \hat{b}^2\{-\mathrm{Ln}[1-F(Th)]\}^{2/\hat{c}}\\ &= x_i(k_i)^{-2/\hat{c}}\,(k_{Th})^{2/\hat{c}}\\ &= x_i\left(\frac{x_j}{x_i}\right)^{2/\hat{c}}\\ &= x_j^{1-2/\hat{c}}\,x_j^{2/\hat{c}}\end{aligned} \tag{10.44}$$

Thus,

$$Th = x_i^{1-\alpha}x_j^{\alpha}, \quad \alpha = 2/\hat{c} \tag{10.45}$$

The indices of i and j are transposed from Weber and Haykin's result, for they had rank-ordered the clutter in descending order whereas we have it ascending order. An estimate of the shape parameter $\hat{c}$ would be obtained from (10.42).

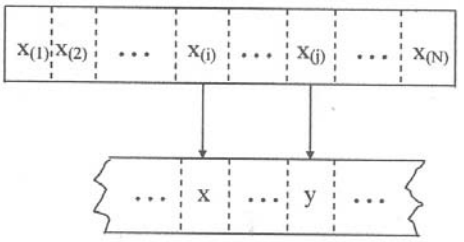
$$\alpha = 2/\hat{c} = \frac{\ln(x_i) - \ln(x_j)}{\ln\{-\ln[1-F(x_i)]\} - \ln\{\ln[1-F(x_j)]\}} \tag{10.46}$$

and, the false-alarm probability by substituting (10.44) into (10.38).

$$P_{fa} = \exp\left\{-\left(\frac{Th}{b^2}\right)^{c/2}\right\} = \exp\left\{-\left[(x_i^{1-\alpha}x_j^{\alpha})/\hat{b}^2\right]^{\hat{c}/2}\right. \tag{10.47}$$

Equation (10.47) is of no use unless we know the denominator of (10.46). The denominator is a random variable in x_i and x_j, as are α and Th. As Weber and Haykin pointed out, the threshold Th is a function of joint probability density function of x_i and x_j, $f_{xy}(x_i, y_j)$.

Figure 10.14 is redrawn for further discussion, and we designate $x_{(i)}$ and $x_{(j)}$ as x and y respectively for shortened notations for convenience.



David [7] and Bury [8] have provided the joint probability density function, rank-ordered in ascending order, x<y.

$$f_{xy}(x, y) = Q \times [F(x)]^{i-1}\ [F(y) - F(x)]^{j-i-1}\ [1 - F(y)]^{N-j}\ f_x(x)\ f_y(y) \qquad (10.48)$$

where

$$Q = \frac{N!}{(i-1)!(j-i-1)!(N-j)!}$$

The false-alarm probability P_{fa} is obtained by the following integral over proper integration limits.

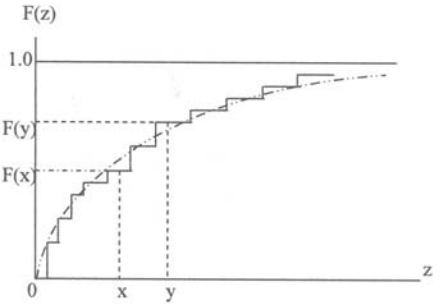
$$P_{fa} = \int_y \left[\int_x g_{xy}(x,y)dx \right] f_{xy}(x,y)\ dy \qquad (10.49)$$

where $g_{xy}(x,y)$ is the right-hand side of (10.47) and $f_{xy}(x,y)$ is (10.48). The proper limits can be visualized from the next figure.

Substituting $g_{xy}(x,y)$ and $f_{xy}(x,y)$ in (10.49), and change of variables yields an expression for P_{fa}:

$$u = \left(\frac{x}{b}\right)^{\hat{c}/2}\ ,\quad v = \left(\frac{y}{b}\right)^{\hat{c}/2}$$

$$\begin{aligned} P_{fa} = Q \cdot \int_{v1=0}^{v2=\infty} \int_{u1=0}^{u2=v} & \exp\{-(u^{1-\alpha}v^{\alpha})\}[1 - \exp\{-u\}]^{i-1} \\ & \times\ \exp\{-u\}\ [\exp\{-v\}]^{N-j+1} \\ & \times\ \exp\{-u\} - \exp\{-v\}]^{j-i+1}\ du\ dv \end{aligned} \qquad (10.50)$$



x can be from zero to y, but cannot be greater than y
y can be from zero to infinity, but cannot be less than x

The parameter α in (10.50) is computed by numerical integration in two dimensions after P_{fa} is set equal to the required value (e.g., 1.0E-6 or 1.0E-5). See Chapter 9 for a two-dimensional integration. Equation (10.50) is programmed in PFA_WH_U.CPP and PFA_WH_C.CPP, the former is for the case of uncensored, the latter for censored samples. The programs are executed with the following conditions:

Uncensored	*Censored*
N=16	N=16
i=3	i=3
j=16	j=12 (four highest clutters censored)
Q=21,841	Q=10,841,800

The parameter α is found iteratively until we obtained Pfa=1.0E-6 (or 1.0E-5). We have found,

α	P_{fa}=1.0E-6	P_{fa}=1.0E-5
Uncensored	2.510044	2.134437
Censored	5.931079	4.580292

The probability of detection for Swerling's target models 1 and 2 is given by (10.49) after a modification on derived by Shor-Levanon [14]. The integral is identical to (10.50) except that the first integrand, an exponential function, is multiplied by D, D=1/(1+SCR).

$$\begin{aligned} P_d = Q \cdot \int_{v1=0}^{v2=\infty} \int_{u1=0}^{u2=v} & \exp\{-D(u^{1-\alpha}v^{\alpha})\}[1-\exp\{-u\}]^{i-1} \\ & \times \exp\{-u\}\,[\exp\{-v\}]^{N-j+1} \\ & \times [\exp\{-u\}-\exp\{-v\}]^{j-i+1}\,du\,dv \end{aligned} \tag{10.51}$$

where Q is given by (10.48)

Equation (10.51) is programmed in PD_WH_U.CPP and PD_WH_C.CPP; U stands for the uncensored, C stands for the censored case. The result for the uncensored case is shown in Figure 10.16, in which we have added lines for no-CFAR and OS-CFAR for comparison. We note that WH-CFAR demands 10.55 dB higher SCR than OS-CFAR at $P_d = 0.5$ even though we have set N=16 and j=16 so that WH-CFAR has a minimum processing loss. We have given up the protection from the spurious targets interference temporarily.

In order to regain the protection we execute PD_WH_C.CPP, the censored version with N=16, i=3 and j=12; four highest clutters spikes (or four probable target returns) are censored. An additional increase of 18.6 dB in SCR is observed. The

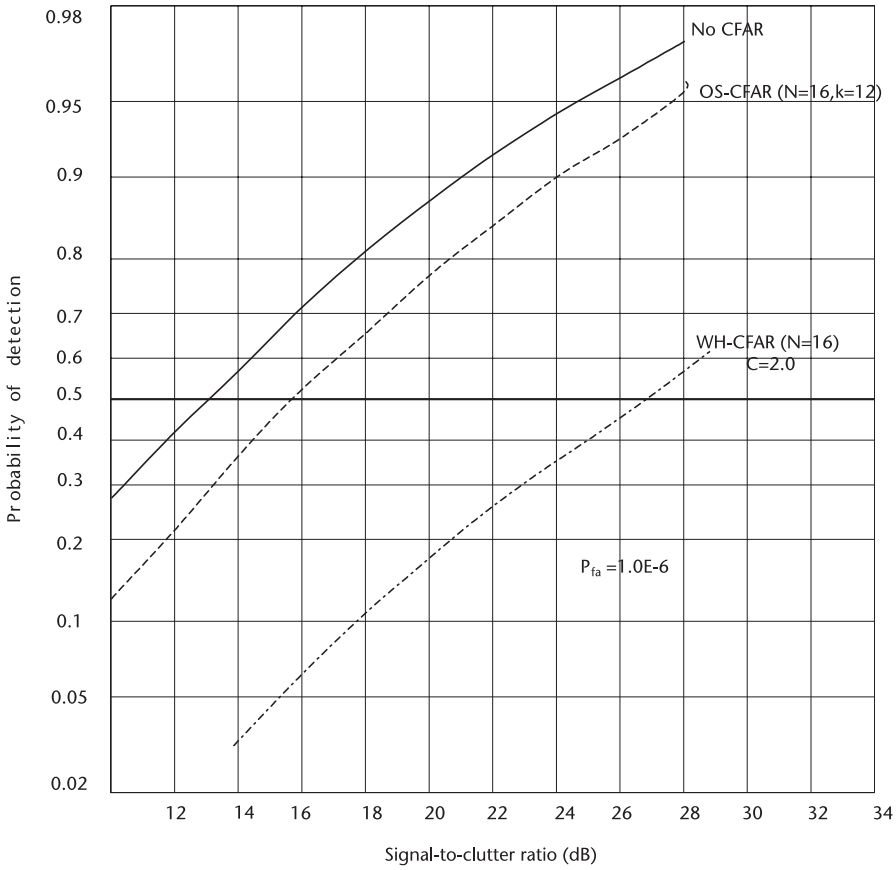


Figure 10.16 P_d versus SCR, WH-CFAR, uncensored.

censored WH-CFAR required 10.6+18.6=29.2 dB higher SCR at $P_d = 0.5$ for an equal protection.

The conclusion and remark by Weber and Haykin [13] and Shor and Levanon [14] are worth a serious consideration in system planning: "Should we estimate the two parameters c and b and pay a higher price for it?" They continue, "if the surveillance sector environments warrant that the minimum c is 1.5 or higher with high degree of certainty, the OS-CFAR with fixed c=1.5 suffers a lesser processing loss than WH-CFAR." On the other hand, the if c is 1.5 or less and varies from the sector to sector, then, there is no other choice but to estimate both parameters and accept the higher loss.

We have not explored other CFAR processing that might be better than WH-CFAR. We shall continue our search.

10.6 Maximum Likelihood CFAR (ML-CFAR)

Another CFAR processing method based on the maximum likelihood estimation is reported by Ravid and Levanon [15]. Their investigation is supported by several research papers on the estimation of Weibull parameters [16–19].

Estimation of two parameters simultaneously from samples, whether the samples are censored or uncensored is computationally very laborious and it is doubtful that real-time processing can be implemented; however, we follow their investigation.

The likelihood function of samples of independent and identically distributed Weibull random variables of population N is given by

$$L(x_i;c,b) = \left(\frac{c}{b^c}\right)^N \prod_{i=1}^{N} x_i^{c-1} \exp\left\{-\left(\frac{x_i}{b}\right)^c\right\} \tag{10.52}$$

We take the natural logarithms of both sides and a partial derivatives of $L(\cdot)$ with respect to c and b. The maximum likelihood estimate of the two parameters is obtained by equating each partial derivative to zero.

$$\hat{b} = \left[\frac{1}{N}\sum_{i=0}^{N} x_i^{\hat{c}}\right]^{1/\hat{c}} \tag{10.53}$$

$$1/\hat{c} = \left[\sum_{i=0}^{N} x_i^{\hat{c}} \ln x_i\right]\left[\sum_{i=0}^{N} x_i^{\hat{c}}\right]^{-1} - \frac{1}{N}\sum_{i=0}^{N} \ln x_i \tag{10.54}$$

The ML estimate of c is obtained from (10.54) through iterative procedure. More than a dozen iterations may be required depending upon the initial estimate of c. An estimate of b is then computed from (10.53) using the $\hat{c}$ just obtained. For a relatively small number of samples N=16, 18, . . . 32, the estimate of c is heavily biased. Bias is an unavoidable feature of most maximum likelihood estimators [8]. Thoman et al. [19] have published factors b[N] for removing the bias $\hat{c}$ when N is small.

N	*b[N]*	*N*	*b[N]*
10	0.859	42	0.968
12	0.883	44	0.970
14	0.901	46	0.971
16	0.914	48	0.972
18	0.923	50	0.973
20	0.931	54	0.975
22	0.938	58	0.977
24	0.943	62	0.979
26	0.947	66	0.980
28	0.951	70	0.981
30	0.955	75	0.983
32	0.958	80	0.984
34	0.960	85	0.985
36	0.962	90	0.986
38	0.964	100	0.987
40	0.966		

The unbiased estimate of $\hat{c}$ is obtained by multiplying $\hat{c}$ by b[N]. The corrected $\hat{c}$ will be used in (10.53) to compute an estimate of b.

The proposed ML processing is shown in Figure 10.17. In order to provide the protection from the interfering targets in the reference window, censoring the highest N–k cells is necessary as in OS-CFAR or WH-CFAR.

Estimates of b and c with censoring are derived by Bury [8], similar to (10.53) and (10.54) with additional terms.

$$\hat{b} = \left[\frac{1}{k}\sum_{i=0}^{N} x_i^{\hat{c}} + \left(\frac{N-k}{k}\right) x_i^{\hat{c}}\right]^{1/\hat{c}} \tag{10.55}$$

$$1/\hat{c} = \left[\sum_{i=0}^{N} x_i^{\hat{c}} \ln x_i + (N-k)\, x_i^{\hat{c}}\right]\left[\sum_{i=0}^{N} x_i^{\hat{c}} + (N-k)\, x_i^{\hat{c}}\right]^{-1} - \frac{1}{k}\sum_{i=0}^{N} \ln x_i \tag{10.56}$$

The computational burden imposed by (10.56) in estimating c through iterative procedure, but still biased, should discourage attempts to implement the processing in real time. Nonetheless we push forward to analyze the proposed ML processing in order to gain some learning experiences.

We analyze ML-CFAR in two steps. First we take up a simpler uncensored case in which the shape parameter c is assumed to be known exactly a priori. Only the

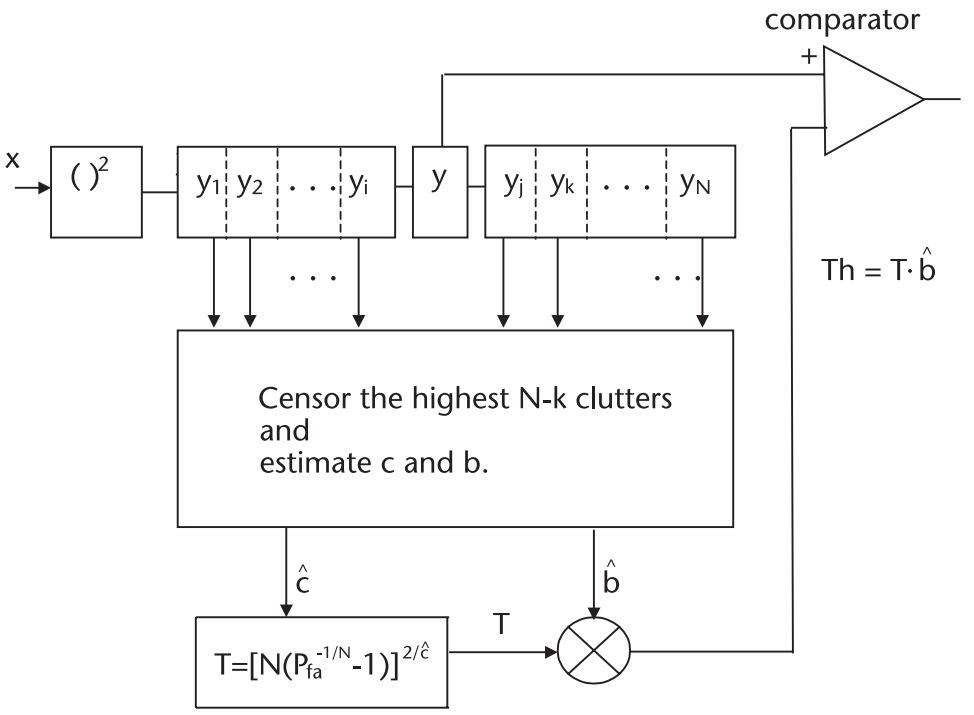


Figure 10.17 ML-CFAR processing.

scale parameter b is unknown, to be estimated by the maximum likelihood principle. First, we would rewrite (10.53) without the caret (^) on c.

$$\hat{b} = \left[\frac{1}{N}\sum_{i=0}^{N} x_i^c)\right]^{1/c} \tag{10.57}$$

The threshold Th is given by

$$Th = T \cdot \hat{b} = T \cdot \left[\frac{1}{N}\sum_{i=0}^{N} x_i^c)\right]^{1/c} \tag{10.58}$$

The false-alarm probability is

$$P_{fa} = \int_{z=0}^{\infty} \left[\int_{y=Tz}^{\infty} f_y(y_o)dy\right] f_z(z)dz \tag{10.59}$$

The integrand inside the bracket is the probability density function of Weibull clutter in the test cell, so that the integral is the probability of clutter that exceeds the threshold (false alarm). The integrand $f_z(z)$ is the probability density function of N samples of the parent Weibull clutter.

$$\int_{y=Tz}^{\infty} f_y(y_o)dy = F_y(y_o \geq Th) = 1 - F_y(Th) = \exp\left\{-\left(\frac{Th}{b}\right)^c\right\} \tag{10.60}$$

$$f_z(z) = \prod_{i=1}^{N}\left(\frac{c}{b}\right)\left(\frac{z_i}{b}\right)^{c-1}\exp\left\{-\left(\frac{z_i}{b}\right)^c\right\} \tag{10.61}$$

Substitution of (10.60) and (10.61) into (10.59) yields an expression for P_{fa} when the shape parameter c is exactly known a priori, and the scale parameter b is estimated by the maximum likelihood method.

$$P_{fa} = \int_{z=0}^{\infty} \exp\left\{-\left(\frac{Th}{b}\right)^c\right\}\prod_{i=1}^{N}\left(\frac{c}{b}\right)\left(\frac{z_i}{b}\right)^{c-1}\exp\left\{-\left(\frac{z_i}{b}\right)^c\right\}\, dz \tag{10.62}$$

Using a change of variable further substitution of (10.58) and integrating we have a compact simple expression for P_{fa}.

$$\text{Let } \left(\frac{z_i}{b}\right)^c = t_i, \text{ then } dz_i = \left(\frac{b}{c}\right)(t_i)^{1/c-}$$

$$P_{fa} = \frac{1}{[1 + T^c/N]} \tag{10.63}$$

So far we have assumed an envelope detector (a linear detector). For a square-law detector we replace c by c/2.

$$P_{fa} = \frac{1}{[1 + T^{c/2}/N]} \tag{10.64}$$

Equation (10.64) shows that the probability of false alarm is independent of b, provided the shape parameter c is known exactly a priori. T is the threshold multiplier, not the threshold.

The multipliers T for linear and a square-law detectors are

$$T_1 = [N(P_{fa})^{-1/N} - 1)]^{1/c} \quad \text{linear detector} \tag{10.65a}$$

$$T_s = [N(P_{fa})^{-1/N} - 1)]^{2/c} \quad \text{square-law detector} \tag{10.65b}$$

Finally, the thresholds Th_l and Th_s are given by

$$Th_1 = T_1 \cdot \hat{b} = [N(P_{fa})^{-1/N} - 1)]^{1/c} \left[\frac{1}{N}\sum_{i=0}^{N} x_i^c)\right]^{1/c} \tag{10.66a}$$

$$Th_s = T_s \cdot \hat{b} = [N(P_{fa})^{-1/N} - 1)]^{2/c} \left[\frac{1}{N}\sum_{i=0}^{N} x_i^{c/2})\right]^{2/c} \tag{10.66b}$$

Equation (10.65b) is programmed in ML_MULTI.CPP, and the result is shown below. Note a sharp rise of the multipliers when c<1.5 for all N.

N	*c=2.0*	*c=1.5*	*c=1.0*	*c=0.5*
12	25.9473	76.8169	673.2649	4.5328E5
14	23.5577	67.5323	554.9672	3.0799E5
16	21.9420	61.4282	481.4505	2.3179E5
18	20.7798	57.1289	431.8011	1.8645E5
20	19.9052	53.9457	396.2188	1.5699E5
22	19.2240	51.4981	369.5615	1.3658E5
24	18.6787	49.5597	348.8940	1.2172E5
26	18.2326	47.9879	332.4281	1.1050E5
28	17.8610	46.6883	318.0162	1.0177E5
30	17.5468	45.5964	307.8900	0.9479E5
32	17.2776	44.6663	298.5172	0.8911E5

The detection probability is given by an identical form to (10.59) except that the integrand inside the square bracket represents the probability density function of the target return plus Weibull clutter in the test cell.

$$P_d = \int_{z=0}^{\infty} \left[\int_{y=Th}^{\infty} f_y(y_1)\, dy \right] f_z(z)\, dz \tag{10.67}$$

The probability density function $f_y(y_1)$ must be the vectorial sum of target return signal and Weibull clutter:

$$f_y(y_1) = \text{pdf of (target signal + Weibull clutter)}$$

The "+" sign signifies a vectorial sum. We have failed to obtain an analytically closed form of the probability density function for $f_y(y_1)$. (The Monte Carlo technique is suggested here.) Absence of an analytic expression forces us to choose a next best plausible alternative. We propose that the statistics of the test cell is an algebraic sum of the mean of Weibull clutter given by (10.28) with c replaced by c/2 and Swerling's target model 1 or 2.

For notational clarity and further discussion, see Figure 10.18. We plan to modify Swerling's target models.

$$\begin{aligned} f_y(y_1) &= \frac{1}{\text{sig} + \sigma_y + b_y} \exp\left\{\frac{-y}{\text{sig} + \sigma_y + b_y}\right\} \\ &= \frac{1}{b_y[1 + \sigma_y/b_y + \text{SCR}]} \exp\left\{\frac{-y}{b_y[1 + \sigma_y/b_y + \text{SCR}]}\right\} \\ &\approx \frac{1}{b_y[1 + \text{SCR}]} \left\{\exp\frac{-y}{b_y[1 + \text{SCR}]}\right\} \quad (\sigma_y/b_y <<< 1) \\ &= \frac{1}{\sigma_y\Gamma(1 + 2/c)[1 + \text{SCR}]} \exp\left\{\frac{-y}{\sigma_y\Gamma(1 + 2/c)[1 + \text{SCR}]}\right\} \end{aligned} \tag{10.68}$$

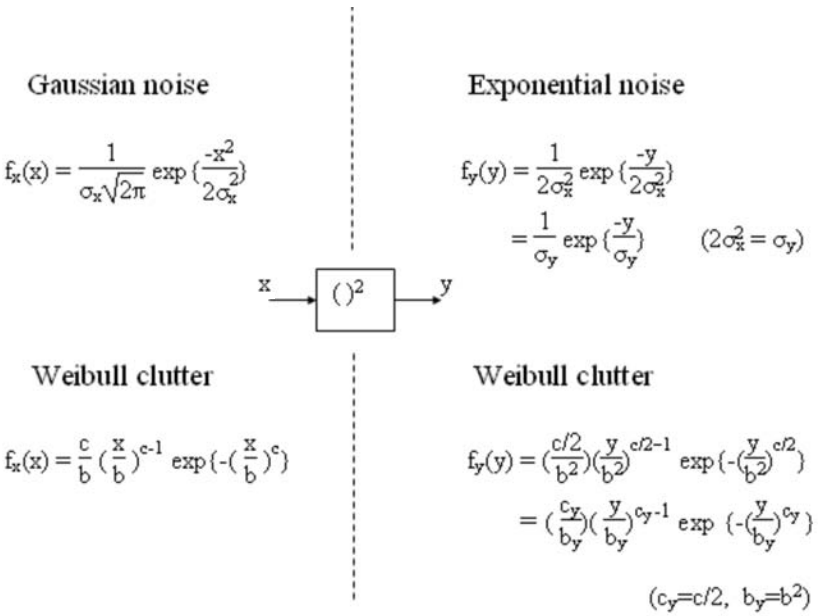


Figure 10.18 Probability density functions before and after a square-law detector.

The bracketed term of (10.67) is directly integrated.

$$\int_{y=Th}^{\infty} f_y(y_1)dy = \int_{y=Th}^{\infty} \frac{1}{\sigma_y\Gamma(1+2/c)[1+SCR]} \exp\left\{\frac{-y}{\sigma_y\Gamma(1+2/c)[1+SCR]}\right\} dy \tag{10.69}$$

$$= \exp\left\{\frac{-Tz}{\sigma_y\Gamma(1+2/c)[1+SCR]}\right\}, \quad Th = Tz$$

The $f_z(z)$ is the probability density function of the sum of N Weibull clutter samples given by Harter and Moore [17].

$$f_z(z) = \frac{N^N}{\Gamma(N)}\left(\frac{c/2}{b^2}\right)\left(\frac{z}{b^2}\right)^{cN/2-1} \exp\left\{-N\left(\frac{z}{b^2}\right)^{c/2}\right\} \tag{10.70}$$

Substituting (10.69) and (10.70) into (10.67), the detection probability P_d is expressed as

$$P_d = \int_{z=0}^{\infty} \exp\left\{\frac{-Tz}{\sigma_y\Gamma(1+2/c)[1+SCR]}\right\}$$

$$\cdot \frac{N^N}{\Gamma(N)}\left(\frac{c/2}{b^2}\right)\left(\frac{z}{b^2}\right)^{cN/2-1} \exp\left\{-N\left(\frac{z}{b^2}\right)^{c/2}\right\} dz$$

The following changes of variable and identity

$$N\left(\frac{z}{b_y}\right)^{c/2} = t,\ z = b_y\left(\frac{t}{N}\right)^{2/c-1},\ dz = b_y\left(\frac{1}{N}\right)\left(\frac{t}{N}\right)^{2/c-1},\ \text{and}\ b_y = \sigma_y\Gamma(1+2/c)$$

are used to obtain the expression for P_d, after a tedious but straightforward algebraic manipulation.

$$P_d = \frac{1}{\Gamma(N)} \int_{t=0}^{\infty} t^{N-1} \exp\left\{-t - \frac{T}{[1+SCR]}\left(\frac{t}{N}\right)^{2/c}\right\} dt \tag{10.71}$$

Equation (10.71) is programmed in PdMLCFAR.CPP with triple conditions; P_{fa}=1.0E-6, N=16, and c=2.0 (or c=1.5, or c=1.0, or c=0.5). The results are shown below and in Figure 10.19.

SCR(dB)	*c=2.0*	*c=1.5*	*c=1.0*	*c=0.5*
14.0	0.4410	0.1202		
16.0	0.5893	0.2440		
18.0	0.7127	0.3974		
20.0	0.8059	0.5508	0.0344	

22.0	0.8720	0.6823	0.0914	
24.0	0.9169	0.7838	0.1916	
26.0	0.9466	0.8567	0.3281	
28.0	0.9659	0.9066	0.4784	
30.0	0.9783	0/9366	0.6183	
32.0	0.9863	0.9616	0.7334	
34.0	0.9913	0.9756	0.8200	
36.0	0.9945	0.9845	0.8813	
38.0	0.9965	0.9902	0.9229	
40.0	0.9978	0.9938	0.9505	
42.0	0.9986	0.9961	0.9684	0.0175
44.0	0.9991	0.9975	0.9799	0.0374

The very high values of threshold multipliers T in (10.65b) when c=1.0 and c=0.5 are reflected in SCR above. We should remind ourselves that the results are without protection from spurious target intrusions in the reference window and that the shape parameter c is assumed to be known exactly a priori. When the highest clutter samples (or targets) are censored, N-k, k<N, we suffer additional loss, computed by,

$$\begin{aligned}
\text{Additional loss}_{(\text{dB})} &= 10 \log \left[\frac{\text{Th} - \text{censored}}{\text{Th} - \text{uncensored}} \right] \\
&= 10 \log \left[\frac{\text{Eq 10.66b: N} = \text{k}}{\text{Eq 10.66b}} \right] \\
&= 10 \log \left[\frac{\text{k}}{\text{N}} \cdot \frac{(\text{P}_{\text{fa}}^{-1/\text{k}} - 1)}{(\text{P}_{\text{fa}}^{-1/\text{N}} - 1)} \right]^{2/\text{c}} \\
&= \left(\frac{2}{\text{c}} \right) 10 \log \left[\frac{\text{k}}{\text{N}} \cdot \frac{(\text{P}_{\text{fa}}^{-1/\text{k}} - 1)}{(\text{P}_{\text{fa}}^{-1/\text{N}} - 1)} \right]
\end{aligned}$$

For P_{fa} = 1.0E-6, N=16, k=12, additional losses are

$$\begin{aligned}
\text{c} = 2.0 \quad & 1.93 \text{ dB} \\
\text{c} = 1.5 \quad & 2.64 \text{ dB} \\
\text{c} = 1.0 \quad & 3.96 \text{ dB} \\
\text{c} = 0.5 \quad & 7.91 \text{ dB}
\end{aligned}$$

P_d vs SCR, shown in Figure 10.19 (uncensored) would slide further to the right by the values shown above. Figure 10.19, uncensored ML-CFAR should be compared with Figure 10.16, for uncensored WH-CFAR.

In the next section we explore an alternative method in estimating c without the iterations of (10.54) or (10.56).

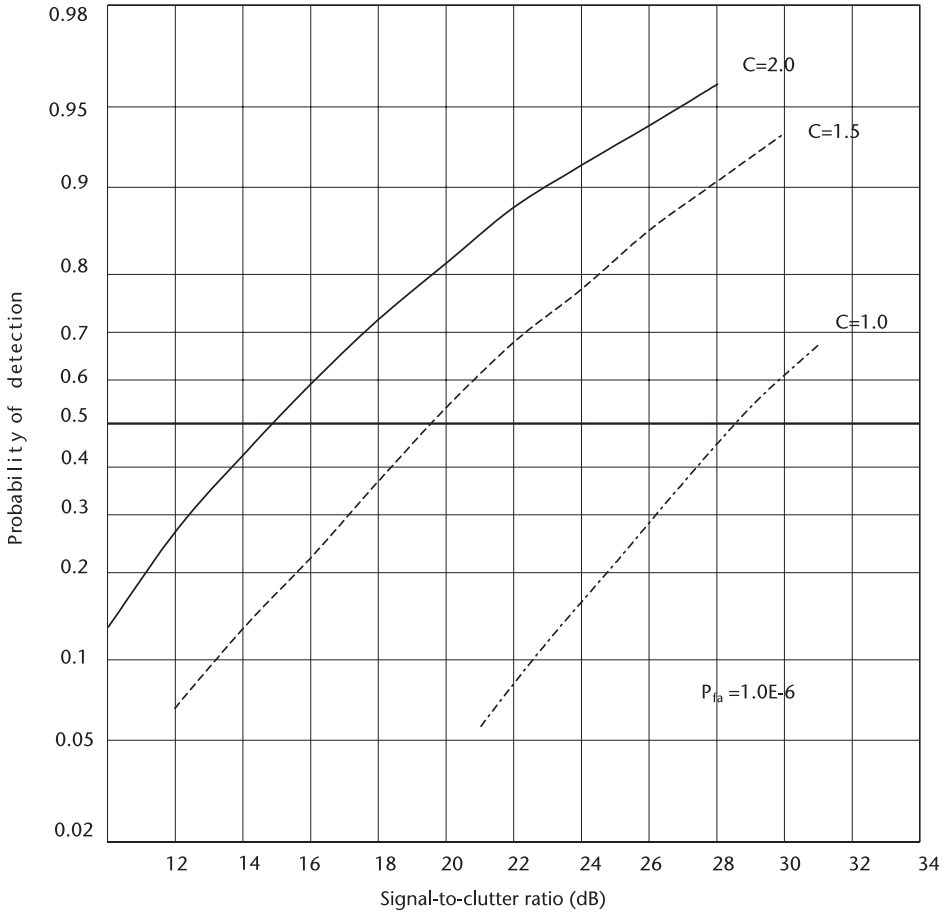


Figure 10.19 P_d versus SCR, ML-CFAR, uncensored, N=16.

10.7 Minimum Mean Square Error CFAR (MMSE-CFAR)

There appears to be an alternative for estimating the shape parameter c without iterations, as some researchers have suggested. We take the ratio of the first moment to the square-root of the second moment of Weibull random variable. We note that the scale parameter b is cancelled.

$$E\{x\} = \int_{x=0}^{\infty} x\, f_x(x)\, dx = b^2\Gamma(1 + 2/c)$$

$$E\{x^2\} = \int_{x=0}^{\infty} x^2\, f_x(x)\, dx = b^4\Gamma(1 + 4/c)$$

$$z = \frac{E\{x\}}{[E\{x^2\}]^{1/2}} = \frac{\Gamma(1 + 2/c)}{[\Gamma(1 + 4/c)]^{1/2}} \tag{10.72}$$

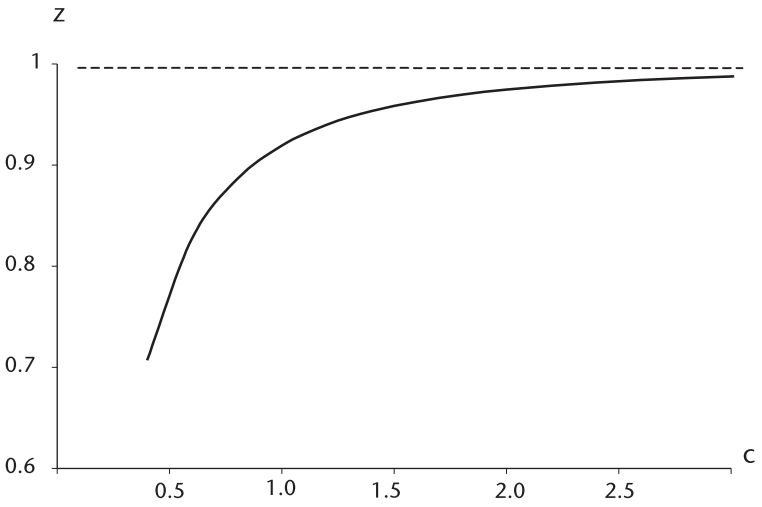


Figure 10.20 $E\{x\}/[E\{x^2\}]^{1/2}$ versus shape parameter c.

Equation (10.72) is shown in Figure 10.20. When we know the value of z we are able to compute the shape parameter by an inverse relationship.

It had been suggested that an estimate of c may be obtained by the process depicted in Figure 10.21.

We have found that the ratio z is heavily biased and that the variance is very large when the number of samples is relatively small. The coefficient of dispersion we discuss in Section. 10.4.1 gives us a clue to the poor estimate.

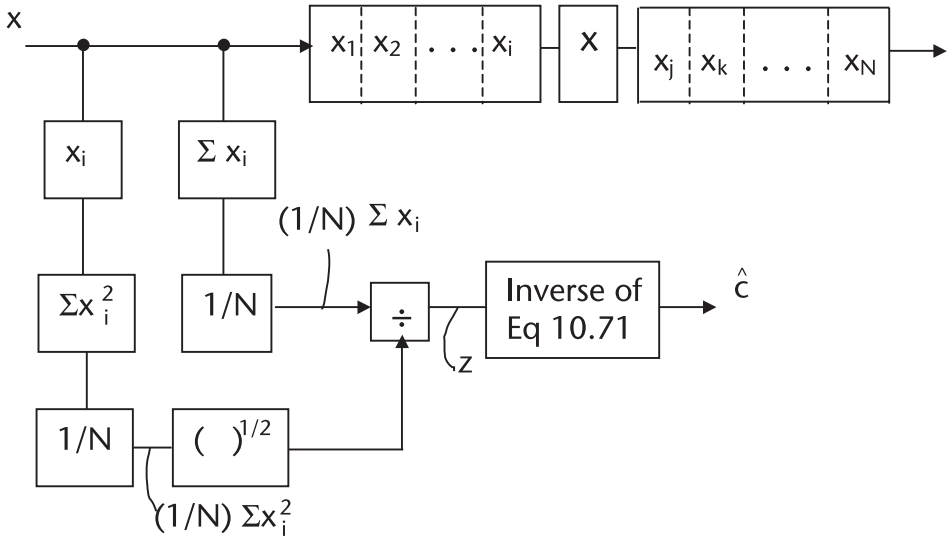


Figure 10.21 Estimate of parameter c.

$$\begin{aligned} \mathrm{CD} &= \frac{\sigma_x}{E\{x\}} = \frac{[\Gamma(1+2/c) - \Gamma(1+1/c)]^{1/2}}{\Gamma(1+1/c)} \\ &= \left[\frac{\Gamma(1+2/c)}{\Gamma^2(1+1/c)} - 1\right] \end{aligned} \tag{10.73}$$

Equation (10.73) is shown in Figure 10.22, which indicates that estimating the shape parameter by (10.7.1) is a very risky proposition throughout the range 0.5<C<2.5. CD is simply too large.

Let us explore a second alternative for avoiding the laborious iterations. We start with the probability distribution function.

We proceed with analysis is done with a linear detector. All the results can easily be converted to a square-law detection by replacing c with c/2, and replacing b with b^2.

$$F_y(y) = \int_{y=0}^{\infty} f_y(y)\, dy = 1 - \exp\left\{-\left(\frac{y}{b}\right)^c\right\} \tag{10.74}$$

Taking the natural logarithm of both sides twice we have,

$$\ln\{-\ln[1 - F_y(y)]\} = c \ln y - \ln b^c \tag{10.75}$$

Equation (10.75) can be interpreted as a straight line, as shown in Figure 10.23.

$$z = cx + \text{constant}$$

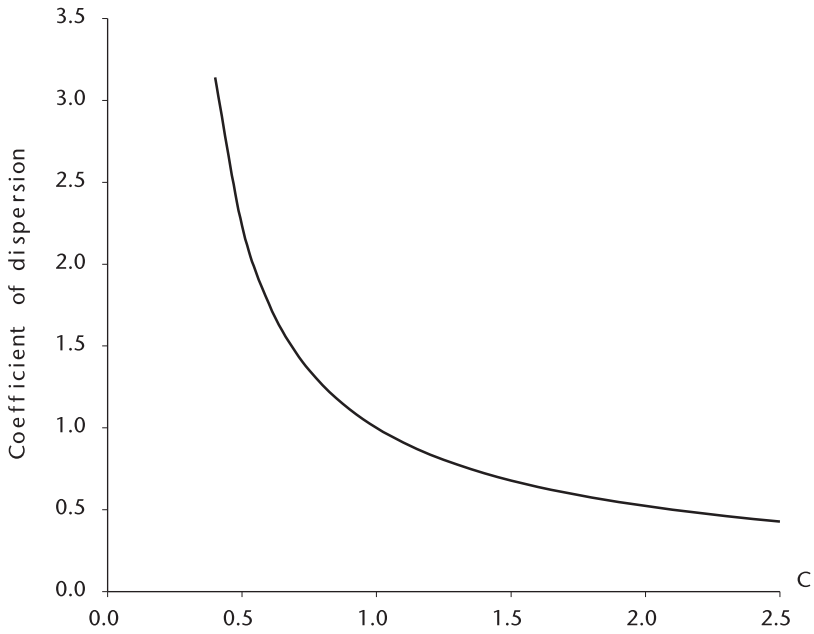


Figure 10.22 Coefficient of dispersion versus c.

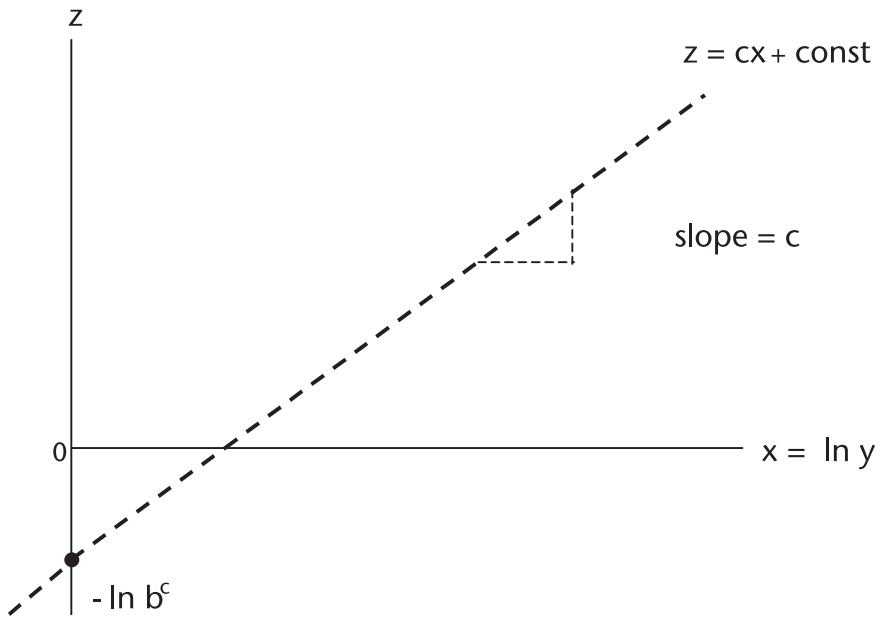


Figure 10.23 Linear equation representation of Weibull distribution.

where

z: $\ln \{-\ln[1-F_y(y)]\}$;
c: The slope of a straight line;
x: $\ln y$;
$\ln b^c$: Constant, intercept point.

How to determine the left-hand side of (10.75) which we have designated as z? Suppose we are given N Weibull random variables, N clutter samples. We rank-order the clutter samples in ascending order in magnitude.

$$y_{(1)} \leq y_{(2)} \leq \ldots \leq y_{(i)} \leq \ldots \leq y_{(N)}$$

so that we have

$$z_{(i)} = c \ln(y_{(i)}) + \ln b^c \tag{10.76}$$

We choose c and b to minimize the sum of squared error of (10.76).

$$\text{minimize} \sum_{i=1}^{N} [z_{(i)} - c \ln(y_{(i)}) - \ln b^c]^2 \tag{10.77}$$

This is the MMSE principle, or the MMSE estimate of the regressive parameters c and b. A scatter plot of rank-ordered clutter samples appear as shown in Figure 10.24.

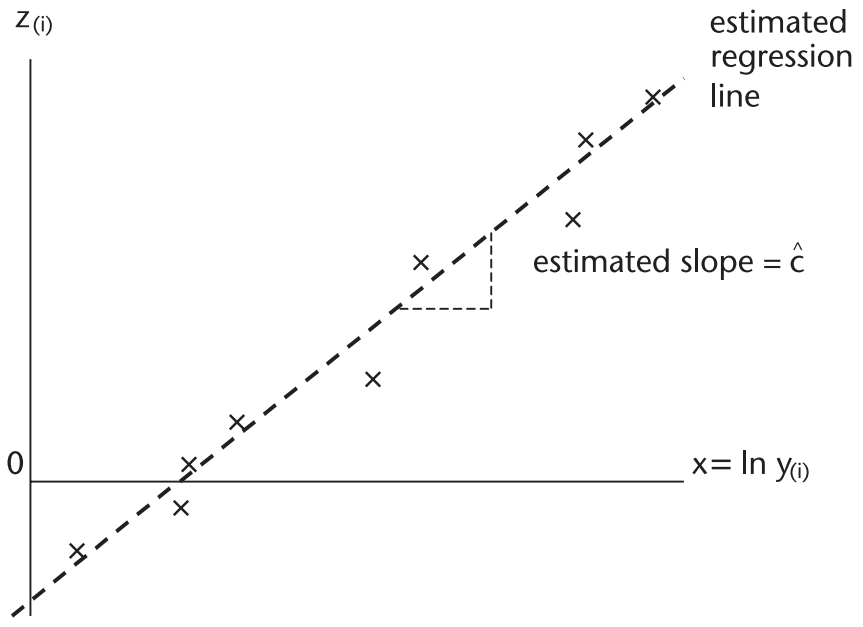


Figure 10.24 Scatter plot and regression line.

The minimization of (10.77) is attained when c and b are estimated to be,

$$\hat{c} = \frac{\sum_{i=1}^{N} z_{(i)} \ln y_{(i)} - N < \ln y > < z >}{\sum_{i=1}^{N} [\ln y_i]^2 - N[< \ln y >]^2} \tag{10.78}$$

$$\hat{b} = [\exp\{- <z> + \hat{c} <\ln y>\}]^{1/\hat{c}} \tag{10.79}$$

where

$z_{(i)}$ and $y_{(i)}$ are rank-ordered pairs;
$<\ln y> = \frac{1}{N}\sum \ln y_i$, average of natural logarithm of y_i;
$<z> = \frac{1}{N}\sum z_i$, average of z_i.

Equation (10.75) is rewritten in sampled form as follows.

$$\ln \{- \ln[1 - F_y(y_i)]\} = c \ln y_i - \ln b^c$$

In order to estimate c and b through MMSE regression we must know $F_y(y_i)$ given y_i, i=1, 2, 3, . . . , N. An estimate of $F_y(y_i)$ is given by Bury [8] and Ross [20].

$$E\{F_y(y_i)\} = \frac{i}{N+1} \tag{10.80}$$

so that

$$\begin{aligned}\hat{z}_i &= \ln\{-\ln[1 - E\{F_y(y_i)\}]\} \\ &= \ln\left\{-\ln\left[1 - \frac{i}{N+1}\right]\right\} \\ &= \ln\left\{-\ln\left[\frac{N+1-i}{N+1}\right]\right\}\end{aligned} \tag{10.81}$$

An estimate $\hat{z}_i$ obtained from (10.81) is used in (10.78) and (10.79). A program is written in WBL_MMSE.CPP to check out the estimate of c and b. The program proves it can be implemented for a on-line, real-time operation.

MMSE-CFAR processing is shown in Figure 10.25. Two parameters are changed after a square-law detector. The censoring of the clutter samples is simply replacing N by k, after the rank ordering. MMSE-CFAR is the most robust approach to the Weibull environment. It does not require an iterative procedure, a heavy computation load of doubtful real-time processing.

The two parameters after a square-law detector are the modified forms of (10.78) and (10.79):

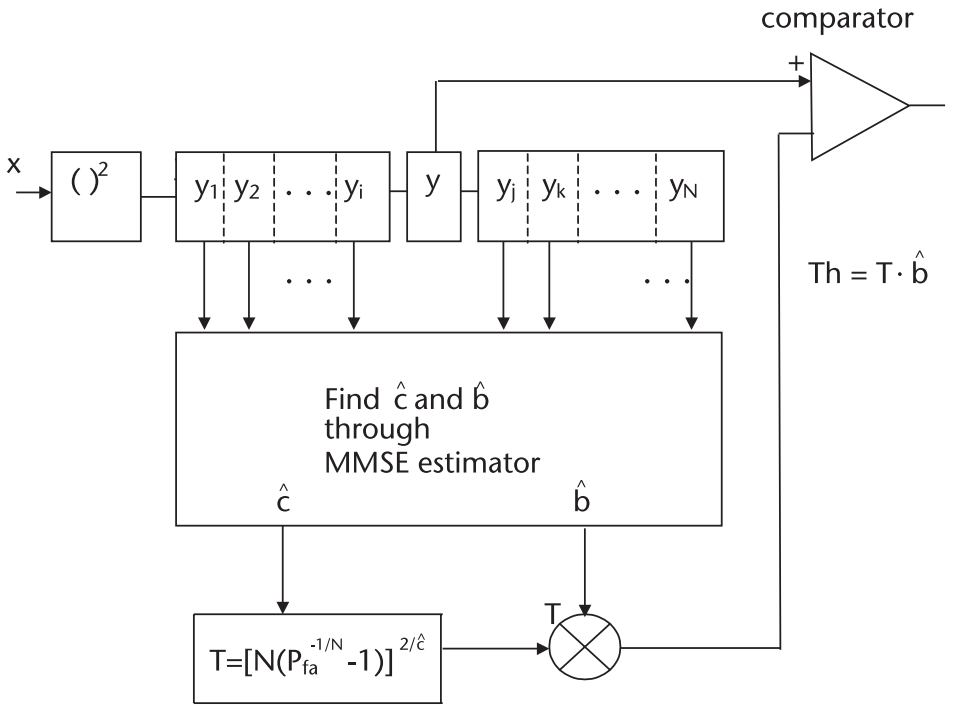


Figure 10.25 MMSE-CFAR processing.

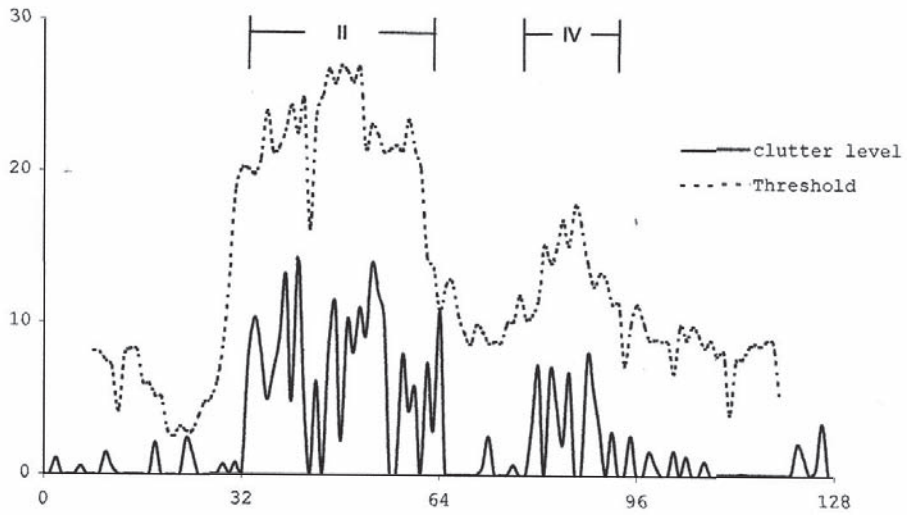


Figure 10.26 Weibull clutter and threshold level.

$$\hat{c}/2 = \frac{\sum_{i=1}^{N} z_{(i)} \ln y_{(i)} - N \langle \ln y \rangle \langle z \rangle}{\sum_{i=1}^{N} [\ln y_i]^2 - N[\langle \ln y \rangle]^2} \tag{10.82}$$

$$\hat{b}^2 = [\exp\{- \langle z \rangle + \hat{c}/2 \langle \ln y \rangle\}]^{2/\hat{c}} \tag{10.83}$$

We have removed the assumption in ML-CFAR analysis that "the shape parameter c is known a priori". The two parameters are obtained by the MMSE technique given by (10.81)–(10.83). The detection curves in Figure 10.19 remain correct without any assumption. The clutter-edge response and the threshold are shown in Figure 10.26. The scenario of clutter level profile is similar (but not identical) to CA-CFAR and OS-CFAR. We have chosen C=1.0 in the region II, and c=0.75 in the region IV to show the spiky clutters. The clutter power levels are equal to those of CA-CFAR and OS-CFAR, however. The clutter power is the sum of the mean squared and the variance:

$$\begin{aligned} \text{Clutter power} &= b^2\,\Gamma(1+2/c) \quad \text{liner detector} \\ &= b^4\,\Gamma(1+4/c) \quad \text{square-law detector} \end{aligned}$$

10.8 Conclusion

We have barely scratched the surface of the subject of CFAR processing. Many potential techniques to achieve better solutions against clutter disturbances remain unexplored.

List of Programs

Program	*Features*
(1) CA_MULTI.CPP	Computes the threshold multiplier T of CA-CFAR
(2) PdCACFAR.CPP	Computes the detection probability P_d vs SNR of CA-CFAR
(3) PdNoCFAR.CPP	Computes the detection probability P_d when the number of reference cells is infinity
(4) CA_LOSS.CPP	Computes CA-CFAR loss in decibels versus the number of reference cells
(5) CA_CFAR.CPP	Clutter-edge response and threshold level of CA-CFAR processing
(6) OS_MULTI.CPP	Computes the threshold multiplier T of OS-CFAR
(7) OS_THRES.CPP	Computes the threshold Th of OS-CFAR
(8) OS_LOSS.CPP	Computes loss in decibels in OS-CFAR processing
(9) Pd_OS_Nk.CPP	Computes detection probability of OS-CFAR, the number of reference cells is N; the rank-ordered cell is k
(10) OS_CFAR.CPP	Clutter-edge response and threshold of OS-CFAR
(11) Pfa_WH_U.CPP	Computes the false alarm probability of Weber-Haykin CFAR, clutter uncensored
(12) Pfa_WH_C.CPP	Computes the false alarm probability of Weber-Haykin CFAR, clutter censored
(13) Pd_WH_U.CPP	Computes the detection probability, WH-CFAR, clutter uncensored
(14) Pd_WH_C.CPP	Computes the detection probability, WH-CFAR, clutter censored
(15) ML_MULTI.CPP	Computes the threshold multiplier T of maximum likelihood CFAR, ML-CFAR
(16) PdMLCFAR.CPP	Computes the detection probability, ML-CFAR
(17) WBL_MMSE.CPP	Estimates two parameters of Weibull clutter
(18) SORT_X.H	by the MMSE principle; a header file for sorting data
(19) MSE_CFAR.CPP	Clutter-edge response and threshold,
(20) SORT_Y.H	MMSE-CFAR; a header file sorting data
(21) SIMPSONZ.H	A header file for function integration in PdMLCFAR.CPP

References

[1] Finn, H. M., and R. S. Johnson, "Adaptive Threshold Mode with Threshold Control as a Function of Spatially Sampled Clutter Level Estimates," *RCA Review*, Vol. 9, No. 3, Sept. 1968.

[2] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, New York, NY: McGraw-Hill, 1965.

[3] Whalen, A. D., *Detection of Signals in Noise*, New York, NY: Academic Press, 1971.

[4] Burdic, W. S., *Radar Signal Analysis*, Englewood Cliff, NJ: Prentice-Hall, 1968.

[5] Rubinstein, R. Y., *Simulation and the Monte Carlo Method*, New York, NY: John Wiley & Sons, 1981.

[6] Rohling, H., "Radar CFAR Thresholding in Clutter and Multiple Target," *IEEE, Trans.*, Vol. AES-19, No. 4, July 1983.

[7] David, H. S., *Order Statistics*, New York, NY: John Wiley & Sons, 1961.

[8] Bury, K.V., *Statistical Models in Applied Sciences*, New York, NY: John Wiley & Sons, 1975.

[9] Abramowitz, M., and I. Stegun, *Handbook of Mathematical Functions with Formula, Graphs and Mathematical Tables*, Nat'l Bureau of Standards, series 55, 1965, Eq. [6.2.1] and Eq. [6.2.2].

[10] Weibull, W., *A Statistical Theory of the Strength of Materials*, Stockholm, Sweden: Ing. Vetenskaps Akademy, 1939.

[11] Boothe, R. R., *The Weibull Distribution Applied to the Ground Clutter Back-scatter Coefficient*, U.S. Army Missile Command, Report No. RE-TR-69-15, AD 691-109, June 1969.

[12] Sekine, Mo. and Mao, Y. H., *Weibull Radar Clutter*, London, UK: IEE, Peter Peregrinus, 1990.

[13] Weber, P., and S. Kaykin, "Ordered Statistic CFAR Processing for Two-Parameter Distributions with Variable Skewness," *IEEE Trans.,* Vol. AES-21, No. 6, Nov. 1985.

[14] Shor, M., and N. Levanon, "Performance of Order Statistics CFAR," *IEEE Trans.*, Vol. AES-27, No. 2, Mar. 1991.

[15] Ravid, R., and N. Levanon, "Maximum Likelihood CFAR for Weibull Background," *IEE Proceedings (London)*, 139, Pt. F3, June 1992.

[16] Cohen, A. C., Jr., "Maximum Likelihood Estimation in the Weibull Distribution Based on Complete and Censored Samples," *Technometrics*, Vol. 7, No. 4, 1965.

[17] Harter, H. L., and A. H. Moore, "Maximum Likelihood Estimation of Parameters of Gamma and Weibull Population from Complete and Censored Samples," *Technometrics*, Vol. 7, No. 4, 1965.

[18] Dubey, J. N., "Some Percentile Estimates for Weibull Parameters," *Technometrics*, Vol. 9, 1967.

[19] Thoman, D. R., L.J. Bain, and C.E. Antis, "Inferences on the Parameters of Weibull Distribution," *Technometrics*, Vol. 11, 1969.

[20] Ross, S. M. *Introduction to Probability and Statistics for Engineers and Scientists*, New York, NY: John Wiley & Sons, 1987.

Selected Bibliography

Oberhettinger, F., and L. Baddi, *Tables of Laplace Transforms*, New York, NY., Springer-Verlag, 1970, [Eq. 2.5].

CHAPTER 11

Moving Target Indicator

11.1 Introduction

The object of moving target detection and indication is to reject returned signals from stationary objects such as buildings, hills, and islands and detect the signal from moving targets such as airborne aircrafts or maneuvering surface ships on the sea. Figure 11.1 illustrates the phase shifts of returned signals from a moving target and a constant phase from a stationary target.

A simplified receiver structure that detects the phase shift from a moving target is depicted and the waveform and response of a delay-line canceller is shown in Figure 11.2.

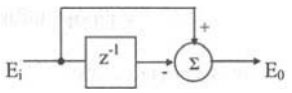
The output of the phase detector is applied to a delay-line canceller. The previous return is subtracted from the present return. The output of the canceller is zero for a stationary target or clutter whereas it is nonzero for a moving target.

We call the phase detector plus delay-line canceller an moving target indicator (MTI) filter. The desired frequency response of an MTI filter relative to the spectrum of stationary clutter is shown in Figure 11.3.

Many techniques are available to realize the delay-line canceller. The canceller may have a nonrecursive or recursive structure. There are cancellers that have the two structures combined. We analyze a few of them in the next sections.

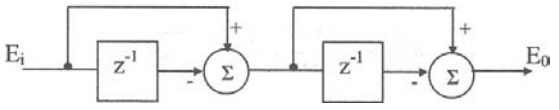
11.2 Nonrecursive Delay-Line Canceller

A single delay-line canceller is shown below. The transfer function and the magnitude of the frequency response are given by



$$H(z) = \frac{E_0(z)}{E_i(z)} = 1 - z^{-1},\ z^{-1} = e^{j\omega T}$$
$$|H(j\omega)| = 2\left|\sin\frac{\omega T}{2}\right| \tag{11.1}$$

A double-delay line canceller is shown below, and the transfer function and the corresponding magnitude of the frequency response are



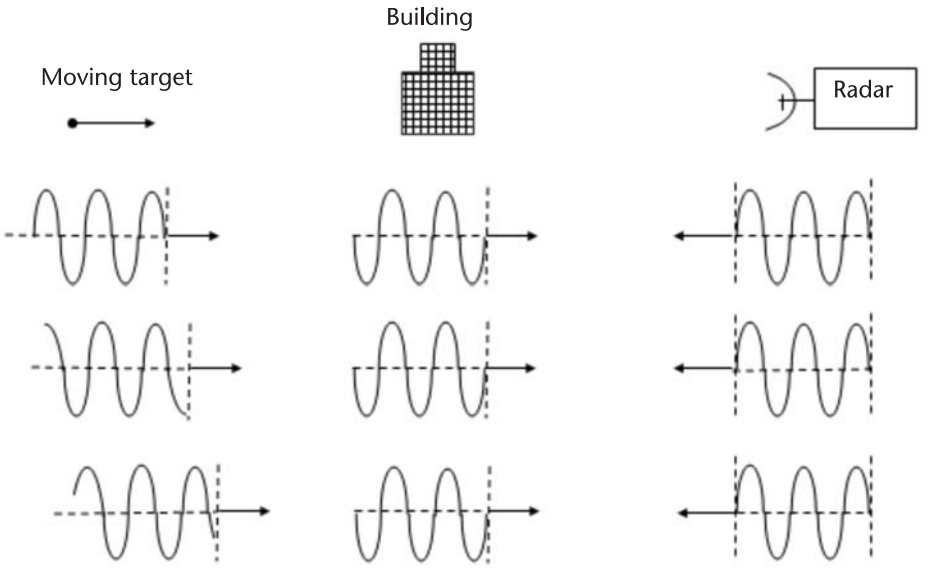
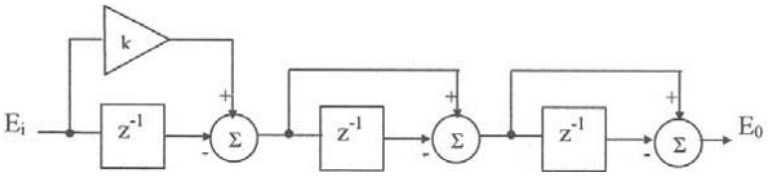


Figure 11.1 Moving target causes phase shifts on returned signals.

$$H(z) = (1 - z^{-1})^2 = 1 - 2z^{-1} + (z^{-1})^2$$
$$|H(j\omega)| = |1 - e^{-j\omega T}|^2 = 4\sin^2\left(\frac{\omega T}{2}\right) \tag{11.2}$$

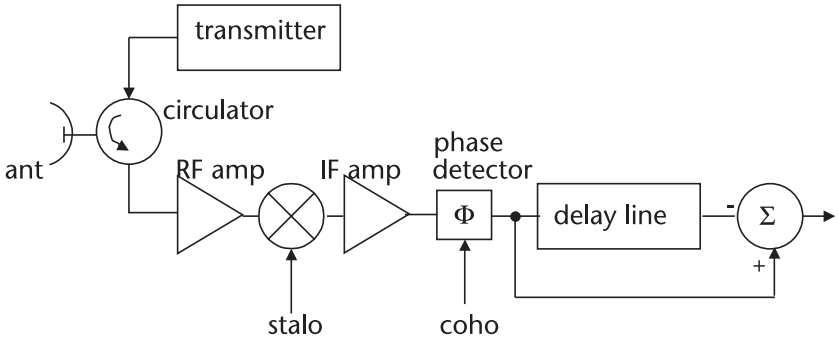
The frequency responses are shown in Figure 11.4 for a single- and double-delay line cancellers after normalization. The responses are similar in general, except in the vicinity of the notch-frequencies.

A triple-delay-line canceller with a feed-forward gain k is shown. An adjustment of the gain produce different frequency response. The transfer function and the magnitude of frequency response are shown in Figure 11.5.



The transfer function and the magnitude of the frequency response are

$$H(z) = (1 - z^{-1})^2(k - z^{-1})$$
$$|H(j\omega)| = 4\sin^2(\frac{\omega T}{2})\{[k\cos\ \omega T + \cos^2\ \omega T - \sin^2\ \omega T]^2 \tag{11.3}$$
$$+ [k\sin\ \omega T + \sin(2\omega T)]^2\}^{1/2}$$



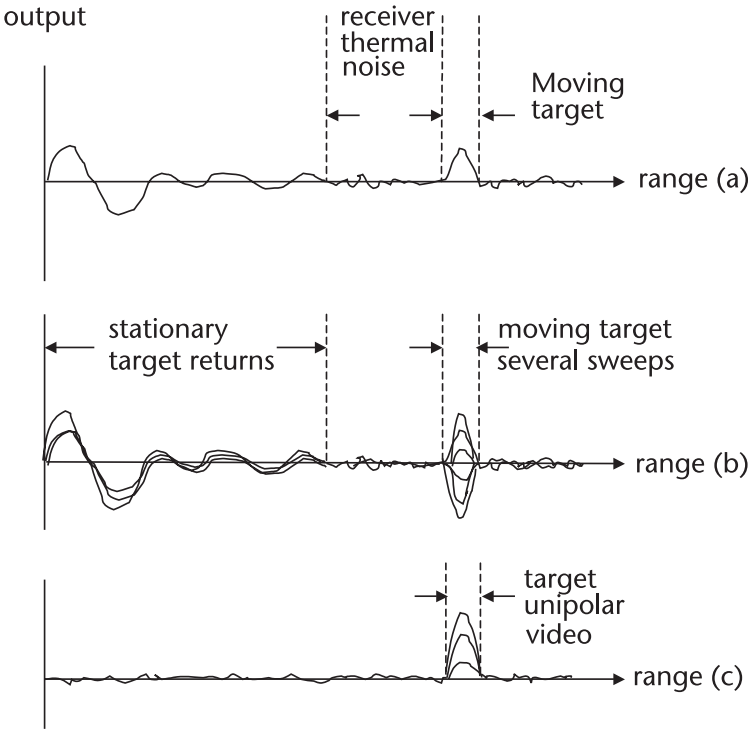


Figure 11.2 Waveform and response of delay line canceller: (a) phase detector output, single sweep, (b) phase detector output, multiple sweeps, and (c) output of delay line canceller with rectification.

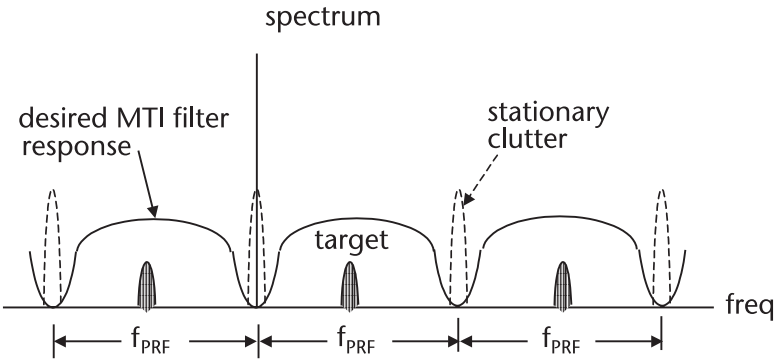


Figure 11.3 Desired frequency response of an MTI filter.

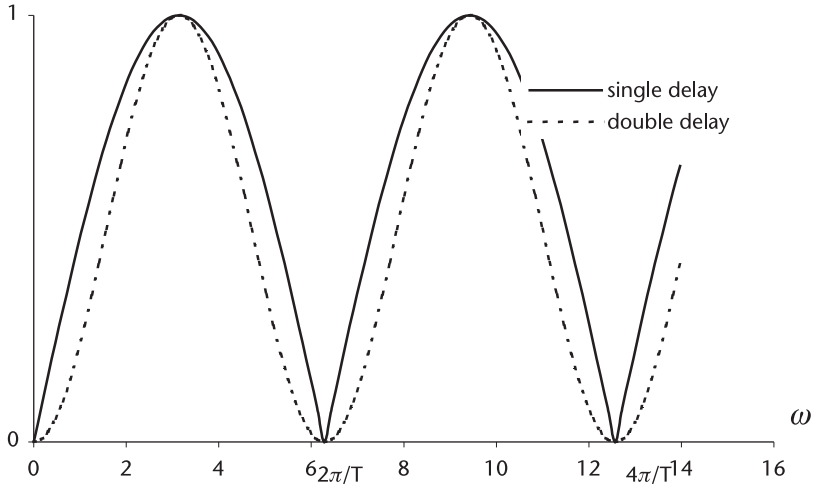


Figure 11.4 Frequency response of a single and double canceller.

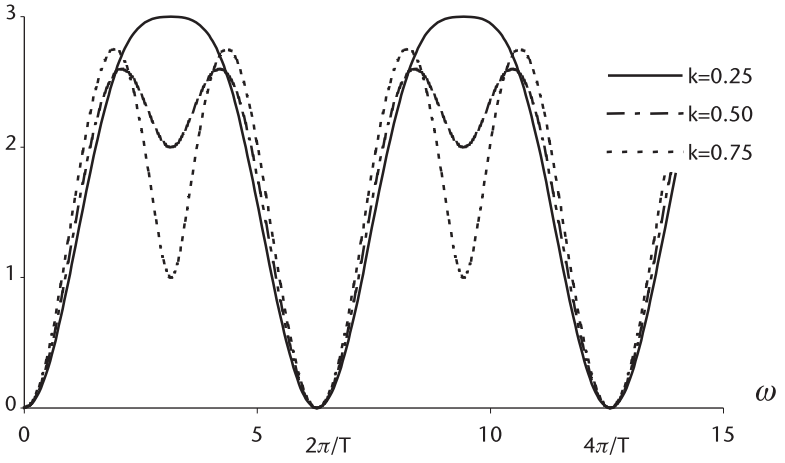
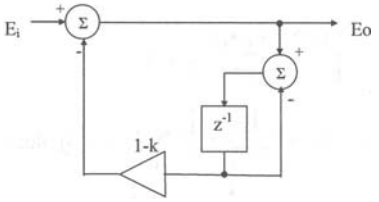


Figure 11.5 Frequency response of a triple-delay canceller.

There are many more different nonrecursive delay-line cancellers with or without gain block in the feed-forward path.

11.3 Recursive Delay-Line Canceller

One example of a recursive single-delay-line canceller is shown.



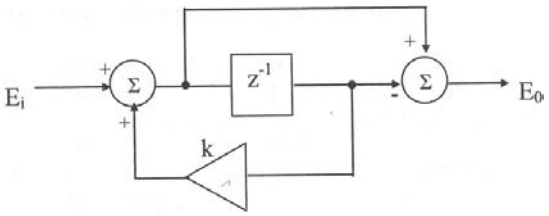
$$H(z) = \frac{1 - z^{-1}}{1 - kz^{-1}} \tag{11.4}$$

$$|H(j\omega)| = \frac{2\left|\sin\frac{\omega T}{2}\right|}{[1 + k^2 - 2k\cos\omega T]^{1/2}}$$

When k = 0, this canceller is identical to the single-delay canceller of (11.4).

The feedback gain control factor k shapes the frequency response. The reciprocal of (1-k) is, in practice, equal to the number of pulses received over the 3-dB antenna beamwidth (i.e., when k = 0.9 the number of pulses is 10).

Again, there are many recursive multiple-delay line cancellers. We give two examples of cancellers with recursive and nonrecursive loops. Readers may benefit by consulting Lindon and Steinberg [1] and Scheleher [5] for multiple-delay cancellers.



$$H(z) = \frac{1 - z^{-1}}{1 - kz^{-1}}$$

$$|H(j\omega)| = \frac{2\left|\sin\frac{\omega T}{2}\right|}{[1 + k^2 - 2k\sin\omega T]^{1/2}} \tag{11.5}$$

The canceller shown above has recursive and nonrecursive loop, however, the frequency response is identical to Figure 11.6.

One example of a double-delay line canceller with recursive and nonrecursive loops is shown below, and the frequency response in Figure 11.7.

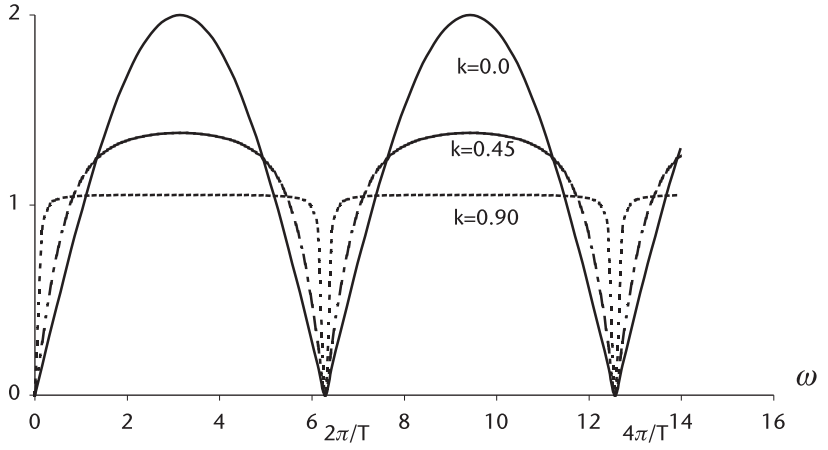
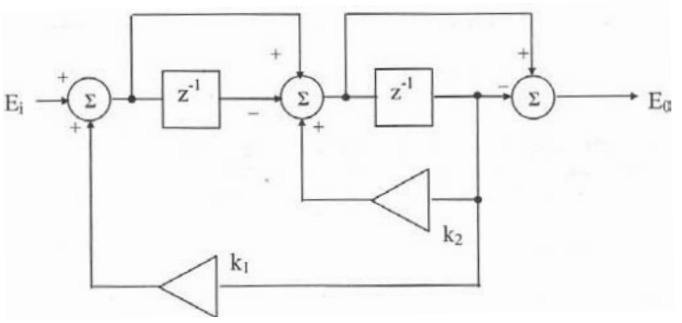


Figure 11.6 Frequency response of a single-delay line recursive canceller.



$$H(z) = \frac{(1 - z^{-1})^2}{1 + (k_1 + k_2)z^{-1} + k_1 z^{-2}}$$

$$|H(j\omega)| = \left| \frac{(1 - e^{-j\omega T})^2}{1 + (k_1 + k_2)e^{-j\omega T} + k_1 e^{-j2\omega T}} \right| \tag{11.6}$$

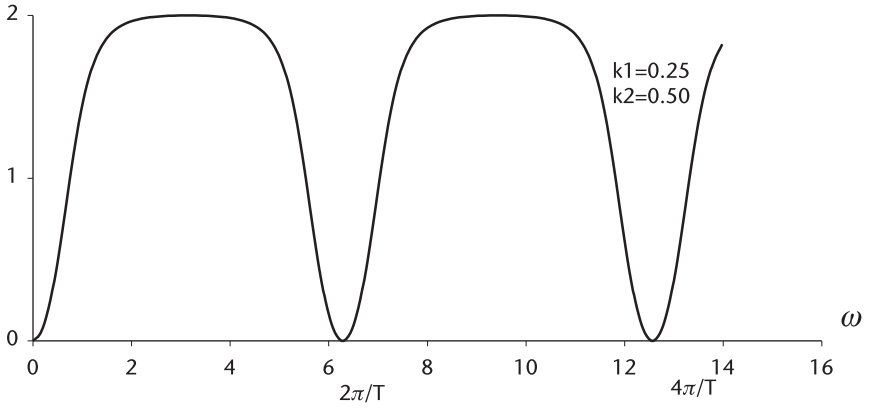
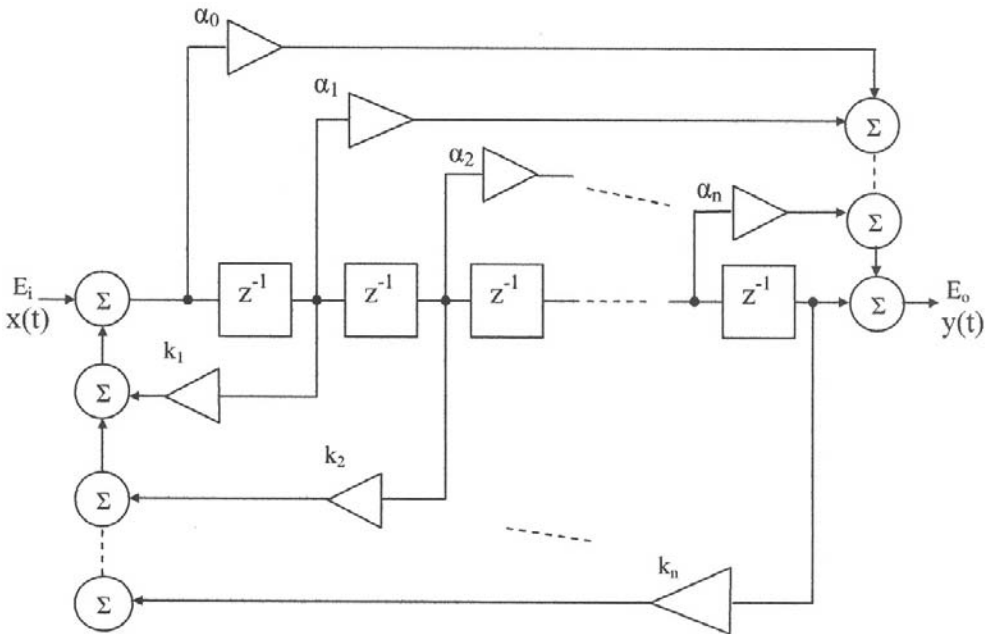


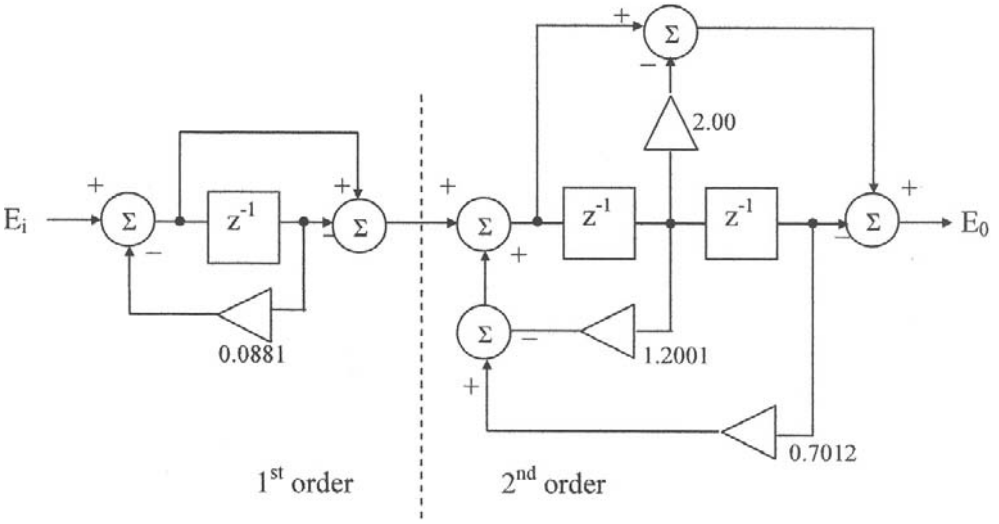
Figure 11.7 Frequency response of double-delay canceller with feedback gain k_1 and k_2.

A canonical configuration of the multiple-delay canceller with recursive and nonrecursive loops is show below.



$$H(z) = \frac{\alpha_0 z^n + \alpha_1 z^{n-1} + \alpha_2 z^{n-2} + \ldots + \alpha_n}{z^n + k_1 z^{n-1} + k_2 z^{n-2} + \ldots + k_n} \tag{11.7}$$

The last example of a triple-delay canceller with 3-pole Chebyshev coefficients applied to the denominator of (11.7) is shown below.



$$H(z) = \frac{(z-1)^3}{(z-0.08821)(z^2-1.2002z+0.7012)} \tag{11.8}$$

Equation (11.8) is programmed in DELAY_C3.CPP, and the frequency response is shown in Figure 11.8.

Several cancellers are analyzed: recursive, nonrecursive, and combined-structure. Cancellers differ from one another in the frequency response, however, all cancellers have nulls at multiples of the pulse repetition frequency, f_{PRF}.

Figure 11.9 shows the frequency response of the air route surveillance radar (ARSR) without canceller, mentioned in Chapter 8, where the system parameters are listed. The null points in Doppler frequency and the corresponding radial velocities of 41 m/sec, 83 m/sec, 124 m/sec, and so on, are indicated. The radar receiver would be unable to detect targets at these velocities. We call them "blind speeds."

$$f_c = 1.3 \text{ GHz}, \qquad f_{PRF} = 360 \text{ Hz}$$

$$v_{blind} = k\frac{\lambda f_{PRF}}{2}, \qquad k = 0, \pm 1, \pm 2, \ldots$$

$$f_{dopper} = \frac{2\, v_T}{\lambda}$$

λ: transmitter carrier wavelength, $\lambda = \frac{c}{f}$, m

v_T: target radial velocity, m/sec

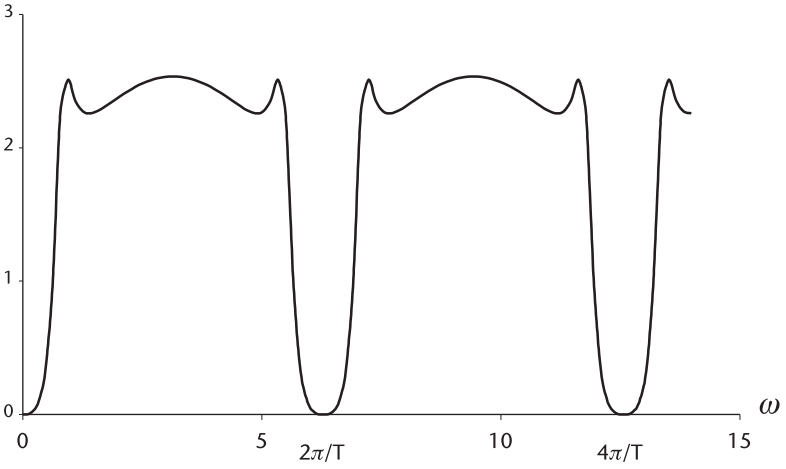


Figure 11.8 Frequency response of triple canceller 3-pole Chebyshev coefficients.

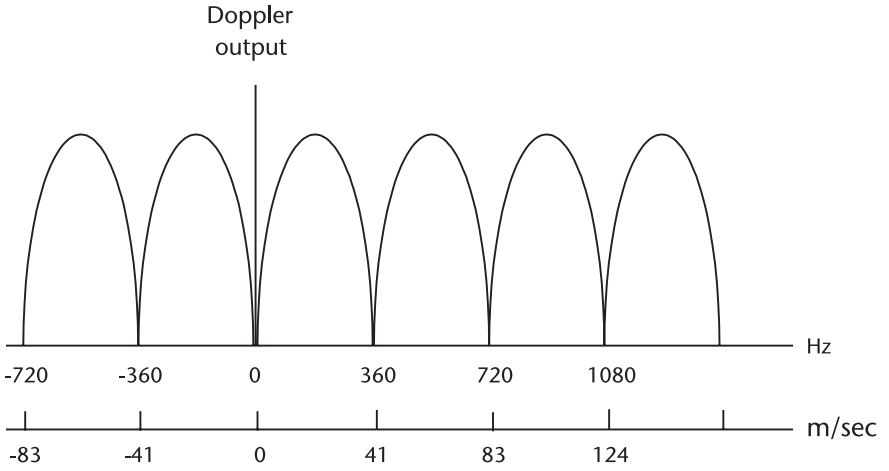


Figure 11.9 Blind speeds of air route surveillance radar (ARSR).

11.4 Blind Speeds and Staggered PRFs

The blind speeds inherent to all delay-line cancellers impose serious problems in moving target indication. The problem can be mitigated by varying the pulse repetition frequency and using corresponding delay-lines in the cancellers. The pulse repetition frequency is thereby altered to switching the first blind speed. We call this staggered PRFs.

Two examples of stagger-time management are shown in Figure 11.10.

The frequency response of a double-stagger with a single-delay canceller is given by

$$|H(j\omega)| = \left[\frac{1}{2}\left|1 - e^{-j\omega T_1}\right|^2 + \frac{1}{2}\left|1 - e^{-j\omega T_2}\right|^2\right]^{1/2}$$

where T_1= T–Ts and T_2 =T+Ts, and the power response is given by

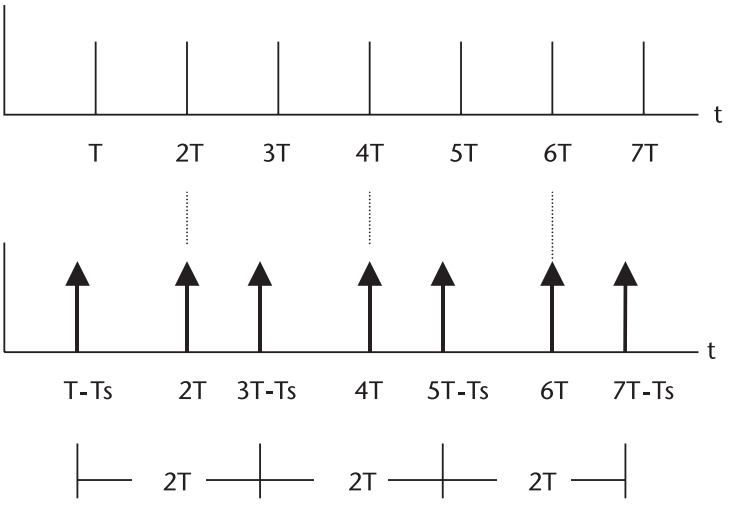
$$|H(j\omega)|^2 = \frac{1}{2}\left|1 - e^{-j\omega T_1}\right|^2 + \frac{1}{2}\left|1 - e^{-j\omega T_2}\right|^2 \tag{11.9a}$$

$$= 2.0 - \cos\,\omega T_1 - \cos\,\omega T_2 \tag{11.9b}$$

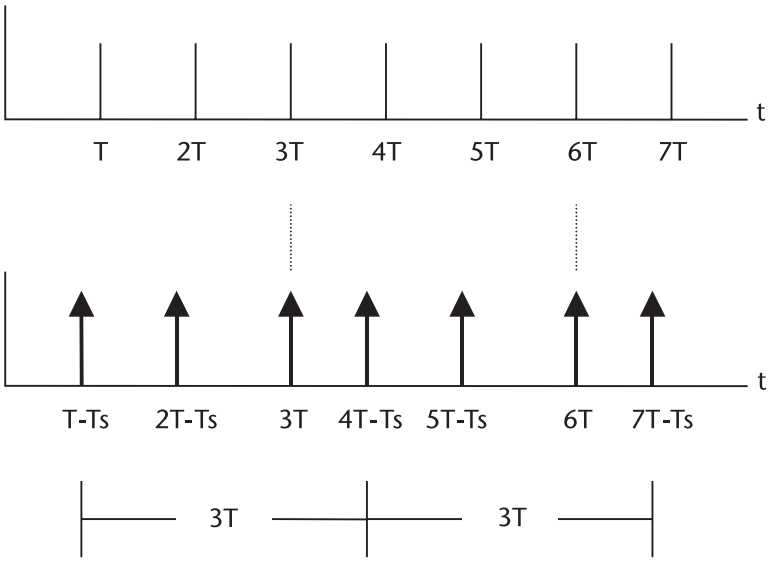
Eq(11.9) can be further simplified to

$$|H(j\omega)|^2 = 2.0[1.0 - \cos\,\omega T \cos\,\omega Te]$$

where T is the unstaggered period (in Figure 11.10) and e = Ts/T is the fraction of (Ts < T) of delay period. We often hear about the "stagger ratio" rather than the stagger fraction. For a double-stagger when $T_1 = T - Ts$ and $T_2 = T + Ts$, or $T_1 = T(1 - e)$ and $T_2 = T(1 + e)$, the stagger ratio r is given by



(a) Double Staggers



(b) Triple staggers

Figure 11.10 Stagger timing diagram. (*From:* [5], p. 395.)

$$r = \frac{T_1}{T_2} = \frac{1-e}{1+e} \quad \text{or} \quad e = \frac{1-r}{1+r}$$

The higher the stagger ratio approaching unity, the higher the first blind speed; however, there are some deep attenuations in the passband.

The power response for a double-stagger with a double-delay canceller is given by

$$|H(j\omega)|^2 = \frac{1}{2}\left|a + b\ e^{-j\omega T_1} + c\ e^{-j\omega(T_1+T_2)}\right|^2 + \frac{1}{2}\left|a + b\ e^{-j\omega T_2} + c\ e^{-j\omega(T_1+T_2)}\right|^2 \tag{11.10a}$$

where a, b, and c are the binomial coefficients: a=1, b= -1, c=1.

Eq(11.10 a) can be further simplified to

$$|H(j\omega)|^2 = 6.0 - 8.0\cos\omega T\cos\omega Te + 2.0\cos 2\omega T \tag{11.10b}$$

where $e = Ts/T$

The power response for a triple-stagger with a double-delay canceller is given by

$$|H(j\omega)|^2 = |1.0 - 2.0e^{-j\omega T_1} + e^{-j\omega(T_2+T_3)}|^2 + |1.0 - 2.0e^{-j\omega T_2} + e^{-j\omega(T_1+T_3)}|^2 + |1.0 - 2.0e^{-j\omega T_3} + e^{-j\omega(T_1+T_2)}|^2 \tag{11.11a}$$

where $T_1 = T(1-e)$, $T_2 = T$, $T_3 = T(1+e)$, and $e = Ts/T$, and the simplified form is

$$|H(j\omega)|^2 = \frac{1}{3}[18.0 - 8.0(1.0 + 2.0\cos\omega Te)\cos\omega T + 2.0(1.0 + 2.0\cos\omega Te)\cos 2\omega T] \tag{11.11b}$$

Eq (11.11) is programmed in MTI_3s2d.CPP and the result is shown in Figure 11.11.

For example, if the ARSR radar should not be blind to an aircraft at speeds below 1.0 Mach (330 m/sec), the stagger ratio of 3/5 or higher must be designed. The product of the order of canceller and the number of staggers should be less than the number of return signal over the antenna beamwidth so that the delay lines are completely filled, and the integrator (if recursive loop is involved) has reached a steady-state level. In practice, nonrecursive cancellers are preferred because of their superior transient response.

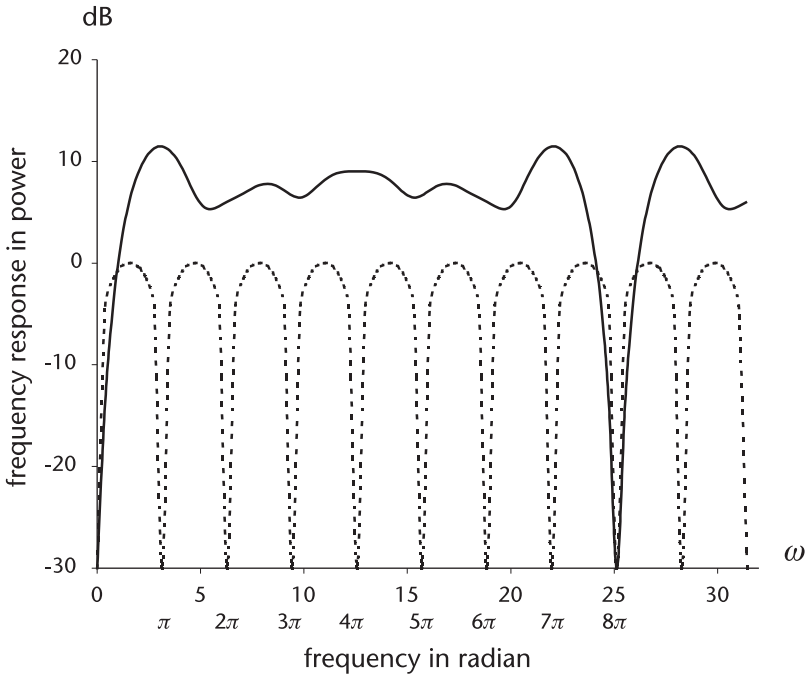


Figure 11.11 A triple-stagger, double-delay-line canceller.

For the case of the ARSR, the number of hits (return pulses) is

$$N_B = (\theta/\dot{\theta})f_{prf} = \frac{1.35}{36}(360) = 13.5 \rightarrow 13$$

Thus, a double (or triple) canceller with triple (or double) staggers would be a good candidate structure for the ARSR.

Since there are so many different cancellers that can be candidates, it is logical to establish certain criteria to evaluate the performance of cancellers.

An evaluation of an MTI filter would include the order of canceller, the number of staggers, a recursive or nonrecursive, the binomial coefficients or other polynomial coefficients for the weights of transfer function, the complexity of signal processing, and so forth.

We study two performance criteria; the clutter attenuation and the improvement factor in the next section.

11.5 Clutter Attenuation and Improvement Factor

This section introduces two quantities that provide a measure of MTI filter performance: clutter attenuation, CA, and improvement factor, I. The clutter attenuation is defined as the ratio of clutter input power to clutter output power. The improvement factor is defined as the ratio of signal-to-clutter ratio at the output to that of at the input. Both quantities are dimensionless.

$$\text{CA} = \frac{\text{clutter power at the input}}{\text{clutter power at the output}}$$

$$\text{I} = \frac{(\text{S/C})_\text{o} \text{ at the output}}{(\text{S/C})_\text{i} \text{ at the input}}$$

Thus,

$$\text{I} = \left(\frac{\text{S}_\text{o}}{\text{S}_\text{i}}\right)\left(\frac{\text{C}_\text{i}}{\text{C}_\text{o}}\right) = (\text{average gain of filter for target}) \times \text{CA} \tag{11.12}$$

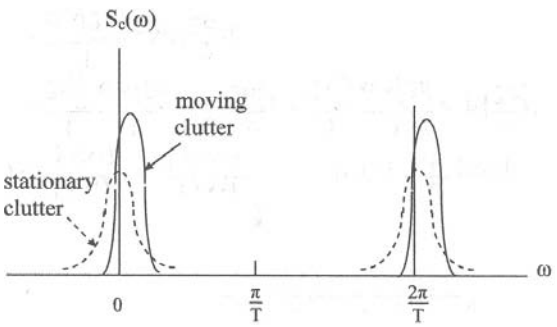
The clutter attenuation CA is obtained by the ratio of the following two integrals.

$\text{S}_\text{c}(\omega)$: clutter power spectral density

$|\text{H}(\text{j}\omega)|$: the magnitude of canceller response

$$\text{CA} = \frac{\int_{-\infty}^{\infty} \text{S}_\text{c}(\omega)\text{d}\omega}{\int_{-\infty}^{\infty} \text{S}_\text{c}(\omega)|\text{H}(\text{j}\omega)|^2\text{d}\omega} \tag{11.13}$$

The lower limit of integration will be replaced by zero and the upper limit by $2\pi/\text{T}$ since the spectrum is periodic in $2\pi/\text{T}$.



If we assume that the clutter spectrum is a Gaussian much narrower than $2\pi/\text{T}$, and that for the moment the clutter has zero mean velocity, the clutter spectrum would be expressible as

$$\text{S}_\text{c}(\omega) = \frac{\text{P}_\text{c}}{\sigma_\text{c}\sqrt{2\pi}} \exp\left\{\frac{-\omega^2}{2\sigma_\text{c}^2}\right\} \tag{11.14}$$

P_c : peak power of clutter spectrum

σ_c : the standard deviation of clutter spectrum

Equation (11.14) is substituted into (11.13), and CA is to be evaluated for various cancellers.

$$\mathrm{CA} = \left[\frac{1}{\sigma_c \sqrt{2\pi}} \int_0^{2\pi/\mathrm{T}} \exp\left\{ \frac{-\omega^2}{2\sigma_c^2} \right\} |\mathrm{H}(\mathrm{j}\omega)|^2 \, \mathrm{d}\omega \right]^{-1}$$

For a single canceller, whose response is given by (11.1, the integration is

$$\begin{aligned} &\frac{4}{\sigma_c \sqrt{2\pi}} \int_0^{2\pi/\mathrm{T}} \exp\left\{ \frac{-\omega^2}{2\sigma_c^2} \right\} \sin^2\left(\frac{\omega \mathrm{T}}{2} \right) \mathrm{d}\omega \\ &\approx \frac{(2)(4)}{\sigma_c \sqrt{2\pi}} \int_0^{\infty} \exp\left\{ \frac{-\omega^2}{2\sigma_c^2} \right\} \left(\frac{\omega \mathrm{T}}{2} \right)^2 \mathrm{d}\omega \quad \left(\frac{\omega \mathrm{T}}{2} \right) <<< 1 \\ &= (\mathrm{T}\sigma_c)^2 \end{aligned}$$

Thus,

$$\mathrm{CA} = \frac{1}{(\mathrm{T}\sigma_c)^2}, \text{ and } \mathrm{I} = \frac{2}{(\mathrm{T}\sigma_c)^2} \tag{11.15}$$

Equation (11.15) indicates that the CA and I are inversely proportional to the variance of the clutter power spectrum; the wider the spectrum width, the lesser the clutter attenuation and the improvement factor.

The exact expressions for CA and I are obtained through autocorrelation function as follows:

When the clutter input is x(t), the mean-squared clutter input power is

$$\mathrm{E}\{\mathrm{x}(\mathrm{t})\mathrm{x}^*(\mathrm{t})\} = \mathrm{E}\{|\mathrm{x}^2(\mathrm{t})|\} = \mathrm{R}(0)$$

When the output y(t) is expressed as, for a single-delay canceller

$$\mathrm{y}(\mathrm{t}) = \mathrm{x}(\mathrm{t}) - \mathrm{x}(\mathrm{t} + \mathrm{T})$$

The mean-squared output power is

$$\begin{aligned} \mathrm{E}\{\mathrm{y}^2(\mathrm{t})\} &= \mathrm{E}\{[\mathrm{x}(\mathrm{t}) - \mathrm{x}(\mathrm{t} + \mathrm{T})]^2\} \\ &= \mathrm{E}\{\mathrm{x}^2(\mathrm{t})\} - 2\mathrm{E}\{\mathrm{x}(\mathrm{t})\mathrm{x}(\mathrm{t} + \mathrm{T})\} + \mathrm{E}\{\mathrm{x}^2(\mathrm{t} + \mathrm{T})\} \\ &= 2\mathrm{R}(0) - 2\mathrm{R}(\mathrm{T}) \end{aligned}$$

The autocorrelation function R(T) and the spectral power spectral density S(ω) are Fourier transform pair:

$$S(\omega) = \int_{-\infty}^{\infty} R(T)e^{-jwt}\, dT \tag{11.16}$$

$$R(T) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega)e^{jwt}\, d\omega \tag{11.17}$$

When x(t) is real, R(T) is real and even. Then, (11.16) and (11.17) take the form

$$S(\omega) = \int_{-\infty}^{\infty} R(T) \cos\, \omega T\, dT$$

$$R(T) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cos\, \omega T\, d\omega$$

and, assuming the clutter spectrum is a Gaussian, R(0) and R(T) are expressible as

$$R(0) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega)\, d\omega = \frac{2}{2\pi} \int_{-\infty}^{\infty} \exp\left\{\frac{-\omega^2}{2\sigma_\omega^2}\right\} d\omega = \frac{\sigma_\omega}{\sqrt{2\pi}}$$

$$R(T) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cos\, \omega T\, d\omega = \frac{2}{2\pi} \int_{-\infty}^{\infty} \exp\left\{\frac{-\omega^2}{2\sigma_\omega^2}\right\} \cos\, \omega T$$

$$= \frac{\sigma_\omega}{\sqrt{2\pi}} \exp\left\{\frac{-T^2\sigma_\omega^2}{2}\right\}$$

Thus, the autocorrelation coefficient $\rho(T)$ is given by

$$\rho(T) = \frac{R(T)}{R(0)} = \exp\left\{\frac{-T^2\sigma_\omega^2}{2}\right\}$$

Substituting the result in CA_1 and I_1 for a single-delay canceller, we have

$$CA_1 = \frac{E\{x^2(t)\}}{E\{y^2(t)\}} = \frac{1}{2[1-\rho(T)]} = \frac{1}{2\left[1 - \exp\left\{\frac{-T^2\sigma_\omega^2}{2}\right\}\right]}$$

$$I_1 = \text{gain} \times CA_1 = \frac{2}{2[1-\rho(T)]} = \frac{1}{1 - \exp\left\{\frac{-T^2\sigma_\omega^2}{2}\right\}}$$

The numerator 2 of I_1 comes from the average power gain of a single-delay canceller, (11.1). The standard deviation of clutter power spectrum σ_ω (rad/sec) will be transformed to an equivalent standard deviation of the clutter velocity σ_v (m/sec) by the Doppler principle, and $T=1/f_{PRF}$, so that the improvement factor can be related to the system parameters λ and f_{PRF}.

$$f_d = \frac{2v_r}{\lambda},\ \sigma_f = \frac{2\sigma_v}{\lambda}\ \ \sigma_\omega = 2\left(\frac{2\pi\sigma_v}{\lambda}\right)\ (2\pi \text{ converts radian/sec to Hz})$$

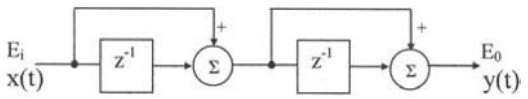
and

$$\frac{T^2\sigma_\omega^2}{2} = 2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2$$

Finally,

$$I_1 = \left[1 - \exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\right]^{-1} \tag{11.18}$$

The expressions for clutter attenuation CA_2 and improvement factor I_2 for a double-delay canceller can be derived by following the identical steps as for the single-delay canceller.



$$y(t) = x(t) - 2x(t+T) + x(t+2T)$$

$$E\{y^2(t)\} = E\{[x(t) - 2x(t+T) + x(t+2T)]^2\} = 6R(0) - 8R(T) + 2R(2T)$$

$$CA_2 = \frac{E\{x^2(t)\}}{E\{y^2(t)\}} = \frac{1}{6\left[1 - \frac{4}{3}\rho(T) + \frac{1}{3}\rho(2T)\right]} \tag{11.19}$$

$$I_2 = \text{gain} \times CA_2 = \frac{6}{6\left[1 - \frac{4}{3}\rho(T) + \frac{1}{3}\rho(2T)\right]} \tag{11.20}$$

The numerator 6 of I_2 comes from the average power gain of the double-delay nonrecursive canceller. Substituting the correlation coefficients

$$\rho(T) = \exp\left\{\frac{-T^2\sigma_\omega^2}{2}\right\} = \exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}$$

$$\rho(2T) = \exp\{-2T^2\sigma_\omega^2)\} = \exp\left\{-2\left(\frac{4\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}$$

The improvement factor I_2 is given by

$$I_2 = \left[1 - \frac{4}{3}\exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\} + \frac{1}{3}\exp\left\{-2\left(\frac{4\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\right]^{-1} \qquad (11.21)$$

The standard deviation of clutter σ_v has been experimentally measured by instrumented radars by many researchers, and compiled by Barton [2, p. 100], listed below.

Table 11.1 Standard Deviation of Clutter Spectrum

Clutter Type	*Wind Speed (Knots)*	σ_v *(m/sec)*
Wooded hills	Calm	0.017
	10	0.04
	20	0.22
	40	0.32
Sea state	-	0.78 ~ 1.0
	8 ~ 20	0.48 ~ 1.1
Rain cloud	-	1.9 ~ 4.0

Equations (11.18) and (11.21) are programmed in IMPRVMT.CPP having $\sigma_v/(\lambda\ f_{PRF})$ as an independent variable, and the result is shown in Figure 11.12.

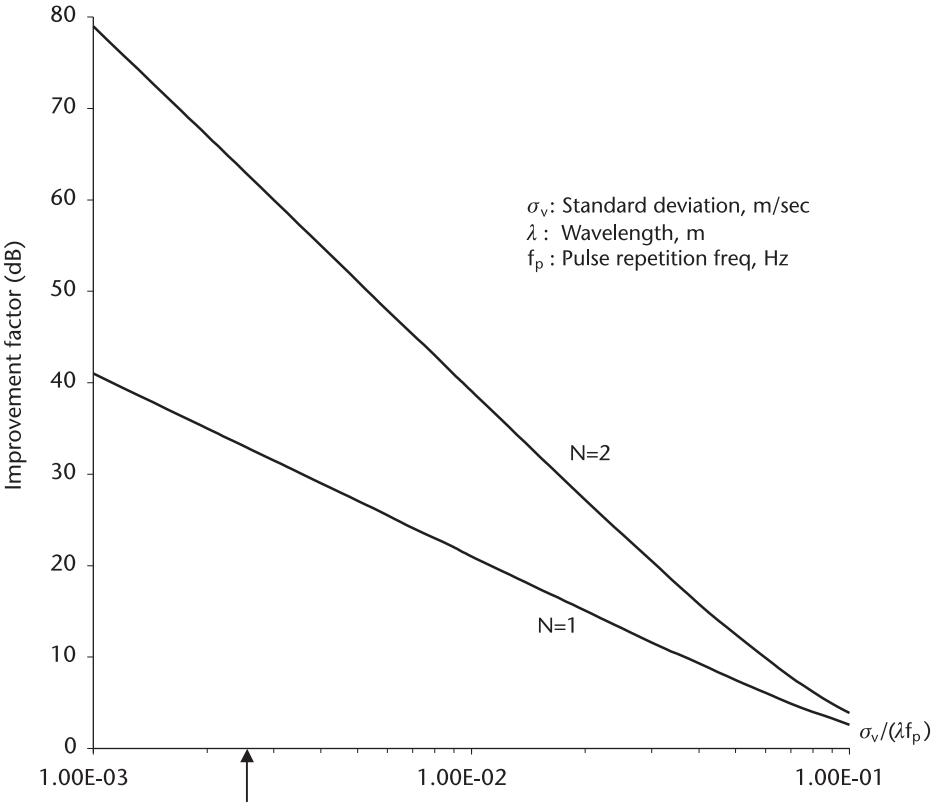


Figure 11.12 Improvement factor, mean velocity = 0.0.

A numerical example is given for the airport surveillance radar (ASR), an arrow on the horizontal axis of Figure 11.12. The system parameters are given in Chapter 8 and repeated below:

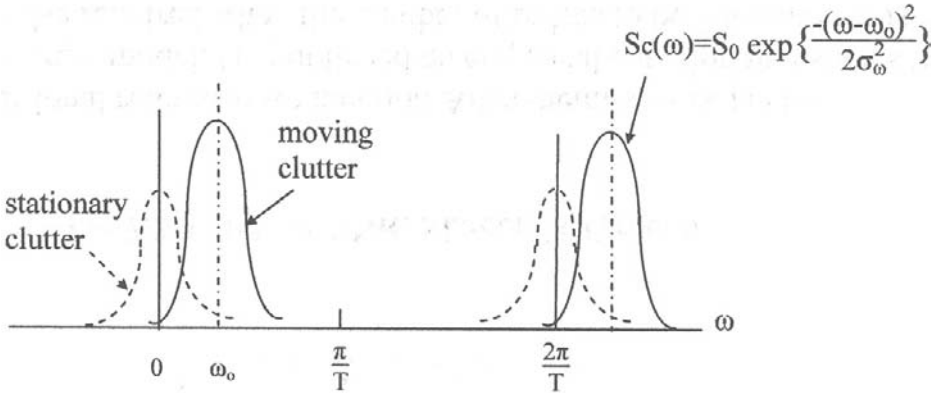
$$f = 2.8 \text{ GHz}, \ \lambda = 10.71 \text{ cm}, \ f_{PRF} = 1{,}200 \text{ Hz}$$

Assume that the land clutter (hills and mountains with forest cover) has the standard deviation σ_v= 0.32 m/sec when the wind speed is 40 knots. The improvement factors I_1 and I_2 are

$$I_1 = \left[1 - \exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\right]^{-1} \approx 2{,}045 = 33.1 \text{ dB}$$

$$I_2 = \left[1 - \frac{4}{3}\exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\} + \frac{1}{3}\exp\left\{-2\left(\frac{4\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\right]^{-1} \approx 2{,}091{,}831 = 63.2 \text{ dB}$$

Next, we shall investigate how the mean velocity of clutter would reduce the improvement factor. The power spectrum of clutter with a nonzero mean velocity is shown below.



The autocorrelation function is derived analogous to the stationary clutter.

$$R(T) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_c(\omega)\cos\omega T\, d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\left\{\frac{-(\omega-\omega_o)^2}{2\sigma_\omega^2}\right\}\cos\omega T\, d\omega$$

Let

$$\begin{aligned} R(T) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\left\{\frac{-\Omega^2}{2\sigma_\omega^2}\right\}\cos[(\Omega+\omega_o)T]\, d\Omega \\ &= \frac{2}{2\pi}\int_{-0}^{\infty} \exp\left\{\frac{-\Omega^2}{2\sigma_\omega^2}\right\}\cos\Omega T\cos\omega_o T\, d\Omega \\ &= \frac{\sigma_\omega}{\sqrt{2\pi}}\cos\omega_o T\exp\left\{\frac{-T^2\sigma_\omega^2}{2}\right\} \end{aligned}$$

We shall relate ω_o to the mean velocity of clutter v_o by

$$f_d = \frac{2v_r}{\lambda},\ 2\pi f_d = \frac{4\pi v_r}{\lambda}$$

$$\omega_o = \frac{4\pi v_o}{\lambda} \text{ and } \omega_o T = \frac{4\pi v_o}{\lambda f_{PRF}}$$

Substituting the result, we have the improvement factors I_1 and I_2 when the clutter has a nonzero mean velocity v_o and the standard deviation σ_v.

$$I_1 = \left[1 - \exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\cos\left(\frac{4\pi v_o}{\lambda\ f_{PRF}}\right)\right]^{-1} \tag{11.22}$$

$$I_2 = \left[1 - \frac{4}{3}\exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\cos\left(\frac{4\pi v_o}{\lambda\ f_{PRF}}\right) + \frac{1}{3}\exp\left\{-2\left(\frac{4\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\cos\left(\frac{8\pi v_o}{\lambda\ f_{PRF}}\right)\right]^{-1} \tag{11.23}$$

We note that exponentials of (11.18) and (11.21) are multiplied by factors depending on the cosine of 4π or 8π times the ratio of the mean velocity v_o to the product of wavelength λ and pulse repetition frequency f_{PRF}. The reduction in the improvement is programmed in REDUCTN.CPP and the result is shown in Figure 11.13.

From Figure 11.12 we read that a double-delay canceller has an improvement factor of 39 dB when the clutter is stationary and $\sigma_v/(\lambda f_{PRF})$=0.01. When the mean

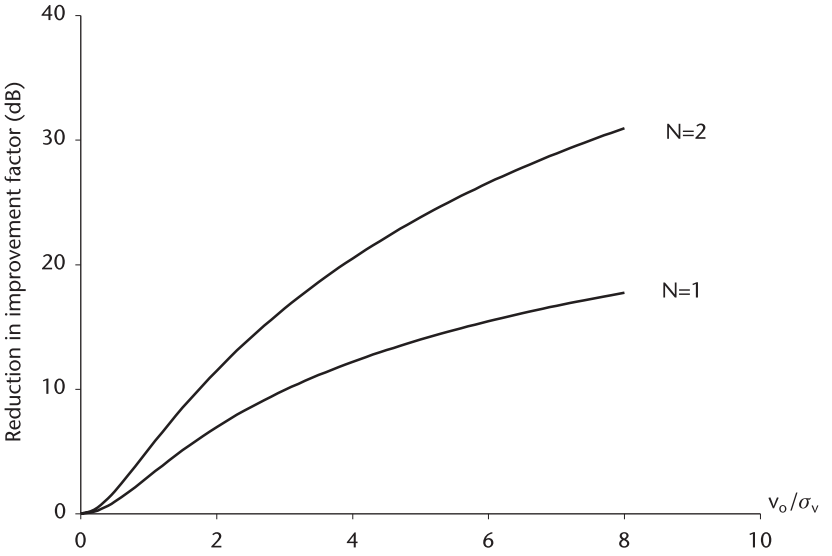


Figure 11.13 Reduction in improvement factor, mean velocity ≠ 0.

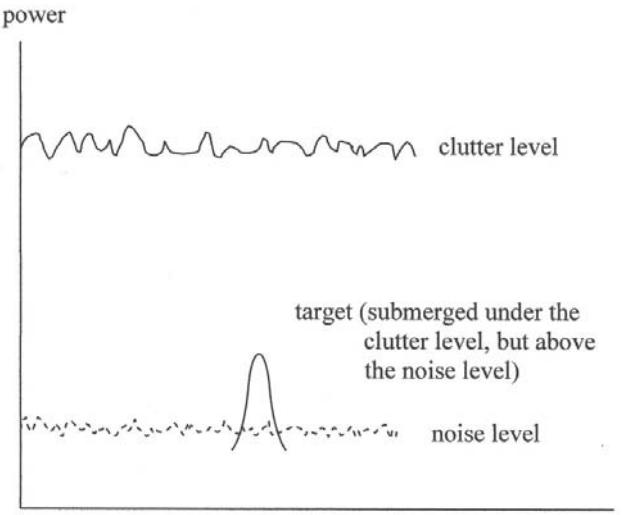
velocity v_o is four times the standard deviation σ_v, Figure 11.13 indicates there is a reduction of 21 dB in improvement factor. The result will be 39 dB 21 dB = 18 dB.

Another example is given. A double canceller is expected to have an improvement factor of 22 dB when the clutter is stationary and $\sigma_v/\lambda f_{PRF} = 0.03$.

However, if the clutter has a mean velocity v_o four times σ_v, the reduction is 21 dB. The result will be 22 dB 21 dB = 1 dB. In this case we do not expect any clutter suppression at all. The mean velocity has a punishing effect on the improvement factor unless the canceller's attenuation notch is matched to the mean velocity of the clutter. Can we estimate the mean velocity and reposition the canceller's null to the mean velocity?

There appears to be a few techniques to accomplish the task. The pulse-pair-processing (PPP) has been applied in estimating the mean and variance of precipitation clutter [6–10]. The TCAR[1] technique, a canceller at IF frequency (or equivalent IQ channel in baseband) has gained the popularity among many developers [11, 12].

Before we leave the subject of improvement factor, we shall bring a refinement to (11.18), (11.24), and Figure 11.12. A question here is: can we expect an improvement of 63.2 dB by the ASR with a double-delay canceller when $\sigma_v = 0.32$ m/sec mentioned earlier. Can we expect an improvement factor of, say, 60 dB when the noise power level is only 20 dB below the clutter level? The noise will pass through the canceller with unchanged power.



When the power spectrum density of the clutter is Gaussian with $\sigma <<< \pi T$, as shown in the previous page, the clutter return and the noise can be combined, and we call the combined signal as an interference. Now we have to contend with the signal-to-interference ratio (SIR), not the signal-to-clutter ratio (SCR) alone. The SIR is given by

$$\text{SIR} = \frac{1}{\frac{1}{\text{SNR}} + \frac{1}{\text{SCR}}}$$

1. TACCAR stands for time average coherent clutter airborne radar, originally developed for airborne MTI.

Accordingly, (11.21) shall be modified.

$$I_2 = \left[1 - \frac{4}{3}\exp\left\{-2\left(\frac{2\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\} + \frac{1}{3}\exp\left\{-2\left(\frac{4\pi\sigma_v}{\lambda\ f_{PRF}}\right)^2\right\}\right]^{-1} \tag{11.24}$$

$$I_2' = (1 - P_{CN})[\text{Eq (11.5.10)}] + P_{CN}$$

where P_{CN} is a fraction of the noise power to the clutter power. Equation (11.24) is programmed in MTI_CNR.CPP and the results are shown in Figure 11.14. A single-delay canceller would be modified similarly.

Next we shall analyze the extent of improvement reduction due to antenna rotation because the rotation causes the spectrum broadening.

We start with antenna radiation pattern assumed to be Gaussian (see other patterns in Chapter 6.)

$$G(\theta) = G_{\theta o}\exp\left\{\frac{-\theta^2}{2\sigma_\theta^2}\right\}$$

When the antenna rotates, the received signal is modulated by G(t) time function as shown below, and the spectrum of clutter will be obtained by Fourier transformation.

$$\begin{cases} G(t) = G_{to}\exp\left\{\dfrac{-t^2}{2\sigma_t^2}\right\} \text{ where } \sigma_t = \sigma_\theta/\dot{\theta},\ \dot{\theta} \text{ is rotation rate} \\ G(\omega) = G_{\omega o}\exp\left\{\dfrac{-\omega^2}{2\sigma_\omega^2}\right\} \end{cases}$$

The standard deviation of the antenna pattern σ_θ can be replaced by the half-power beamwith, $\theta_{(-3dB)} = \theta_B$, which for a Gaussian pattern is related by [3, p. 149].

$$\theta_B = 2.3548\ \sigma_\theta, \text{ so that } G(\theta)\exp\left\{\frac{-\theta^2}{2\sigma_\theta^2}\right\} = G(\theta)\exp\left\{\frac{-2.7726\ \theta^2}{\theta_B^2}\right\}$$

$$\exp\left\{\frac{-\theta^2}{2\sigma_\theta^2}\right\} \rightarrow \exp\left\{\frac{-t^2}{2\sigma_t^2}\right\} \rightarrow \exp\left\{\frac{-\omega^2}{2\sigma_\omega^2}\right\} \rightarrow \exp\left\{\frac{-2.7726\ \theta^2}{\theta_B^2}\right\}$$

$$\sigma_t = \sigma_\theta/\dot{\theta} \qquad 2\sigma_\omega^2 = \frac{1}{2\sigma_t^2} \qquad \theta_B = 2.3548\ \sigma_\theta$$

$$\begin{bmatrix}\text{antenna}\\ \text{rotation}\end{bmatrix} \qquad \begin{bmatrix}\text{Fourier}\\ \text{transform}\end{bmatrix} \qquad \begin{bmatrix}\text{conversion to}\\ -3\text{dB beamwith}\end{bmatrix}$$

$$2\sigma_\omega^2 = \frac{2}{2\sigma_t^2} = \cdot\frac{1}{2}\left(\frac{\theta}{\sigma_\theta}\right)^2,\ \sigma_\omega = \frac{1}{2}\left(\frac{\dot{\theta}}{\sigma_\theta}\right) = \frac{2.3548}{2}\left(\frac{\dot{\theta}}{\theta_B}\right)$$

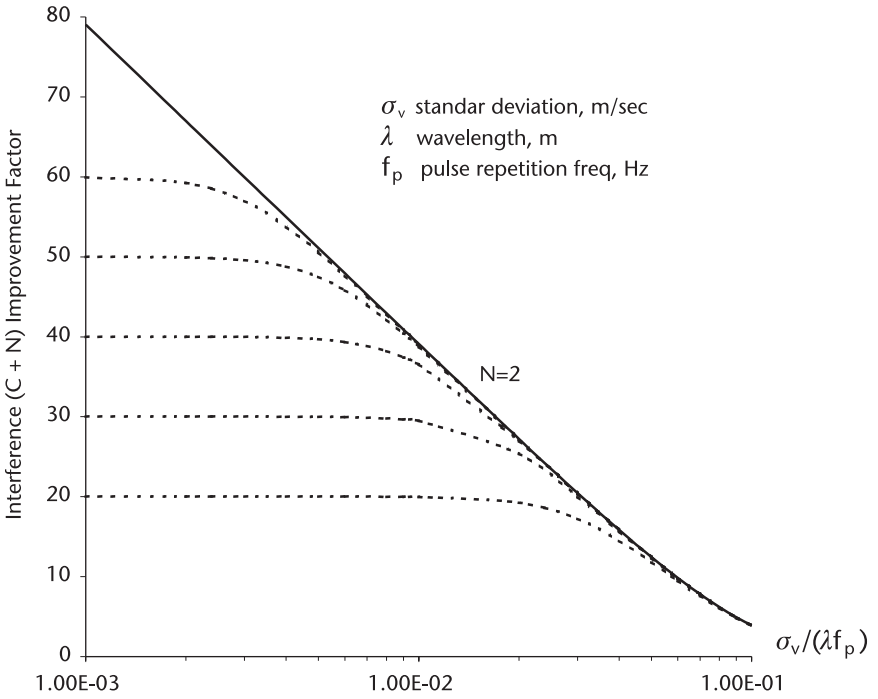


Figure 11.14 Improvement Factor, mean velocity = 0.0, limited by signal-to-interference ratio (SIR).

The standard deviation σ_ω (rad/sec) can be converted to σ_f (Hz) by dividing σ_ω by 2π, and the standard deviation σ_f can be related to that of velocity by Doppler shift.

$$\sigma_f = 0.0338(\dot{\theta}\ \theta_B)\frac{1}{2\pi} = \frac{2\sigma_v}{\lambda}$$

Finally

$$\sigma_v{=}0.0169\left(\theta.\ \theta_B\,\lambda\right), \quad \text{or} \quad \frac{\sigma_v}{\lambda} = 0.0169\left(\dot{\theta}\,\theta_B\right) \tag{11.25}$$

Equation (11.25) is shown in Figure 11.15 having $\dot{\theta}$ (deg/sec) as the independent variable and θ_B as a parameter.

A line for airport surveillance radar (ASR) is added to give readers the magnitude of the spectrum broadening of the stationary clutter due to antenna rotation. We note the standard deviation of stationary clutter for the ASR due to antenna rotarion is σ_v= 0.244 m/sec, which is comparable to all stationary land clutter (see *Table 11.1*). The improvement factor I_2 is lowered to 51 dB compared to 63.2 dB with no rotation.

The total standard deviation of clutter seen by a radar is a square-root of sum of all variance of contribution cause.

$$\sigma_t = [\sigma_c^2 + \sigma_{ar}^2 + \sigma_{si}^2 + \sigma_{pm}^2 + \ldots]^{1/2}$$

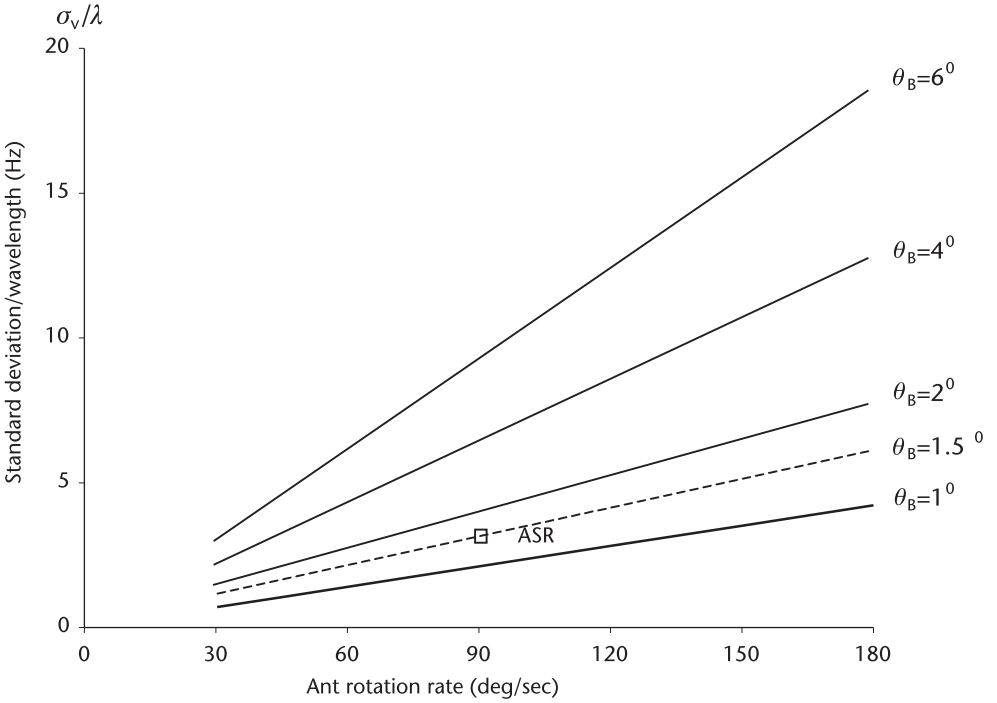


Figure 11.15 Spectrum broadening due to antenna rotation.

where σ_t: total standard deviation
σ_c^2: variance of clutter
σ_{ar}^2: variance due to antenna rotation
σ_{si}^2: variance due to system instability
σ_{pm}^2: variance due to platform motion (tranlational)

The spectrum broadening due to system instability will be analyzed in the next section. The translational broadening dependents on platform velocity, antenna depression angle as well as azimuth angle, pulsewidth, altitude, and so forth.

MTI design for an airborne radar presents a unique challenge due to platform motion. An elaborate motion compensation is required: TACCAR technique plus DPCA[2]. The subject is beyond the scope of this chapter and will not be presented here. A shoreline fixed radar for harbor surveillance or maritime control has to counter the sea clutter with nonzero mean velocity. A seaborne radar, on corvette or frigate for instance, must compensate for the cruising velocity of the vessel (crab-walk included), and the platform must be stabilized for pitch and roll through gimbals mechanism.

A diagram of the clutter situation and a distribution profile for land, sea, and precipitation clutter is shown in Figure 11.16.

2. DPCA stands for displaced phase centered antenna, a variant of the single-plane monopulse antenna with corporate feed network.

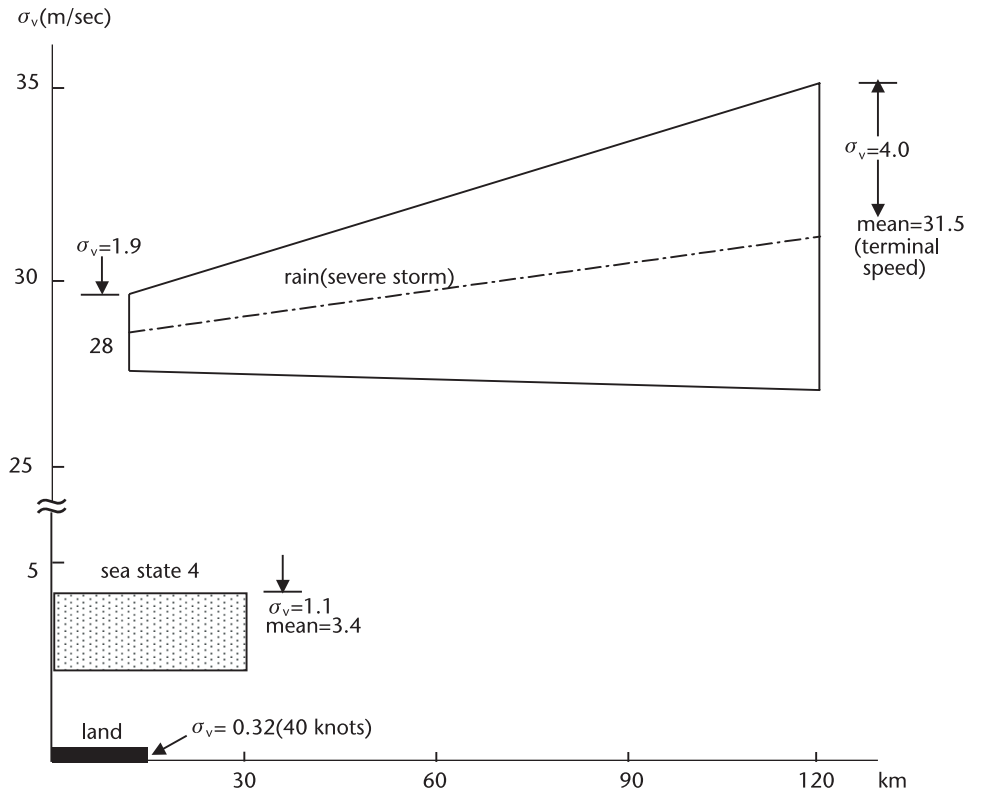


Figure 11.16 Clutter distributions. (*From:* [5, p. 16]).

11.6 Limitations Due to System Instability

The MTI principle is based on the detection of phase change caused by Doppler shift of moving targets, and all cancellers are preceded by phase detector. Unfortunately phase changes are also originated by system instability and we are never able to separate the true source of phase change from the system instability. The system instability imposes limitations on the theoretically achievable improvement factor.

We list the sources of system instabilities and the limitation on the improvement factor [4] (see Table 11.2). We also give numerical examples of the system stability requirements.

Δf: interpulse transmitter frequency shift
$\Delta\varphi$: interpulse phase shift
Δt: pulse-to-pulse timing jitter
Δw: pulsewidth jitter
ΔA: pulse-to-pulse amplitude difference
A: pulse amplitude
τ: transmitted pulsewidth
T: two-way transit time to and from target

Table 11.2 Sources of System Instabilities and Limitation on the Improvement Factor

Source of Instability	*Limit on Improvement Factor*
1. Δf	$I = 20 \log\left(\frac{1}{\pi \Delta f \tau}\right)$
2. $\Delta\varphi$	$I = 20 \log\left(\frac{1}{\Delta\varphi}\right)$
3. Δf (stalo and coho)	$I = 20 \log\left(\frac{1}{2\pi \Delta f T}\right)$
4. Δt	$I = 20 \log\left(\frac{\tau}{\sqrt{2}\sqrt{\tau B}\ \Delta t}\right)$ (Time-bandwidth product $\tau B=1$ for uncoded transmit signal)
5. Δw	$I = 20 \log\left(\frac{1}{\Delta w \sqrt{\tau B}}\right)$
6. ΔA	$I = 20 \log\left(\frac{A}{\Delta A}\right)$

Consider an S-band (3,000 MHz) radar with a 1.0-μs uncoded pulse, and the requirement that no one of six sources of instability limit the improvement factor to less than 40 dB.

1. Transmitter frequency shift must be less than

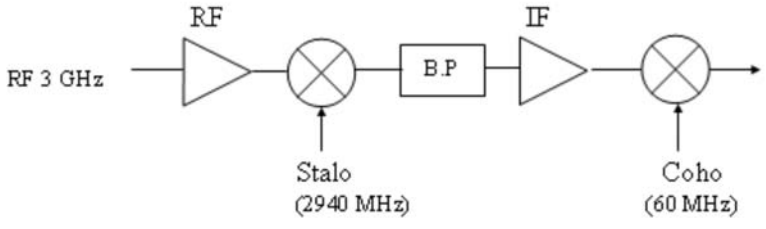
$$\frac{40}{20} = \log\left(\frac{1}{\pi \Delta f \tau}\right), \quad 100 = \left(\frac{1}{\pi \Delta f \tau}\right), \quad \Delta f \le \frac{1}{\pi \tau (100)} = 3{,}100 \text{ Hz}$$

$$\frac{\Delta f}{f} = \frac{3100}{3E+9} \le 1.0E-6, \quad \text{one part in one million}$$

2. Transmitter phase shift must be less than

$$100 = \frac{1}{\Delta\varphi}, \quad \Delta\varphi = \frac{1}{100}\text{rad} \le 0.57 \text{ degree}$$

3. Stalo and Coho frequency shift must be less than, assuming the target is located at 100 km, the Stalo frequency is 2,940 MHz and Coho frequency at 60 MHz



$$100 = \frac{1}{2\pi\Delta fT}, \quad \Delta f = \frac{1}{2\pi T(100)} \approx 2.4 \text{ Hz}$$

$$\text{Stalo}: \quad \frac{2.4}{2.94E9} \leq 8.0E-10$$

$$\text{Coho}: \quad \frac{2.4}{6E7} \leq 4E-8$$

4. Pulse-to-pulse timing jitter must be less than

$$100 = \frac{\tau}{\sqrt{2}\sqrt{\tau B}\,\Delta T} \quad \frac{\Delta t}{\tau} \leq 0.7\% \text{ for uncoded pulse}$$

5. Pulsewidth jitter must be less than

$$100 = \frac{\tau}{\Delta w\sqrt{\tau B}} \quad \frac{\Delta w}{\tau} \leq 1\%$$

6. Pulse amplitude jitter must be less than

$$100 = \frac{A}{\Delta A} \quad \frac{\Delta A}{A} \leq 1\%$$

Readers are encouraged to compute the system stability requirement for improvement factors of 50 dB or higher and appreciate the stringent stability requirements. Out of six, the Stalo stability is the most difficult one to meet.

11.7 A/D Converter Quantization Noise

The quantization noise produced by an A/D converter limits the theoretically predicted improvement factor. When an A/D converter has N bits resolution and the phase detector output range is [+1, –1], the quantized interval ΔN is

$$\Delta N = \frac{2}{(2^N - 1)}$$

The standard deviation of the quantized noise, assuming a unit uniform probability density function, is

$$\sigma = \frac{1}{\sqrt{12}} \frac{2}{(2^N - 1)}$$

The standard deviation is further broadened due to the pulse-to-pulse deviation by a factor of $\sqrt{2}$, and there is another factor of $\sqrt{2}$ introduced by the reduction in average signal amplitude passing through a phase detector [3, p. 11–50] (see Table 11.3).

$$I = 20\log(1/\sigma) = 20\log[\sqrt{3/4}(2^N - 1)]$$

Table 11.3

Number of bits	*Limit on Improvement Factor (dB)*
4	26.27
5	28.58
6	34.74
7	40.83
8	46.88
9	52.92
10	58.95
11	64.97
12	70.99

It appears that an A/D converter with 10 bits or higher resolution would not limit the improvement factor less than 59 dB.

11.8 Clutter Map

So far we have studied the pulse-to-pulse canceller techniques, the temporal processing. When an adequate data storage memory is available, we may consider an alternative, scan-to-scan cancellation technique, the spatial processing.

The clutter map would reject signals that have not moved from the previous scan to the present scan; clutter is stationary between successive scans. Spatial processing has no periodic blind speed as in temporal processing; therefore, we need no staggered PRF management. This simplifies the system operation, however, unfortunately the clutter mapping works only for a stationary fixed position radar and only when the resolution cell is larger than the mean velocity of the clutter, a coarse resolution radar. For a seaborne radar an elaborate coordinate transform in vessel velocity and the correction for yaw of the platform are required. The performance may be unsatisfactory.

11.9 Conclusion

TACCAR is a nonlinear video processing, and it cannot be cascaded as we cascade a single-delay canceller to implement a double-delay or a triple-delay canceller. Instead of TACCAR we should consider an IF vector canceller, which can be cascaded when the improvement factor must be increased for nonstationary clutter.

For the best MTI performance, the system engineer should choose on the lowest possible carrier frequency (the largest wavelength λ) within the constraint of antenna size (or weight), and the highest f_{PRF} permissible for unambiguous range. To wit, the engineer should push an operating point $\sigma_v/\lambda f_{PRF}$ the lowest possible to the left in Figure 11.12.

A bank of Doppler filters has been analyzed for clutter rejections [13, 14]. In particular it was demonstrated successfully for clutter rejection through FFT

List of Programs

(1) DELAY_NR.CPP	Frequency response of single and double delay line canceller, nonrecursive.
(2) DELAY_N3.CPP	Frequency response of triple canceller, nonrecursive, with forward gain k.
(3) DELAY_R1.CPP	Frequency response of single canceller, recursive, with feedback gain k.
(4) DELAY_R2.CPP	Frequency response of double canceller, recursive, with feedback gain k.
(5) DELAY_R3.CPP	Frequency response of triple canceller, with feedback gain k_1 and k_2.
(6) DELAY_C3.CPP	Frequency response of triple recursive canceller, coefficients of 3rd order Chebyshev polynomial.
(7) IMPRVMT.CPP	Computes the improvement factor, the mean velocity of clutter is zero.
(8) REDUCTN.CPP	Computes the reduction in improvement factor for nonzero mean velocity of clutter.
(9) MTI_CNR.CPP	Computes the improvement factor, mean velocity = 0.0, limited by clutter-to-noise ratio.
(10) MTI_3S2D.CPP	Computes the frequency response of a triple-stagger with double-delay-line canceller, the stagger rario=3/5.

processing using eight input samples for radix-2 algorithm [15]. The trend in the industry leans toward the Doppler filter banks since custom-made microprocessor for FFT/IFFT processing are readily available, though expensive.

References

[1] Lindon, D. A., and B. D. Steinberg, "Synthesis of Delay Line Network," *IRE, Trans.* ANE-4, 1957.

[2] Barton, D. K., *Radar System Analysis*, New York, NY: Prentice-Hall, 1964.

[3] Skolnik, M. I., *Introduction to Radar Systems,* New York, NY: McGraw-Hill, 1962.

[4] Shrader, W. W., in *Radar Handbook,* M. I. Skolnik (ed.), New York, NY: McGraw-Hill, 1962, Chapter 17.

[5] Scheleher, D. C., *MTI and Pulsed Doppler Radar,* Norwood, MA: Artech House, 1991.

[6] Zrnic, D. S., "Spectral Moment Estimation from Correlated Pulse Pairs," *IEEE, Trans. AES*, Vol. 13, No. 4, July 1977.

[7] Novik, L. R., and K. M. Glover, "Spectral Mean and Variance Estimation via Pulse Pair," *16th Radar Meteorology Conf. Record,* April 1975.

[8] Bergerm, T., and and Groginsky, H. L., "Estimation of the Spectral Moments of Pulse Trains," *International Conf. of Information Theory*, Tel Aviv, Israel, 1973.

[9] Benham, F. C., Groginsky, H. L., Soltes, A. S., et al., *Pulse Pair Estimation of Doppler Spectrum Parameters,* Raytheon Co., Final Report, Contract No. F-19628-71-C-0126, 1972.

[10] *Pulse Pair Processor, Equipment Information Report,* Raytheon Co., Contract No. F-19628-72-C-0293, May 1974.

[11] Andrew, G. A., *Airborne Motion Compensation Technique, Evaluation of TACCAR*, Naval Research Lab., Report 7407, April 1972.

[12] Dickey, F. R., Jr., M. Labitt, and F. M. Staudaher, "Development of Airborne Moving Target Radar for Long Range Surveillance," *IEEE, Trans.*, Vol. AES-27, No. 6, November 1991.

[13] Andrew, G. A., *Optimum Radar Doppler Processors*, Naval Research Laboratory, NRL FR-7727, May 1974.

[14] Andrew, G. A., *Performance of Cascaded MTI Coherent Integration Filters in Clutter Environment*, Naval Research Laboratory, NRL Report 7533, March 1973.

[15] Irabu, T., E. Kiuchi, T. Hagisawa, et al., "On the Performance of a Two Dimensional Clutter Rejection Filter," *IEEE International Radar Conference Record*, 1980, pp. 311–316.

CHAPTER 12

Miscellaneous Program Routines

This chapter contains 22 miscellaneous program routines we have used on occasion. Some of them do not fit under the proper chapter titles, and some are preparatory works on certain programs.

SORT_BUB.CPP	Sorting an array of data in ascending order analogous to air bubbles floating up to the surface, sorting-by-bubble.
SORT_SEL.CPP	Another sorting routine of data array, sorting-by-selection.
SORT_INX.CPP	Third sorting routine, an array of data sorted in ascending order without disturbing the sequence of the original array, sorting-by-index.
INSERT.CPP	A program routine for an arbitrary data to be inserted in an array.
SEARCH_L.CPP	This program searches an array linearly to find data we have designated (target data). When the target data is found this program identifies the index of the data (location).
SEARCH_B.CPP	Similar to above search program except the search is executed after the array is partitioned into two subarrays repetitively until the data is found, search-by-binary is faster than search-by-linear.
PRIME.CPP	A program that generates the prime numbers.
FCTRIAL.CPP	Factorial, computes factorial of an integer, K!
K!_EXACT.CPP	Computes the exact value of k! when k<=30.
K!_APPRX.CPP	Computes an approximate of k! when k>=20.
PERMUTAT.CPP	Computes permutation, given two integers n and k, n>k. P(n, k) = n!/(n–k)!
COMBINAT.CPP	Computes combination, given two integers n and k, n>k. C(n, k) = n!/k!/(n–k)!

The next three programs demonstrate how to find the root(s) of a function, f(x) = 0. Actually we try to find the crossing point(s) of two functions, $f_1(x)$ and $f_2(x)$. The first two examples employ the "bisection" method and the third example the Newton-Raphson method.

ROOT_F1.CPP Suppose we are given a function f(x) = sinh(x-2.5) – 5.0 exp (–x/2), and we would like to find the root of f(x) = 0.

Two functions are

$$f_1(x) = \sinh(x–2.5), \quad f_2(x) = 5.0 \exp(–x/2)$$

The root (crosspoint) is located somewhere between 3.0 and 4.0 (Figure 12.1). The bisection method is used to find the function values at two test points, x_{low} and x_{high}, where the function changes the sign from negative to positive or positive to negative.

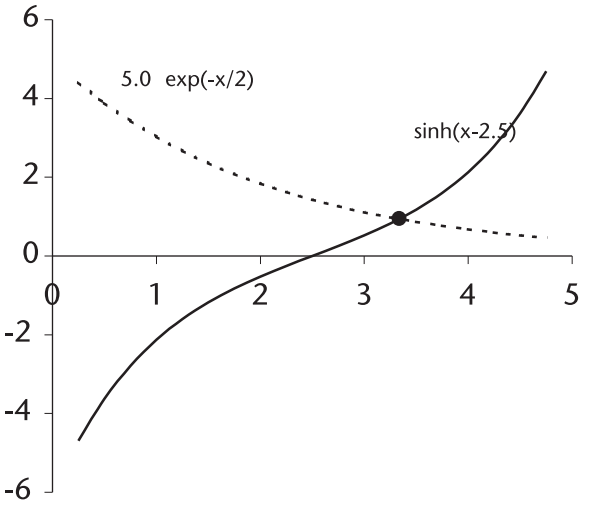
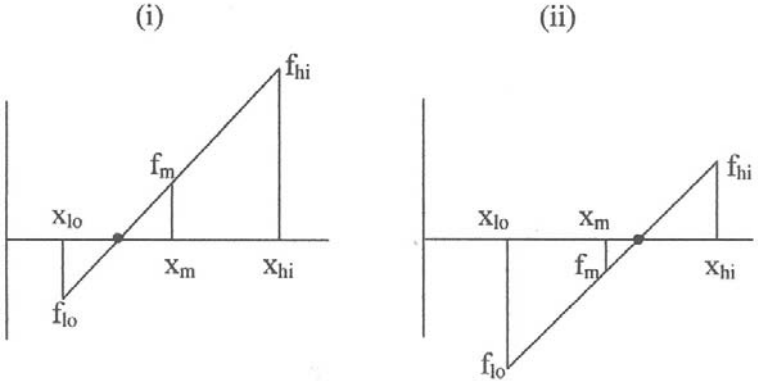


Figure 12.1 Crosspoint between an exponential and hyperbolic sine function.

$$f_{lo} = \sinh(x_{lo} - 2.5) - 5.0 \exp(-x_{lo}/2) \quad f_{lo} \text{ is negative}$$

$$f_{hi} = \sinh(x_{hi} - 2.5) - 5.0 \exp(-x_{hi}/2) \quad f_{hi} \text{ is positive}$$

When we evaluate the function at the midpoint x_m, we shall have two distinctive possibilities:



In case (i) we replace (x_{hi}, f_{hi}) by (x_m, f_m) so that the function will change the sign. In case (ii) we replace (x_{lo}, f_{lo}) by (x_m, f_m). We shall repeat the replacement until the absolute difference $| f_{hi} - f_{lo}|$ reaches some small tolerance (i.e., 1.0E-6 or 1.0E-7). The root of the function is x_m, found at the last iteration. The number of iterations N required is given by

$$N = \log_2(\Delta/\varepsilon) \quad \text{where} \quad \Delta = |x_h - x_l|$$

$$\varepsilon = \text{tolerance specified}$$

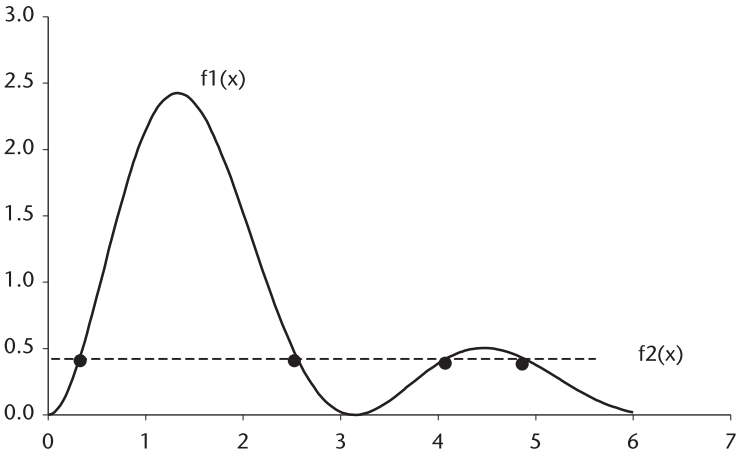


Figure 12.2 Multiple roots.

Caution: If you set the tolerance too small (i.e., 1.0E-16 for float), the round-off errors play a trick on you and the convergence would never succeed.

ROOT_F2.CPP This is second demonstration program by the bisection method when a given function has multiple roots (Figure 12.2).

$$f(x) = 5\ \exp(-x/2)\ \sin^2 x - 0.4$$

$$f_1(x) = 5\ \exp(-x/2)\ \sin^2 x$$

$$f_2(x) = 0.4$$

ROOT_F3.CPP The root(s) of function can be found more efficiently by the Newton-Raphson method if the derivative of the function is easily found (sometimes it is difficult). The iterative procedure to find the root(s) is based on Taylor series expansion of given function:

$$f(x + \varepsilon) = f(x) + \varepsilon f'(x) + \varepsilon^2 f''(x) + \varepsilon^3 f'''(x) + \ \ldots$$

When ε is very small, the terms of ε^2 and the higher power terms can be ignored. Then

$$f(x + \varepsilon) \approx f(x) + \varepsilon f'(x)$$

As ε becomes smaller and smaller, and for a well-behaved function, we can see that

$$f(x+\varepsilon) \rightarrow f(x) + \varepsilon f'(x) = 0$$
$$\varepsilon = -f(x)/f'(x)$$

Thus

$$x + \varepsilon = x - f(x)/f'(x)$$

The root is found by using above relationship iteratively:

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

Caution: When the derivative $f'(x)$ is zero or nearly zero, the Newton-Raphson method would fail. Two functions are parallel (or nearly parallel to each other).

ISUB0(X). CPP This program computes the modified Bessel function of the first kind, zero order.

$$I_0(x) = 1 + \sum_{k=1}^{\infty} [(x/2)^k/k!]^2 = \sum_{k=0}^{\infty} [(x/2)^k/k!]^2$$

GAM_FUNC.CPP This program computes the Gamma function.

$$\Gamma(x_\infty) = \int_{x=0}^{\infty} \frac{e_{-x}\, x^k}{k!}\, dx = \sum_{x=0}^{\infty} \frac{e^{-x}\, x^k}{k!}$$

INC_GAMM.CPP This program computes the incomplete Gamma function.

$$\Gamma(x_b) = \int_{0}^{y_b} \frac{e^{-x}\, x^k}{k!}\, dx = \sum_{x=0}^{y_b} [(e^{-x}\, x^k)/k!]$$

MAX_MIM.CPP This program demonstrates finding the maximum-minimum data in an array by function "Template" when the array data is a mixture of integer, float, or double.

DBL_SUM.CPP This program computes the product of two summations. One summation is indexed in "m", the other in "n." When n = m, the second summation will be excluded (see Chapter 5, Section 5.4.).

$$I = \sum_{m=0}^{5} f_m \sum_{n=0\ n\neq m}^{5} f_n$$

K(m)_JAC.CPP Computes the Jacobian elliptic function.

K(m)_ELP.CPP Computes the elliptic integral when the modulus m is given; 0<=m<1.0.

Readers are encouraged to add their favorite routines to this chapter for future reference.

Index

T