To the Graduate Council:

I am submitting herewith a thesis written by Robert Alan Weddle entitled "A Receiver Model for Improved Target Detection Under High Clutter Conditions for Radar Guided Missiles." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Dr. Daniel B. Koch, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and
Dean of The Graduate School

A RECEIVER MODEL FOR IMPROVED TARGET DETECTION UNDER HIGH CLUTTER CONDITIONS FOR RADAR GUIDED MISSILES

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Robert Alan Weddle

December 1996

DEDICATION

This thesis is dedicated to my loving wife and daughter,

Shonna and Chelsey Weddle,

whose love and support provided the motivation necessary to reach my desired goal.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Daniel B. Koch, for his cooperation and guidance, as well as the other committee members, Dr. Michael J. Roberts and Dr. Herbert Neff for their comments and assistance. I would also like to thank my colleagues at Raytheon Company and at Extended Technologies for their comments, suggestions, and assistance over the past year. Additionally, I would like to express my thanks to my wife, Shonna, for her understanding and endurance over the past six years. Without her support, none of this would be possible. Lastly, I would like to give thanks to God for His anointing and inspiration, without which I can do nothing. All the glory and honor belong to Him.

ABSTRACT

A receiver model for improved target detection under high clutter conditions for radar guided missiles is developed. Within this paper, a typical receiver model is developed and evaluated to illustrate the problem that exists with this type of receiver. Test stimuli are used to evaluate the receiver model. An improvement is proposed for the receiver which eliminates the need for a limiter function. The improved receiver is evaluated using the same test stimuli. A comparison is made between the two receiver models to point out the improvements.

TABLE OF CONTENTS

Chapter **Page**

1. INTRODUCTION **1**

1.1. Radar History 1
1.2. Extraction of Information 2
1.3. Problem Statement 3
1.4. Thesis Organization 5

2. SEMIACTIVE RADAR GUIDED MISSILE SYSTEMS **6**

2.1. Building Blocks of a Semiactive Radar Guided Missile System 6
2.2. The Doppler Effect 8
2.3. Radar Clutter 11

3. A SEMIACTIVE RADAR GUIDED MISSILE SYSTEM MODEL **15**

3.1. Derivation of the Radar Range Equation 16
3.2. Description of the Simulated Radar System 18
3.3. Simulation Methodology 26

4. SIMULATION OF A GENERAL MISSILE RECEIVER **36**

4.1. Receiver Model Development 36
4.2. Simulation Results 38

5. SIMULATION OF AN ALTERNATIVE MISSILE RECEIVER **46**

5.1. Receiver Model Development 46
5.2. Simulation Results 49

6. CONCLUSION **53**

6.1. Comparison of Receiver Models 53
6.2. Future Work 54

LIST OF REFERENCES **55**

APPENDICES **57**

A. Program Listing of General Receiver Model 58
B. Program Listing of Alternate Receiver Model 88

VITA **121**

LIST OF FIGURES

Figure **Page**

2-1: Basic Semiactive Radar Guided Missile System 7

2-2: Semiactive Radar Guided Missile Geometry 10

2-3: Semiactive Guided Missile Signal Spectrum with Fixed Illuminator 13

2-4: Semiactive Guided Missile Signal Spectrum with Moving Illuminator 13

3-1: Illuminator Antenna Pattern 21

3-2: Front and Rear Receiver Antenna Pattern 23

3-3: Clutter Patch Relation to System Geometry 25

3-4: Glistening Region Relation to Clutter Patch 27

3-5: System Geometry for the High Altitude Condition 28

3-6: System Geometry for the Low Altitude Condition 32

3-7: Front Antenna Spectrum for High Altitude Condition 31

3-8: Rear Reference Spectrum for High Altitude Condition 31

3-9: Front Antenna Spectrum for Low Altitude Condition 34

3-10: Rear Reference Spectrum for Low Altitude Condition 34

4-1: General Receiver Model Block Diagram 37

4-2: Local Oscillator Spectrum 37

4-3: Downconverted Front Antenna Spectrum for High Altitude Condition 39

4-4: Downconverted Rear Antenna Spectrum for High Altitude Condition 39

Figure **Page**

4-5: Limited Rear Spectrum for High Altitude Condition 41

4-6: Baseband Doppler Video Spectrum for High Altitude Condition 41

4-7: Downconverted Front Antenna Spectrum for Low Altitude Condition 42

4-8: Downconverted Rear Reference Spectrum for Low Altitude Condition 42

4-9: Pure Baseband Doppler Video Spectrum for Low Altitude Condition 43

4-10: Limited Rear Spectrum for Low Altitude Condition 44

4-11: Baseband Doppler Video Spectrum for Low Altitude Condition 45

5-1: Alternative Receiver Model Block Diagram 47

5-2: Front Tuned LO Spectrum for High Altitude Condition 47

5-3: Front Tuned LO Spectrum for Low Altitude Condition 48

5-4: Downconverted Front Antenna Spectrum for High Altitude Condition 50

5-5: Baseband Doppler Video Spectrum for High Altitude Condition 50

5-6: Downconverted Front Antenna Spectrum for Low Altitude Condition 51

5-7: Baseband Doppler Video Spectrum for Low Altitude Condition 52

1. INTRODUCTION

Radar is an electromagnetic sensing system that can detect and locate objects beyond sight. It operates by transmitting a waveform and detecting an echo signal. The intent for radar is to extend one's senses for observing the environment. Radar is not a substitute for the human eye, but is capable of doing what the eye cannot do. Radar can see through conditions the human eye cannot, such as darkness, haze, fog, rain, and snow. Radar is also able to measure the distance or range to an object, which is perhaps its most important attribute [1-8].

The name *radar* is a contraction of the words *radio detection and ranging*. The name reflects the emphasis placed on its early development. It was developed as an early warning detection device for hostile aircraft. Although modern radar systems can extract more information from the return signal, the measurement of range is still the most important function of a radar system.

1.1. Radar History

The basic principle of radar detection was demonstrated by Heinrich Hertz in 1888 and was later tested in Germany in the early 1900s. However, radar detection was not considered a full-fledged technology until World War II. The threat of a bomber aircraft strike and the lack of an early warning system required individuals in Europe and

the United States to perfect radar. Great Britain was first in deploying an operational radar system (the Chain Home in 1937). Using available components from radio technology, the Chain Home radar operated at a frequency of about 25 MHz [2], [4], [6-8].

With the end of World War II, many of the war-time developments were immediately applied to civil uses. A continuous succession of technological advances has led to a further broadening of radar applications, both military and civil. The major civil areas of radar application are air traffic control, aircraft navigation, ship safety, space, remote sensing and law enforcement. Many of the civilian applications of radar are also used by the military. The traditional role of radar for the military application has been for surveillance, navigation, and for the control and guidance of weapons.

1.2. Extraction of Information

Although radar was developed for the detection and ranging of a target, a radar is capable of providing more information. Detection of a target means that the radar is aware of its presence. However, in most practical cases, the radar system desires to know more about the target's location in space and its nature. The extraction of useful target information is therefore an important part of radar operation [8].

The more information that is known about a target prior to detection, the more efficient the radar system will be. For example, if the target location were known, the

antenna could be pointed in the necessary direction. This would speed the detection process by eliminating the need to search for the target in empty space. Or, if the relative velocity were known, the receiver could be pre-tuned to the correct frequency, eliminating the need to search the frequency band over which the doppler shift might occur. The doppler frequency shift is of importance for targets viewed with continuous wave radar [8].

When the doppler frequency shift is large and unknown, an effective method for detection of the unknown doppler shift is to employ a bank of filters each matched to a different value of the unknown frequency so as to cover the entire range of uncertainty. When a signal appears somewhere within the doppler baseband, the output of one of the filters will be greater than the rest. This not only results in efficient detection, but the doppler frequency shift is found as well [8].

1.3. Problem Statement

Radar guided missiles have become one of the most widely used applications of radar. However, not much has been published since most of the detailed data is classified. This paper will discuss the performance of a typical semiactive continuous wave (CW) radar guided missile system in the presence of uncontrollable environmental noise called clutter.

Radar guided missile systems operating in high clutter scenarios have difficulty detecting the target. When the clutter levels are near the target return signal level, such as in low altitude scenarios, locking on the target instead of an image reflected from the earth's surface can be a challenging undertaking. A typical radar system receiver uses a rear reference signal to mix with the front target spectrum to produce a baseband video spectrum which contains the target doppler information. This paper shows how this system will exhibit target detection problems in high clutter scenarios. A modified receiver model is then developed which eliminates the effects of the rear clutter component. By eliminating the rear clutter element, the resulting target doppler spectrum shows an improvement.

A typical semiactive missile radar system receiver was modeled and simulated to evaluate its performance. The missile receiver model performs the mixing operations necessary to produce a baseband doppler spectrum. The doppler spectrum contains the necessary information for target detection and missile guidance. System noise variables such as local oscillator (LO) phase noise, were included in the receiver simulation. An alternate receiver model was also developed and simulated. The rear reference and front doppler spectra were generated using a clutter derived noise generation program. The rear spectrum contains the illuminator, rear clutter, and the target. The target has greatly reduced power since the signal is detected by a sidelobe of the rear antenna pattern. The front spectrum contains the target, an image, mainlobe clutter, and illuminator spillover. The illuminator spillover power is reduced, similar to the target in the rear spectrum, due

to the detection of the illuminator in a sidelobe of the front antenna pattern. This simulation was not a guidance simulation, rather, it calculated a doppler spectrum for one particular point in flight.

Simulation of the two semiactive radar guided missile receiver models was performed with and without clutter in the front and rear input spectra to illustrate performance of each receiver under clear environments and under the effects of high clutter.

1.4. Thesis Organization

This paper is divided into six chapters. Chapter 2 introduces the concepts of radar as it pertains to the guidance of missiles. A mathematical model for a semiactive continuous wave radar guided missile receiver is developed in Chapter 3. This model is simulated for a system performance baseline characterization and the results are discussed in Chapter 4. Chapter 5 contains the alternate receiver model development and the system simulation results. Conclusions are presented in Chapter 6.

2. SEMIACTIVE RADAR GUIDED MISSILE SYSTEMS

Homing guidance of a radar guided missile provides the highest accuracy of all available guidance methods. Homing systems can be characterized on the basis of the source of the sensed radar energy as passive, semiactive, and active. Passive homing tracks energy originating from the target (e.g., jamming or radar transmissions). An active homing system is a self-contained radar that transmits its own radar energy at the target and tracks the target-reflected energy. The semiactive system includes an external radar that illuminates the target while the missile receives and tracks the target-reflected energy to extract guidance information. This paper will deal with the semiactive radar guided missile system [4], [8].

2.1. Building Blocks of a Semiactive Radar Guided Missile System

Radar guided missiles have become one of the most widely used applications of radar. The basic continuous wave (CW) semiactive system is conceptually illustrated in Figure 2-1. The illuminator maintains the target within its radar beam throughout the engagement. The missile receives the target-reflected illumination in its front antenna and a sample of the directly received illumination in its rearward-looking reference (rear) antenna. The front and rear signals are coherently detected against each other, resulting in a spectrum which contains the doppler-shifted target signal at a frequency roughly proportional to closing velocity. A narrowband frequency tracker searches the spectrum,

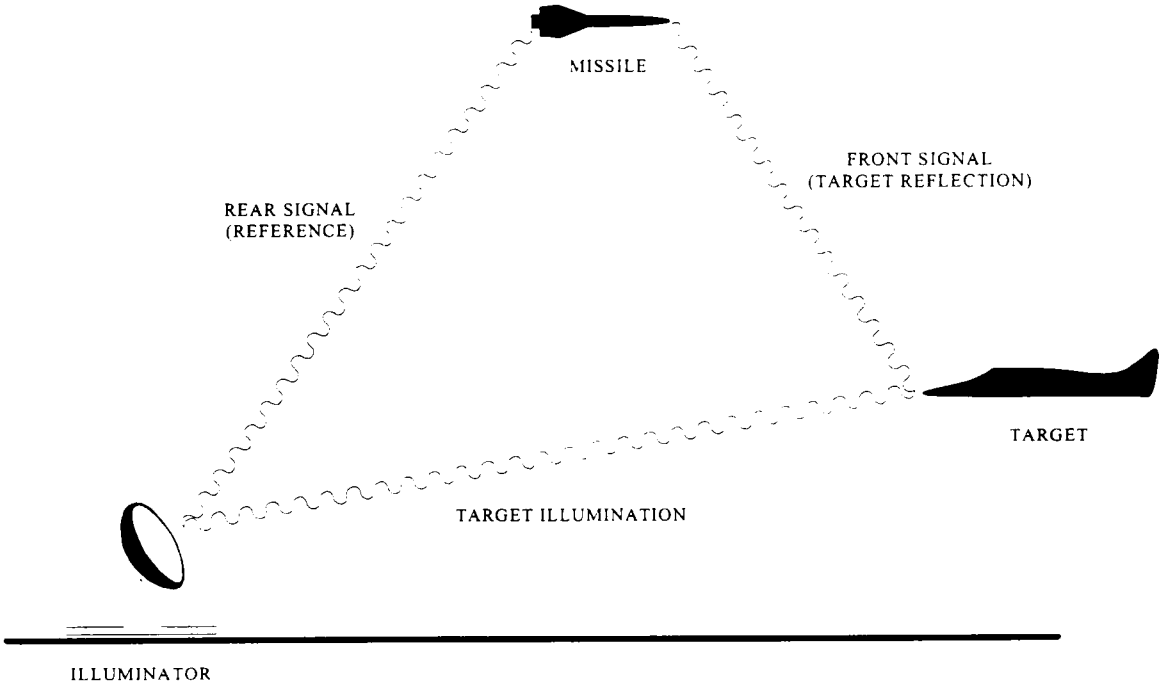


Figure 2-1: Basic Semiactive Radar Guided Missile System

locks onto the target return, and extracts guidance information from it. The use of CW provides the capability of discriminating against clutter on the basis of doppler frequency and thus allows low-altitude operation [8].

The extraction of information from the returned echoes of a radar transmitter is performed by analyzing the returned signals in terms of the detected frequency changes or phase changes of the signal. When a signal is reflected from a target moving in a relative radial direction, an effective shift in frequency is experienced. This frequency shift is known as the *doppler effect*, and the magnitude of the frequency shift is measured by determining the frequency difference between the transmitted frequency and the frequency of the returned signal. This measurement is made in the frequency domain and is the means of determining the relative radial velocity of the moving target in relation to the position of the radar [4].

2.2. The Doppler Effect

The doppler effect is that effect which gives an apparent change in frequency if either the source of radiating energy or the reflecting target is in radial motion relative to the other. The doppler frequency shift is a function of the relative radial velocity and the oscillating frequency and their relationship is shown in the formula [8]:

$$f_d = \frac{2V_r}{1} = \frac{2V_r f_o}{c} \qquad (2\text{-}1)$$

where:

f_d = Doppler Frequency Shift

V_r = Relative Velocity

λ = Wavelength

f_o = Transmitted Frequency

c = Velocity of Propagation

Note that the relative radial velocity parameter is important in measuring doppler frequency shift, since a vehicle moving at a high velocity in a circle about a fixed radiating source does not create a doppler frequency shift, as the distance between the two objects does not change. Further, if the vehicle and the radar are moving in a straight line at the same velocity and in the same direction there is also no doppler effect between the two as there is no change in range [4], [7], [8].

The geometry for the doppler frequency relationships in a semiactive radar guided missile system is shown in Figure 2-2. The radial velocity components of the three system elements contribute to the doppler shift of the received signal. Note that the coordinate system shown is relative to the plane containing all three system elements: the illuminator, missile, and target [8, 10]. The rear reference doppler frequency is a function of the radial velocity of the missile with respect to the illuminator. The front target doppler frequency depends on the radial velocities of the illuminator (V_I), missile (V_M),

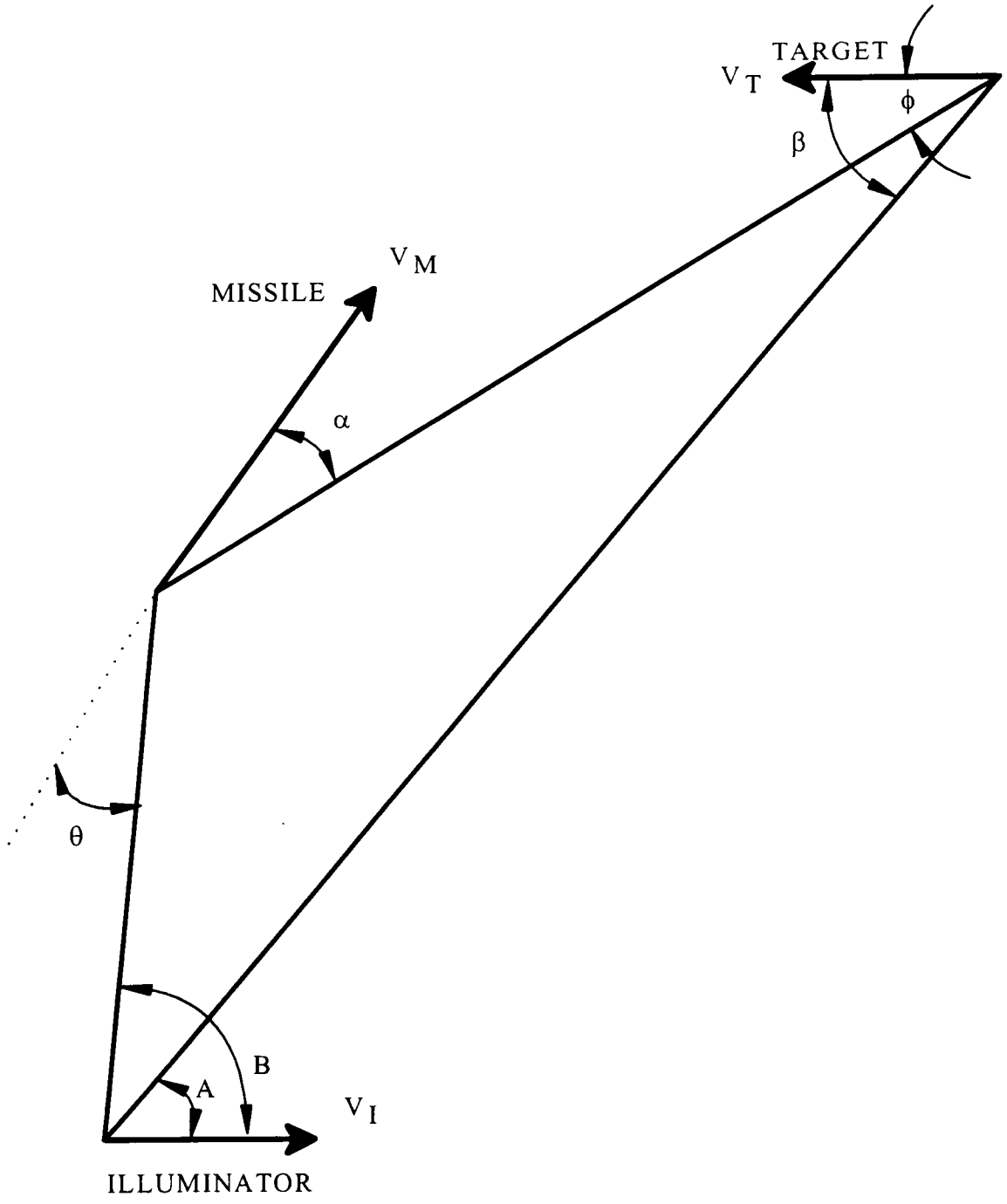


Figure 2-2: Semiactive Radar Guided Missile Geometry

and target (V_T). The resulting spectrum, when the front signal is mixed against the rear, is the difference of the two [8].

$$f_{rear} = -\frac{f_o}{c} V_M \cos q + \frac{f_o}{c} V_I \cos A \tag{2-2}$$

$$f_{front} = \frac{f_o}{c} (V_T \cos f + V_T \cos b + V_M \cos a + V_I \cos B) \tag{2-3}$$

$$f_d = f_{front} - f_{rear} \tag{2-4}$$

Substitution of Equations 2-2 and 2-3 into equation 2-4 yields

$$f_d = \frac{f_o}{c} (V_M \cos q + V_M \cos a + V_T \cos f + V_T \cos b + V_I \cos B - V_I \cos A) \tag{2-5}$$

2.3. Radar Clutter

In addition to the target, large interfering signals exist within the spectrum of interest. These large interfering signals for radar guided missiles are normally caused by unwanted ground return. Ground return falls into two categories: mainlobe return and sidelobe return. However, all is considered clutter for the radar guided missile. The clutter doppler can be calculated by using the same equations used for the target. This can be accomplished by assuming that the clutter patch is the target with a velocity V_T = 0.

A typical clutter spectrum for a ground-to-air missile is shown in Figure 2-3. The system contains a fixed illuminator, thus the minimum doppler frequency is limited by the following equation [8]:

$$f_{d_{FEED}} = f_o - \frac{f_o}{c} V_M \tag{2-6}$$

This frequency is called *feedthrough.* The velocity of the missile creates a doppler return by the main-lobe of the antenna called *main-lobe* clutter (MLC). The MLC frequency is marked *clutter* in Figure 2-3 and is determined by the following equation [8]:

$$f_{d_{MLC}} = f_o + \frac{f_o}{c} V_M \tag{2-7}$$

The *sidelobe* clutter spectrum extends from feedthrough up to MLC. The target frequency is determined by the following equation [8]:

$$f_{d_T} = f_o + \frac{f_o}{c}(2V_T + V_M) \tag{2-8}$$

Figure 2-4 contains a spectrum which corresponds to a moving illuminator or an air-to-air missile system. The frequency representing feedthrough is limited by the following equation [8]:

$$f_{d_{FEED}} = f_o - \frac{f_o}{c}(V_M - V_I) \tag{2-9}$$

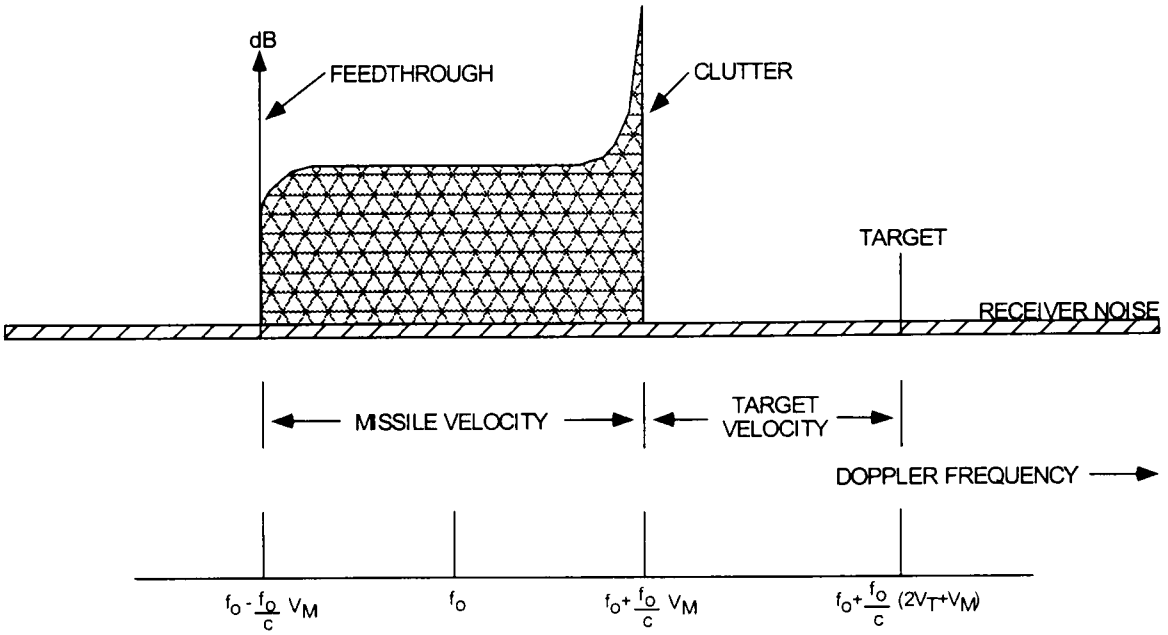


Figure 2-3: Semiactive Guided Missile Signal Spectrum with Fixed Illuminator

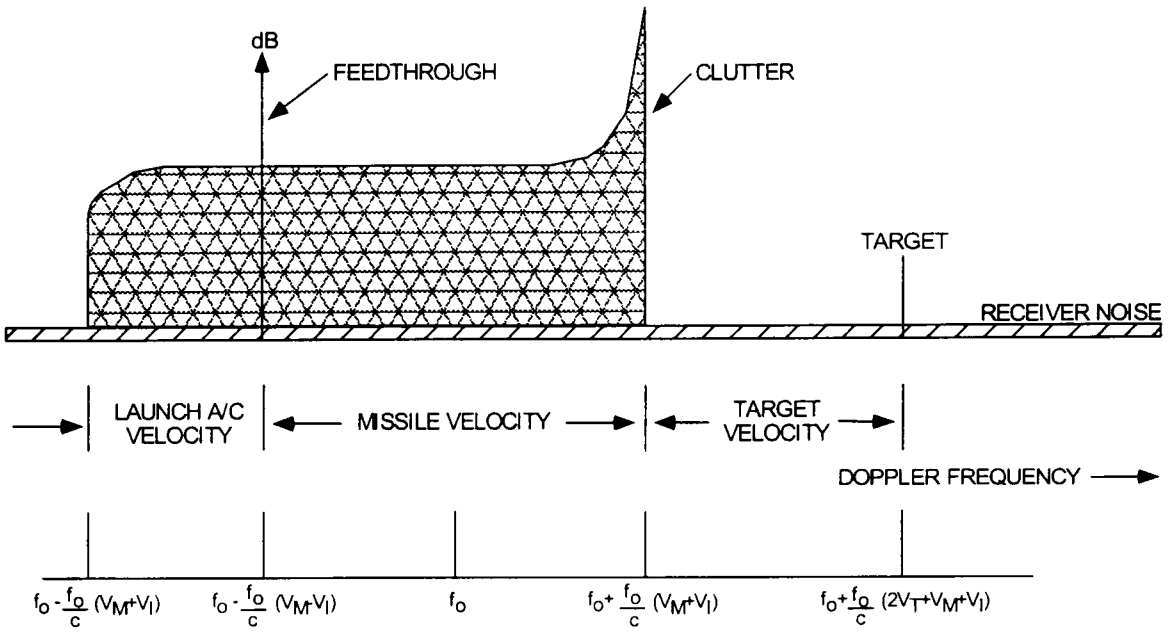


Figure 2-4: Semiactive Guided Missile Signal Spectrum with Moving Illuminator

Notice that the clutter spectrum extends below feedthrough. This is explained by the illuminator's backlobe which produces a return from the clutter patch behind the aircraft. The minimum doppler frequency is then given by the following equation [8]:

$$f_{d_{\min}} = f_o - \frac{f_o}{c}(V_M + V_I) \tag{2-10}$$

The MLC frequency is now given by the following equation [8]:

$$f_{d_{MLC}} = f_o + \frac{f_o}{c}(V_M + V_I) \tag{2-11}$$

The target frequency is determined by the following equation [8]:

$$f_{d_T} = f_o + \frac{f_o}{c}(2V_T + V_M + V_I) \tag{2-12}$$

For simplicity, discussions will be limited to systems with a fixed illuminator [8].

3. A SEMIACTIVE RADAR GUIDED MISSILE SYSTEM MODEL

Any radar system model begins with the development of the radar range equation. A radar system uses the phenomenon of a body reflecting high-frequency electromagnetic energy to obtain an echo return to a high frequency transmission. The detection performance of any radar depends upon many factors that are related in what is traditionally known as the radar equation [8]. There are a number of forms of this equation. One common form equates the ratio of the returned signal power, S, to the receiver noise power, N, as shown in the following: [8]

$$\frac{S}{N} = \frac{PG^2 \lambda^2 \sigma}{(4\pi)^3 R^4 N_F kTBL} \tag{3-1}$$

where:

P = Transmitted Power

G = Antenna Gain

λ = Wavelength

σ = Target Cross Section

R = Range of Target

N_F = Noise Figure of Receiver

k = Boltzman's Constant

T = Temperature

B = Bandwidth

L = Assorted Losses

3.1. Derivation of the Radar Range Equation

The radar range equation is derived by starting with the target power density at range R [8].

$$P_d = \frac{P}{4\pi R^2} \quad (\mathrm{W/m^2}) \tag{3-2}$$

If the signal is directed by an antenna with gain, G, then

$$P_d = \frac{PG}{4\pi R^2} \quad (\mathrm{W/m^2}) \tag{3-3}$$

The reflected power density from the target will be a function of the target cross section, σ. The reflected power from the target is:

$$P_r = \frac{PG\sigma}{4\pi R^2} \quad (\mathrm{W}) \tag{3-4}$$

The power density of the reflected signal back at the radar is:

$$P_d = \frac{PG\sigma}{(4\pi R^2)^2} \quad (\mathrm{W/m^2}) \tag{3-5}$$

The power of the reflected signal, S, at the radar is the product of the power density times that area of the antenna. (A = aperture)

$$S = \frac{PG\sigma A}{(4\pi R^2)^2} \quad \text{(W)} \tag{3-6}$$

Substituting the relationship of gain and aperture,

$$A = \frac{G\lambda^2}{4\pi} \quad (\text{m}^2) \tag{3-7}$$

then,

$$S = \frac{PG^2\lambda^2\sigma}{(4\pi)^3 R^4} \quad \text{(W)} \tag{3-8}$$

When equating as S/N, the noise figure of the receiver, N_F, and the thermal noise power, kTB, become part of the denominator.

$$\frac{S}{N} = \frac{PG^2\lambda^2\sigma}{(4\pi)^3 R^4 N_F kTB} \tag{3-9}$$

The term L, representing a number of losses in the radar system and propagation losses, is introduced accordingly, making the equation:

$$\frac{S}{N} = \frac{PG^2\lambda^2\sigma}{(4\pi)^3 R^4 N_F kTBL} \tag{3-10}$$

If R_o is designated as the range that provides unity S/N, then the equation becomes:

$$R_o^4 = \frac{PG^2 \lambda^2 \sigma}{(4\pi)^3 N_F kTBL} \tag{3-11}$$

When this range is known, then the S/N at other ranges can be calculated by using the relationship:

$$\frac{S}{N} = \left(\frac{R_o}{R}\right)^4 \tag{3-12}$$

Significantly, the S/N is inversely proportional to the fourth power of range. Knowing the above relationships, one can calculate the parameters for specified ranges that provide useful return signals.

3.2. Description of the Simulated Radar System

A hypothetical radar system consists of a ground based radar for illumination and a missile with front and rear reference antennas receiving the front and rear signals, respectively. Also the effects of clutter must be modeled for the accuracy of the system. The following parameters are used to define the system [11].

Illumination Antenna

Peak Power Gain	0 dB
First Sidelobe	-23 dB
Power	10 MW
3dB Beamwidth	0.65 degrees
Backlobe Gain	-0.6 dB
Illuminator Height	25 meters
Effective Antenna Aperture Diameter	1.0615 meters
Wavelength	0.01 meters

Front and Rear Reference Receiver Antenna

Peak Power Gain	0 dB
First Sidelobe	-13 dB
3dB Beamwidth	4 degrees
Backlobe Gain	-2.6 dB
Antenna Diameter	14.625 cm
Feed Horn Aperture Width	2.25 cm
Wavelength	1.0 cm

The illuminator antenna pattern used is a cosine aperture. The power gain of a cosine aperture antenna is as follows [7]:

$$G_I(\theta) = \left(\frac{4}{\pi}\left(\frac{\sin\left(\frac{\pi a}{\lambda}\sin(\theta) + \frac{\pi}{2}\right)}{\frac{\pi a}{\lambda}\sin(\theta) + \frac{\pi}{2}} + \frac{\sin\left(\frac{\pi a}{\lambda}\sin(\theta) - \frac{\pi}{2}\right)}{\frac{\pi a}{\lambda}\sin(\theta) - \frac{\pi}{2}}\right)\right)^2 \qquad (3\text{-}13)$$

where:

a = Effective Aperature Diameter

θ = Look Angle

The antenna pattern for the illuminator is shown in Figure 3-1.

The front and rear reference receiver antennas are modeled as a combination of a circular aperture with a rectangular blockage (feed horn). The antenna pattern is found by subtracting the blockage pattern from the pattern for a circular dish. The gain for a circular aperture (normalized) is given by [9]:

$$G_{circ}(\phi) = \left(2\frac{J_1\left(\frac{2\pi r_o}{\lambda}\sin\phi\right)}{\left(\frac{2\pi r_o}{\lambda}\sin\phi\right)}\right)^2 \qquad (3\text{-}14)$$

where:

J_1 = First Order Bessel Function

r_o = Antenna Radius

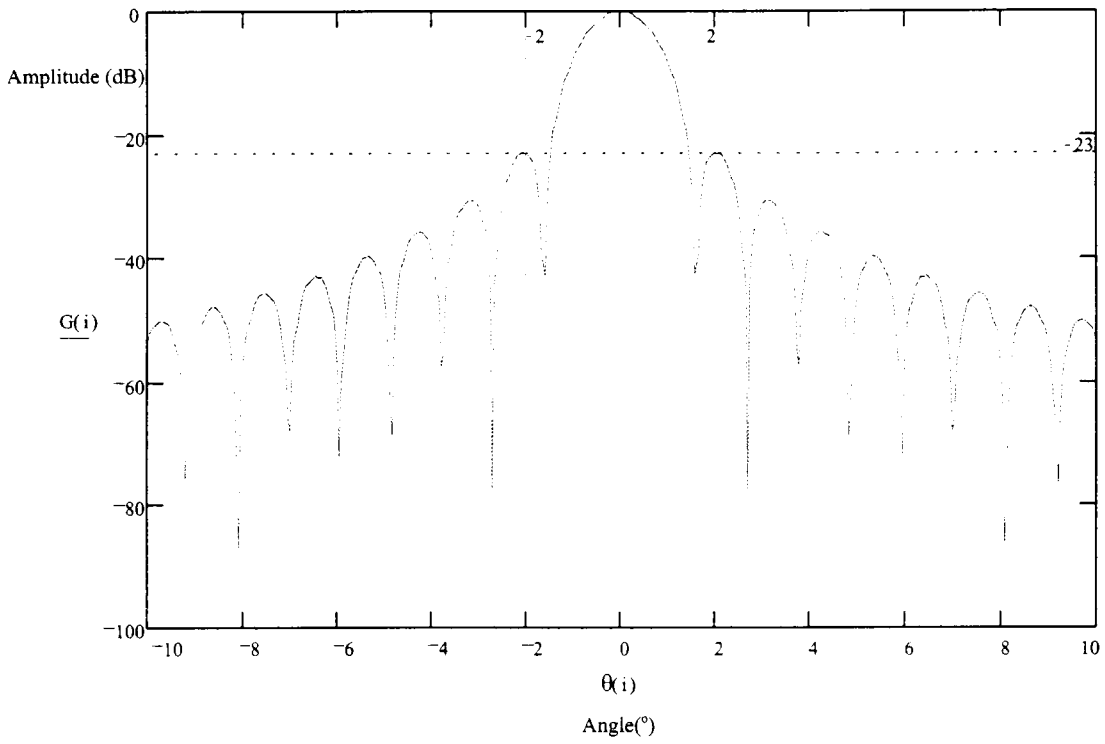


Figure 3-1: Illuminator Antenna Pattern

λ = Wavelength

ϕ = Look Angle

Power gain for a rectangular aperture (normalized) is obtained from [7]:

$$G_{rect}(\phi) = \left(\frac{\sin\left(\left(\frac{\pi a}{\lambda}\right)\sin\phi\right)}{\left(\frac{\pi a}{\lambda}\right)\sin\phi}\right)^2 \tag{3-15}$$

where:

a = Aperture Width

The resultant normalized receiver antenna power gain can be written as follows [7]:

$$G_r(\phi) = \left(\frac{1}{1-K}\left(2\frac{J_1\left(\frac{2\pi r_o}{\lambda}\sin\phi\right)}{\frac{2\pi r_o}{\lambda}\sin\phi} - K\frac{\sin\left(\frac{\pi a}{\lambda}\sin\phi\right)}{\frac{\pi a}{\lambda}\sin\phi}\right)\right)^2 \tag{3-16}$$

where K relates the relative amplitude of the circular dish and the aperture blockage. K was chosen to set the blockage pattern 21 dB below the circular dish pattern [7]. The resulting pattern is shown in Figure 3-2.

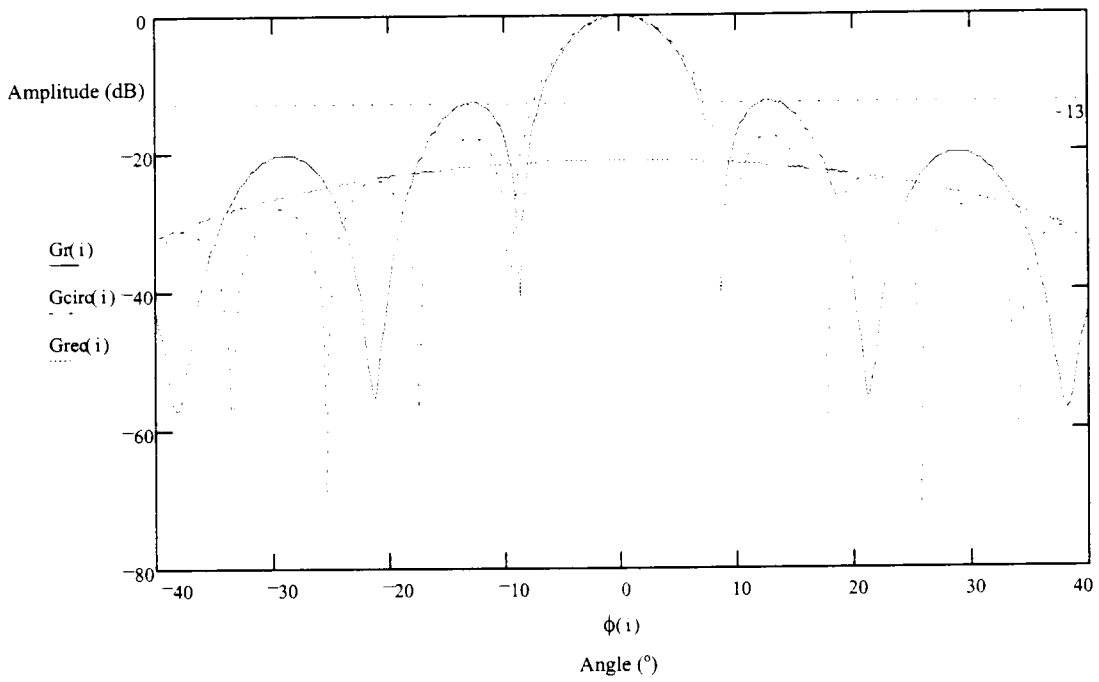


Figure 3-2: Front and Rear Receiver Antenna Pattern

The clutter model describes the relationships between the illuminator, missile, target, and the earth's surface. The clutter model is broken down into three parts, main lobe clutter, sidelobe (feedthrough) clutter, and glistening [11]. Figure 3-3 illustrates the geometric relationship of clutter to the missile receivers.

Main lobe clutter is caused by the increased gain of the missile front antenna main lobe. The location in doppler frequency is determined by the angle the main lobe makes with the missile velocity vector. When the front antenna main lobe is aligned with the missile velocity vector, the main lobe clutter will occur at the doppler frequency of the missile velocity vector. If the front antenna main lobe makes a 90° angle with the missile velocity vector, for example, then the main lobe clutter would occur at zero doppler frequency. This phenomenon also occurs with the rear main lobe [11].

The sidelobe clutter is due to diffuse return from the surface. This return is not highly directional and comes from every point on the surface that receives illumination. The shape of the sidelobe clutter is determined predominantly by the receiver antenna pattern. The relative power level of the clutter starting at the antenna main lobe clutter frequency is approximately a replica of the antenna pattern [11].

The rise in energy below the sidelobe clutter is due to a different mechanism of reflection called glistening. This reflection comes from a small area on the surface called the glistening region. The glistening region is centered about the mirror-like reflection

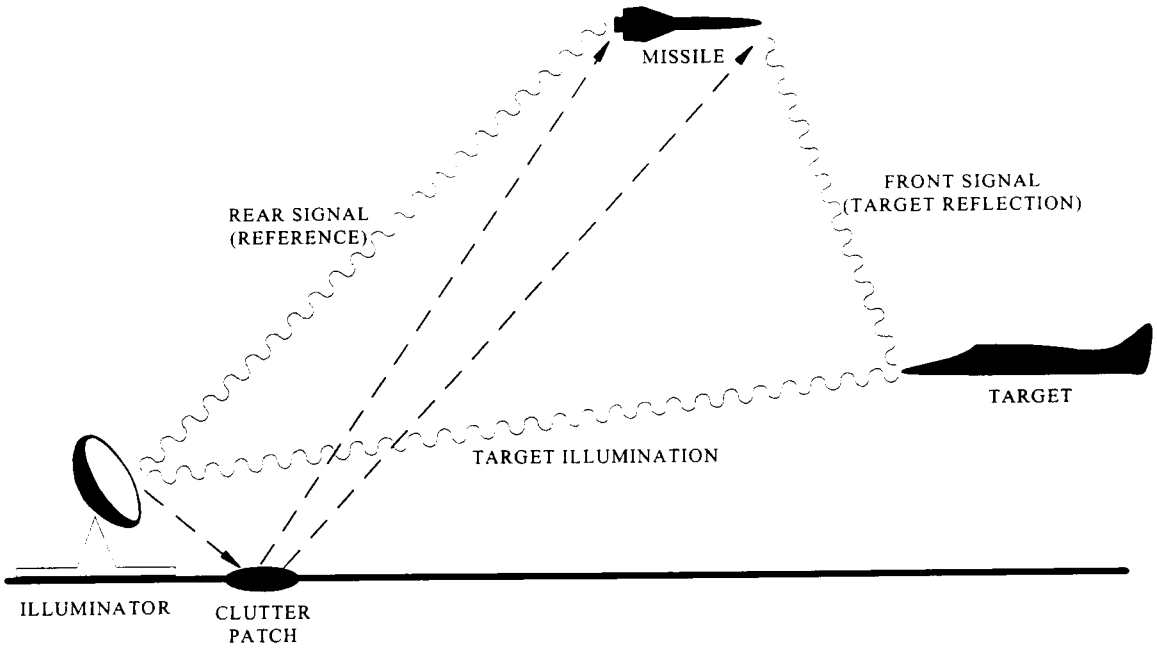


Figure 3-3: Clutter Patch Relation to System Geometry

point as shown in Figure 3-4. The glistening reflection is highly directional. This directivity causes the glistening reflection power to be a strong function of position within the glistening region. The closer to the illuminator in position, the stronger the power of the region [11].

3.3. Simulation Methodology

To set up a method for performance evaluation of the two receiver models developed, a *pure* input test and a *noisy* input test were performed. The *pure* input can be correlated to a high altitude target condition for a missile system. The *noisy* input can be correlated to a low altitude target condition.

The high altitude target is said to be a *pure* system due to the low noise level. The illuminator antenna is pointing at the high altitude target, thus creating a large angle between the antenna main beam and the surface. The high altitude condition is illustrated in Figure 3-5. The radiated power that reflects from the surface can be attributed to a low power sidelobe of the illuminator antenna. In addition, the reflected energy is propagated a much greater distance, thus reducing the received energy by the missile. In some cases the thermal noise of the receiver is higher than the clutter return.

The front and rear input spectra for the high altitude scenario are generated using a modified version of Raytheon Electronic System's (RES's) Point-in-Flight (PIF) simulation program. The PIF simulation program models each element, the illuminator,

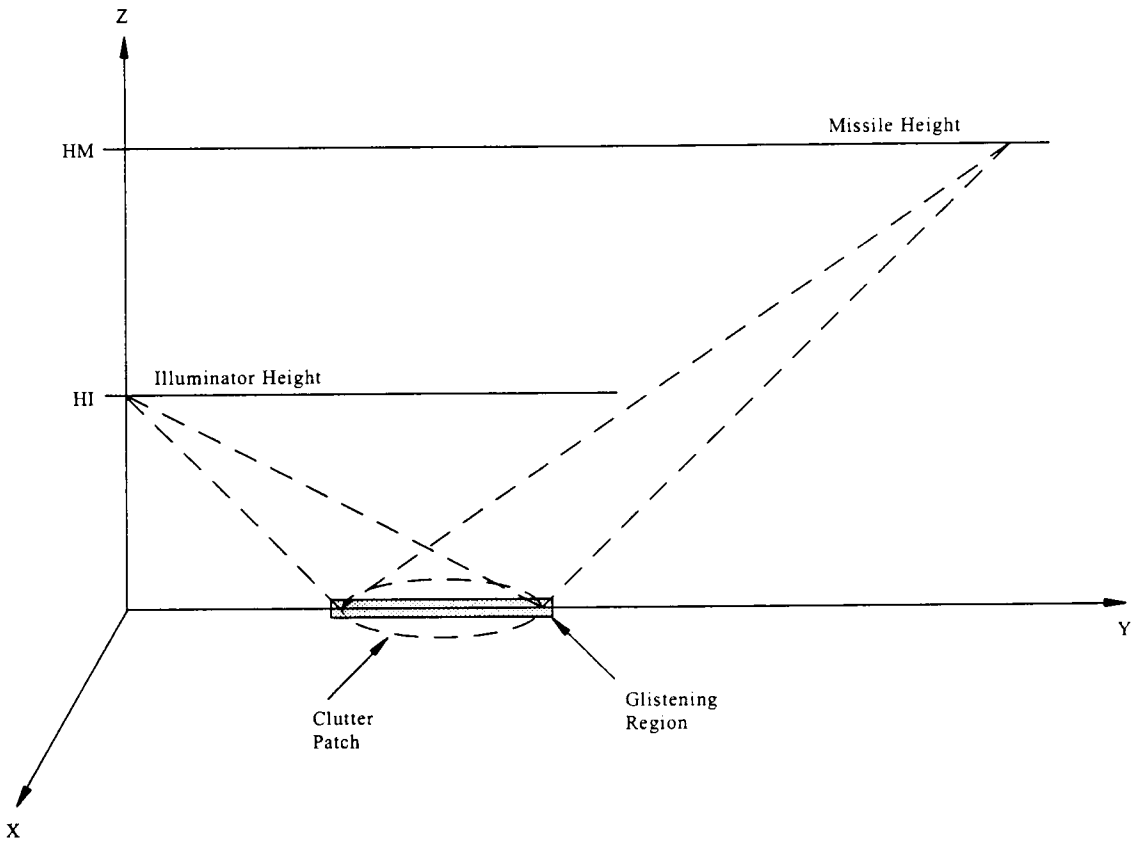


Figure 3-4: Glistening Region Relation to Clutter Patch

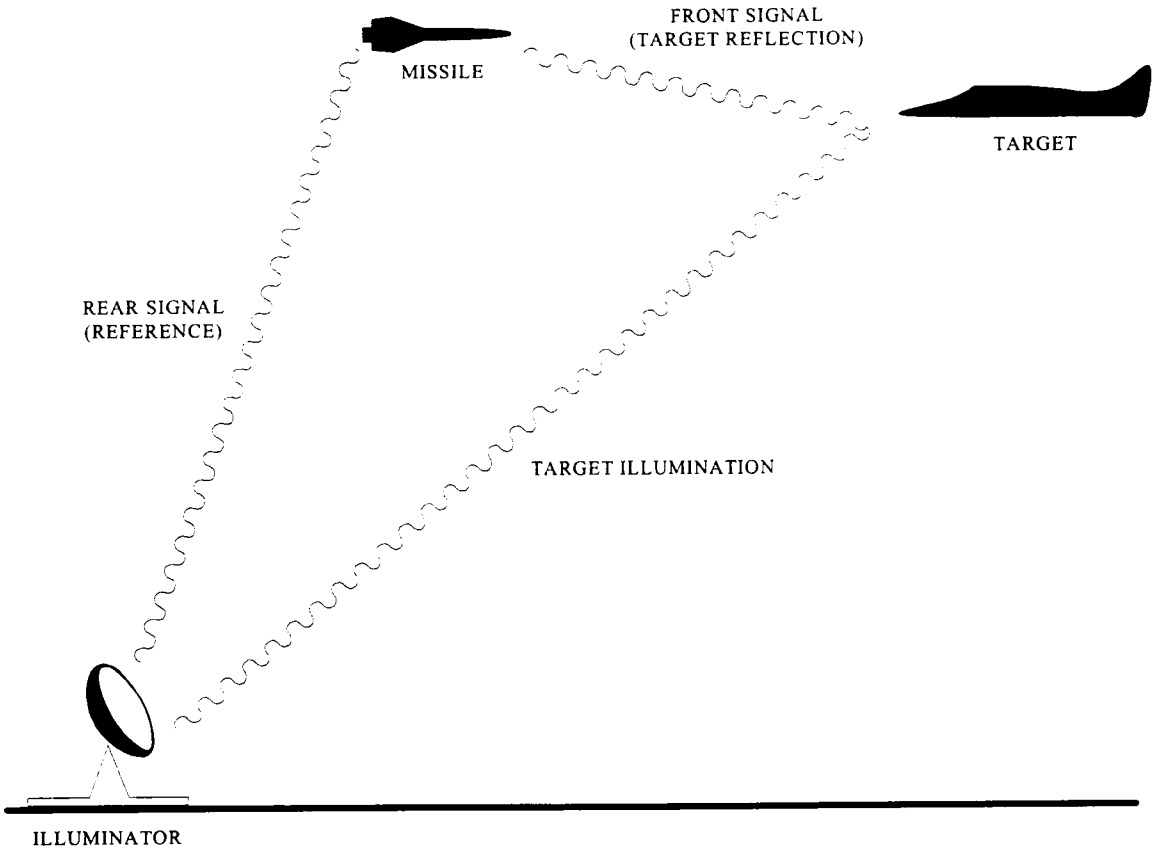


Figure 3-5: System Geometry for the High Altitude Condition

the missile, and the target, of a semiactive radar guided missile system. The program uses the radar range equation to calculate the missile front and rear input spectra. The power at each frequency point is calculated and summed for all contributing sources to provide an overall power value. The contributing sources include the direct illumination signal, the target reflection signal, and all energies re-radiated from the surface (clutter).

The program requires the user to supply geometric information, as well as velocity information. In addition, the program requires the illumination antenna and the missile front and rear antenna specifications. The program was modified to model the radar system specified in Section 3.2. Table 3-1 shows the system geometry for the high altitude condition. Figure 3-6 shows the PIF program results for the front antenna spectrum. Note that the frequency axis for all plots represent absolute baseband frequency. If the frequency is doppler the title will be reflected as such. Figure 3-7 shows the PIF program results for the rear antenna spectrum.

The low altitude target is said to be a *noisy* system due to the high noise level. Unlike the high altitude scenario, the illuminator is pointing at a low altitude target, creating a small angle between the antenna main beam and the surface. The low altitude condition is illustrated in Figure 3-8. The radiated power that reflects from the surface originates from a higher power sidelobe, if not part of the main lobe, of the illuminator antenna. Also, the reflected energy is propagated a relatively short distance. In some cases the input noise power is greater in magnitude than the input target power.

Table 3-1: Radar System Input Parameters for High Altitude Condition

Position (m)			Velocity (m/s)		
Variable Name	**Variable Description**	**Variable Value**	**Variable Name**	**Variable Description**	**Variable Value**
CM	Missile Cross Range Position	-22322.75	VMC	Missile Cross Range Velocity	-326.29
RM	Missile Down Range Position	106070.20	VMR	Missile Down Range Velocity	1382.76
HM	Missile Height Position	22616.92	VMH	Missile Height Velocity	-686.63
CT	Target Cross Range Position	-22834.76	VTC	Target Cross Range Velocity	-54.09
RT	Target Down Range Position	110277.90	VRT	Target Down Range Velocity	-814.58
HT	Target Height Position	21371.24	VTH	Target Height Velocity	13.85
CI	Illuminator Cross Range Position	0.00	VIC	Illuminator Cross Range Velocity	0.00
RI	Illuminator Down Range Position	0.00	VIR	Illuminator Down Range Velocity	0.00
HI	Illuminator Height Position	25.00	VIH	Illuminator Height Velocity	0.00

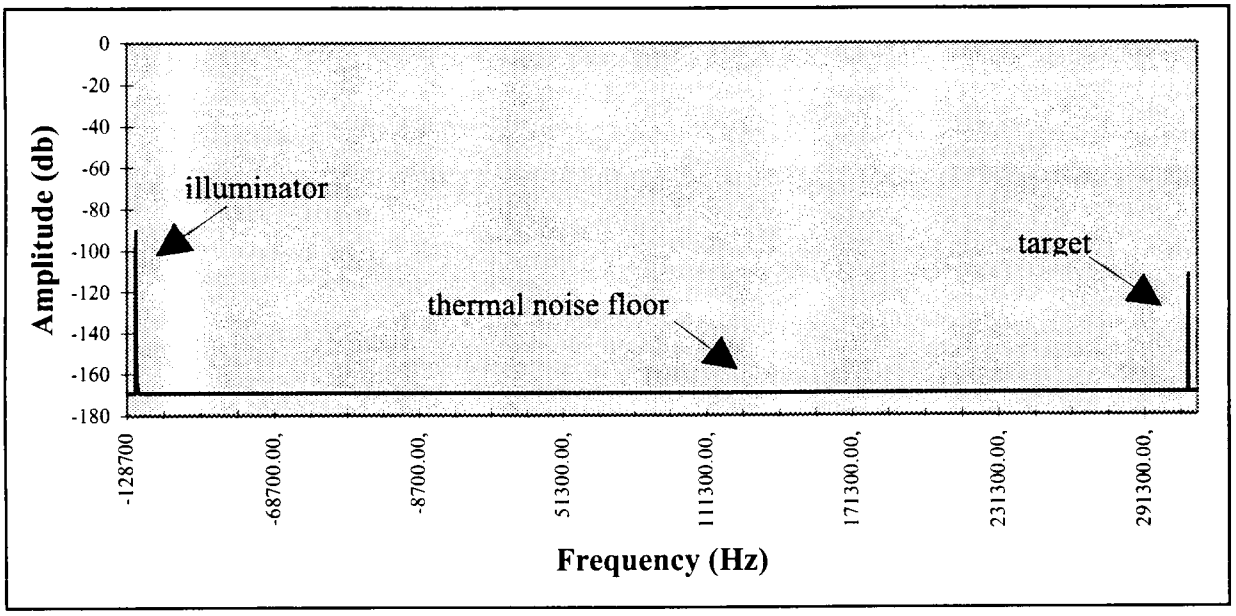


Figure 3-6: Front Antenna Spectrum for High Altitude Condition

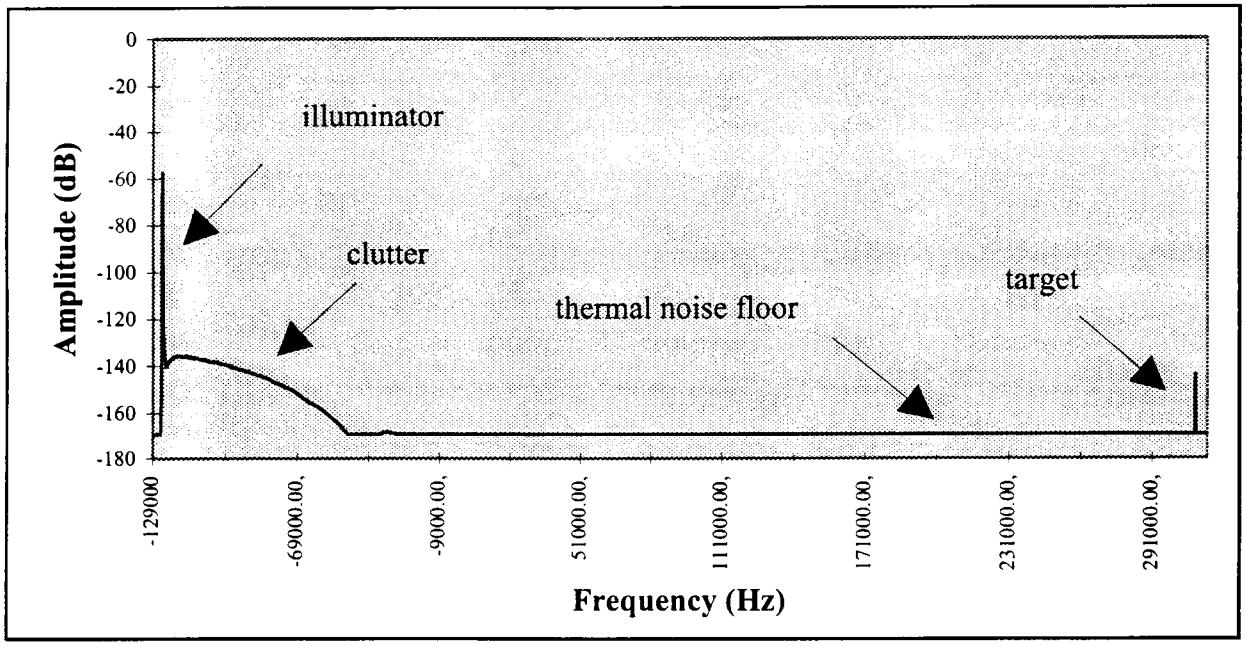


Figure 3-7: Rear Reference Spectrum for High Altitude Condition

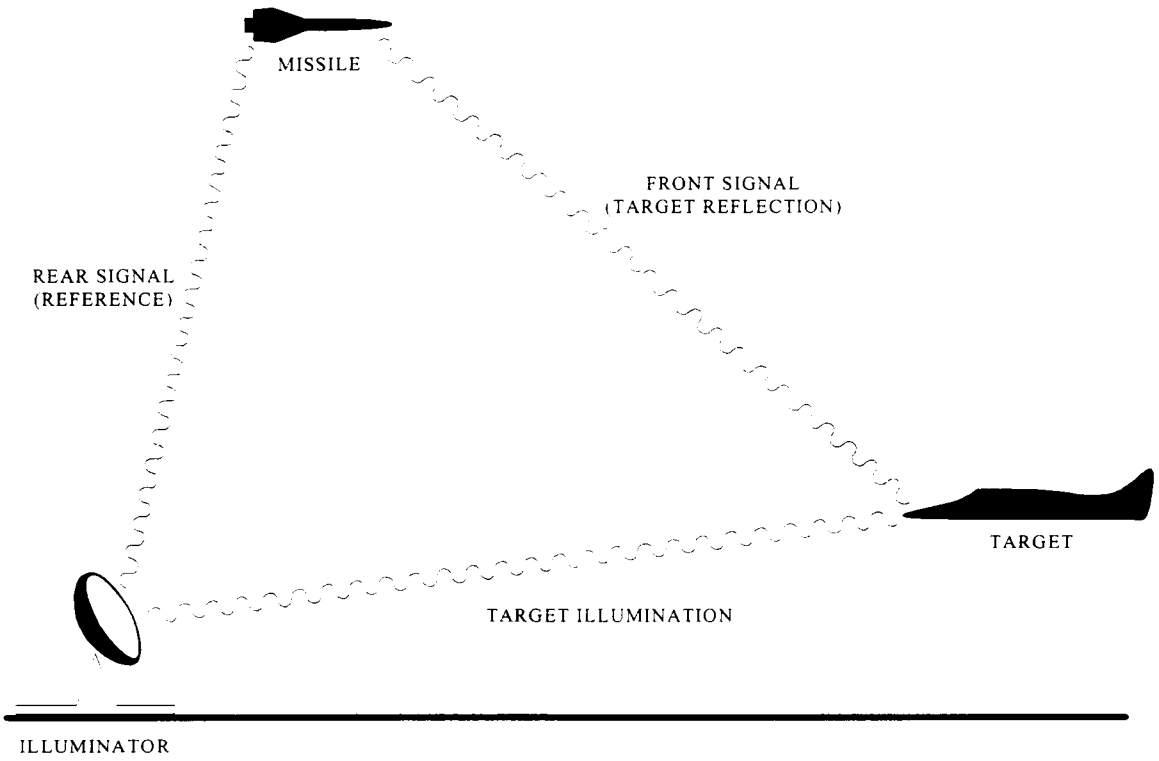


Figure 3-8: System Geometry for the Low Altitude Condition

The front and rear input spectra for the low altitude scenario were generated using the same modified version of RES's PIF simulation program. The system geometry for the low altitude condition are shown in Table 3-2. Figure 3-9 shows the simulation results for the front antenna spectrum. Figure 3-10 shows the simulation results for the rear antenna spectrum.

Table 3-2: Radar System Input Parameters for Low Altitude Condition

Position (m)			Velocity (m/s)		
Variable Name	**Variable Description**	**Variable Value**	**Variable Name**	**Variable Description**	**Variable Value**
CM	Missile Cross Range Position	-681.46	VMC	Missile Cross Range Velocity	-111.87
RM	Missile Down Range Position	6648.39	VMR	Missile Down Range Velocity	1085.34
HM	Missile Height Position	559.90	VMH	Missile Height Velocity	-249.86
CT	Target Cross Range Position	-914.40	VTC	Target Cross Range Velocity	0.00
RT	Target Down Range Position	9405.28	VRT	Target Down Range Velocity	-231.65
HT	Target Height Position	6.10	VTH	Target Height Velocity	0.34
CI	Illuminator Cross Range Position	0.00	VIC	Illuminator Cross Range Velocity	0.00
RI	Illuminator Down Range Position	0.00	VIR	Illuminator Down Range Velocity	0.00
HI	Illuminator Height Position	25.00	VIH	Illuminator Height Velocity	0.00

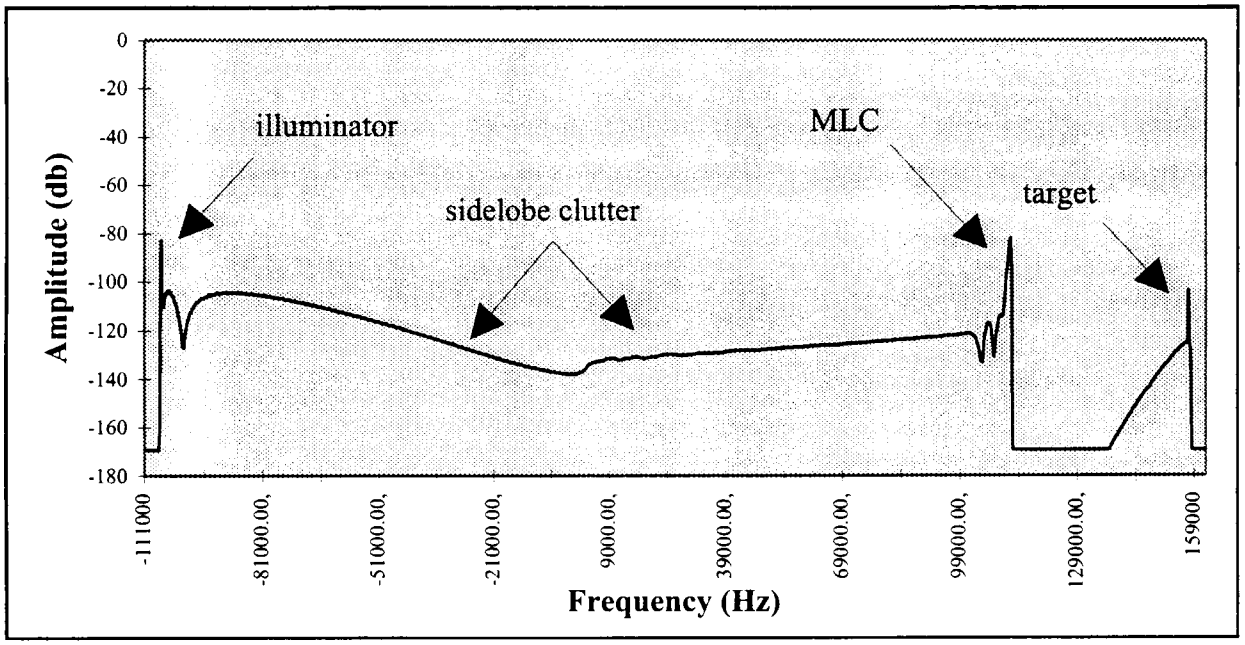


Figure 3-9: Front Antenna Spectrum for Low Altitude Condition

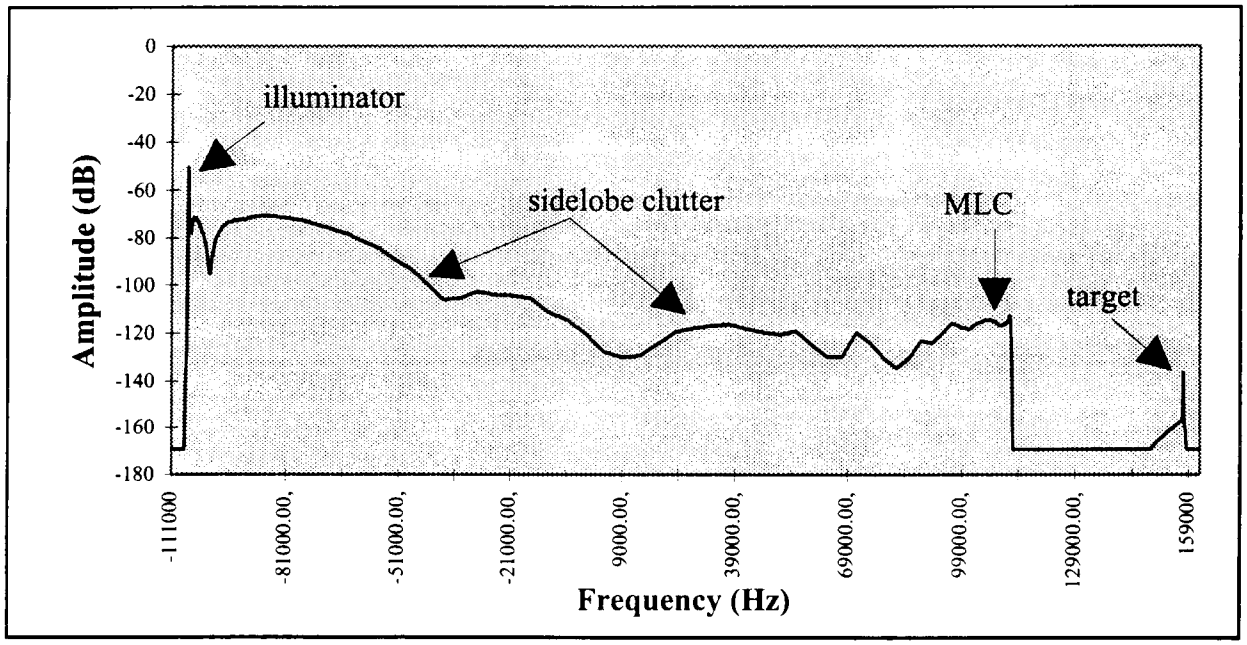


Figure 3-10: Rear Reference Spectrum for Low Altitude Condition

Each receiver model will be simulated using the resulting front and rear spectra produced by RES's PIF simulation under both test conditions stated above. The high altitude target condition simulation results will be used to establish a baseline for each receiver model. It is important to know each receiver model's performance without noise. Both models are expected to perform acceptably under this condition. The low altitude target condition simulation results will be used to determine the actual performance of each receiver model.

4. SIMULATION OF A GENERAL MISSILE RECEIVER

The function of the radar receiver is to detect desired echo signals in the presence of noise, interference, or clutter. It must separate wanted from unwanted signals, and amplify the desired signals to a level where target information can be processed by the remainder of the missile guidance system. The design of the radar receiver depends not only on the type of waveform to be detected, but on the nature of the noise, interference, and clutter echoes with which the desired echo signal must compete [7].

4.1. Receiver Model Development

The receiver must extract the target information from the front antenna waveform or spectrum. To extract the target data, it is necessary to shift the data to baseband. This can be accomplished by mixing the front spectrum with the rear spectrum, as seen in Figure 4-1. Typically, the radar frequency is in the microwave region, thus requiring conversion of the input spectra to an intermediate frequency (IF). This simplifies amplifier and filter design. A local oscillator (LO) is used to downconvert the input spectra to a more usable center frequency. The LO spectrum is shown in Figure 4-2 [13]. The source code for the general receiver model is shown in Appendix A.

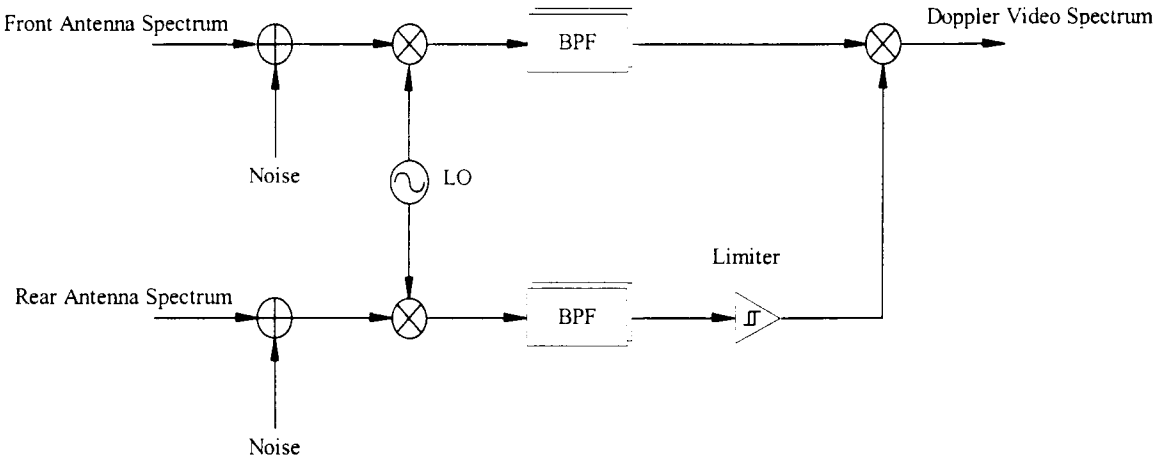


Figure 4-1: General Receiver Model Block Diagram

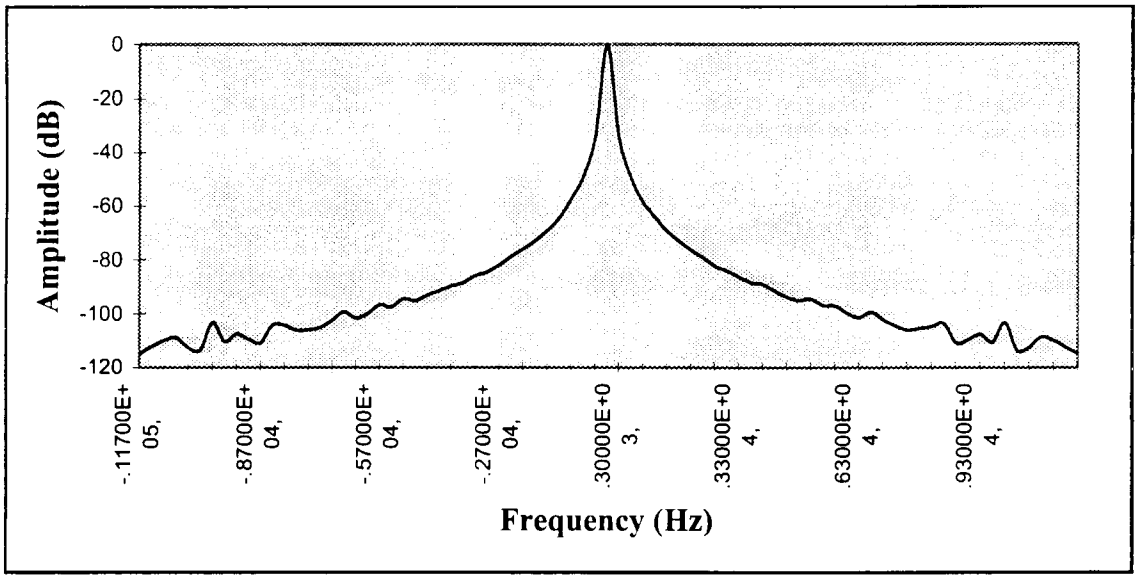


Figure 4-2: Local Oscillator Spectrum

4.2. Simulation Results

The results of the simulation were divided into two categories, the high altitude low clutter environment and the low altitude high clutter environment. The receiver was simulated using both sets of input spectra.

Following the block diagram in Figure 4-1, the receiver thermal noise is added to both the front and rear antenna spectra. In addition, the LO is used to downconvert the front and rear spectra to an IF frequency that will meet the requirements of the internal receiver circuitry.

The front antenna spectrum shown in Figure 3-6 is convolved with the LO spectrum shown in Figure 4-2 to produce the resultant spectrum in Figure 4-3. Notice that since the LO has phase noise sidebands, the downconverted front antenna spectrum is slightly corrupted. However, the levels are quite low since the LO is *clean* with respect to high power spurious noise.

Similarly, the rear reference spectrum shown in Figure 3-7 is convolved with the same LO spectrum as the front antenna spectrum to produce the resultant spectrum in Figure 4-4. This spectrum shows very little change when convolved with the LO spectrum.

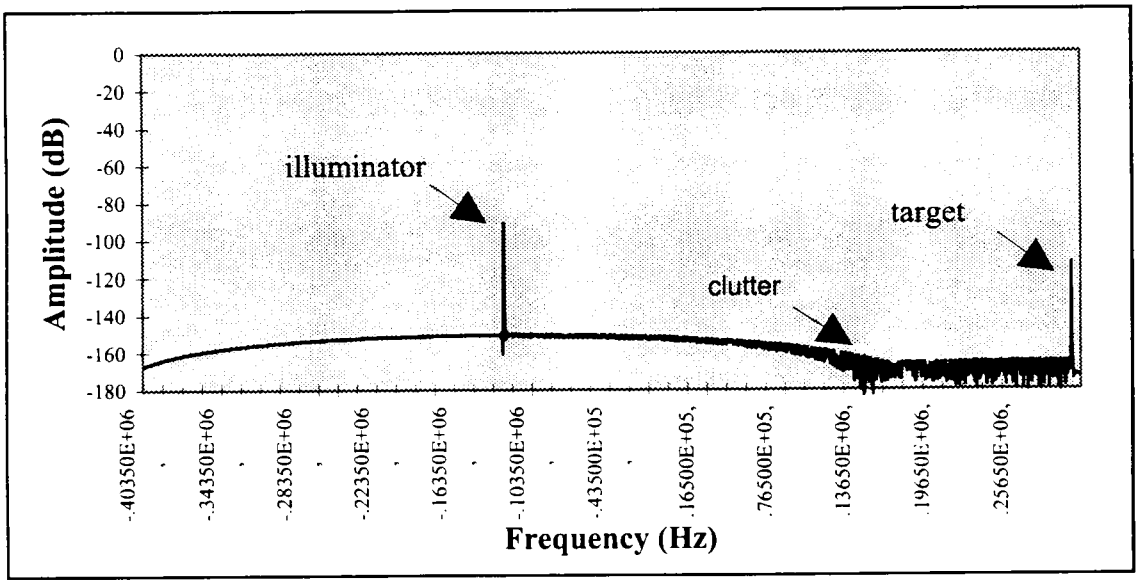


Figure 4-3: Downconverted Front Antenna Spectrum for High Altitude Condition

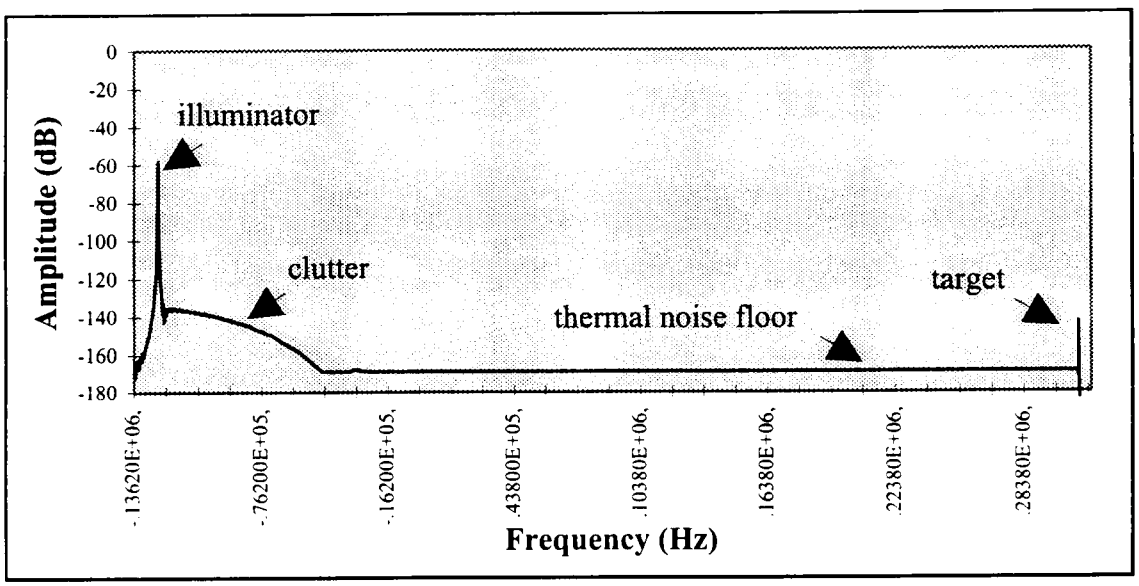


Figure 4-4: Downconverted Rear Antenna Spectrum for High Altitude Condition

In theory, the downconverted front and rear spectra can be mixed directly. However, in missile systems this is not true. The physical mixer must have a constant rear input power to mix with the front spectrum. In a missile system, the rear input spectrum fades as the missile moves away from the illuminator. In this system, a high gain amplifier and a limiter must be used to provide this constant power. The limited rear spectrum is shown in Figure 4-5. The spectrum of the rear reference is reflected about the illuminator. It is this function that causes the receiver difficulty in detecting the target, as will be demonstrated in the low altitude condition.

The resultant baseband doppler video spectrum is shown in Figure 4-6. The target to clutter noise ratio is shown to be approximately 50 dB. Notice how the clutter of the limited rear spectrum is spread in the direction of the target. In this particular case, the clutter levels were low enough that they did not significantly affect the target signal to clutter ratio.

The receiver was then tested using the low altitude high clutter condition, the test case. The front antenna spectrum is shown in Figure 3-9 and the rear reference spectrum is shown in Figure 3-10. Again these spectra were convolved with the LO, shown in Figure 4-2, to produce the downconverted spectra shown in Figure 4-7 and Figure 4-8, respectively.

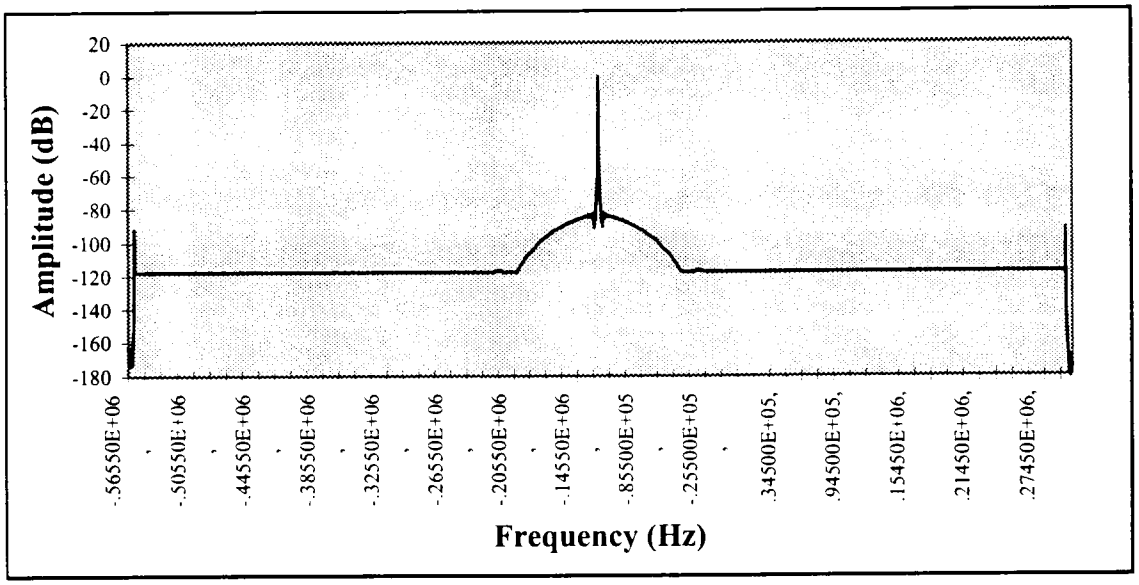


Figure 4-5: Limited Rear Spectrum for High Altitude Condition

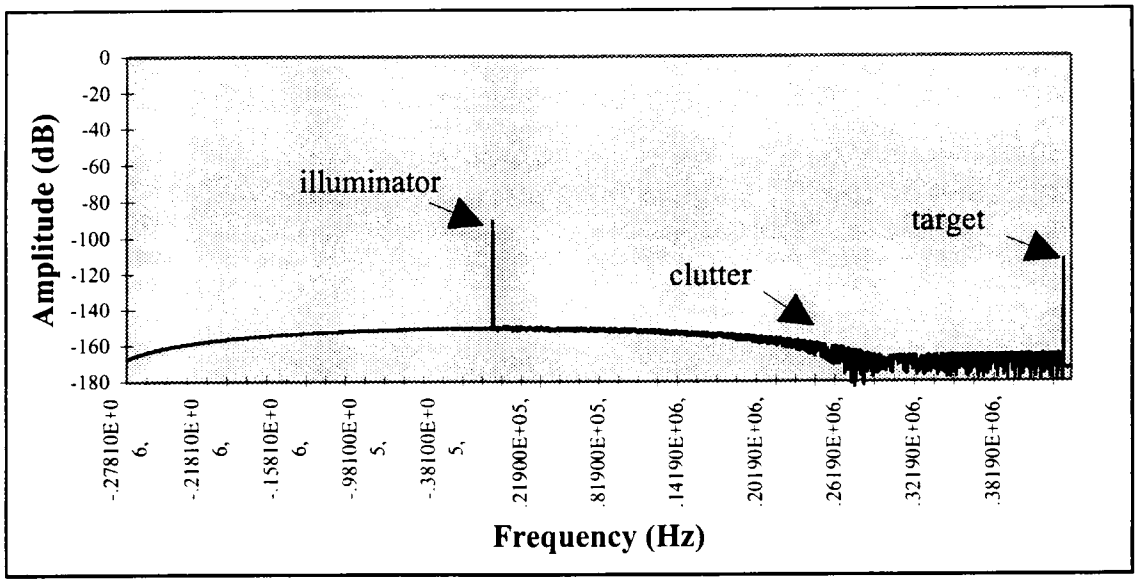


Figure 4-6: Baseband Doppler Video Spectrum for High Altitude Condition

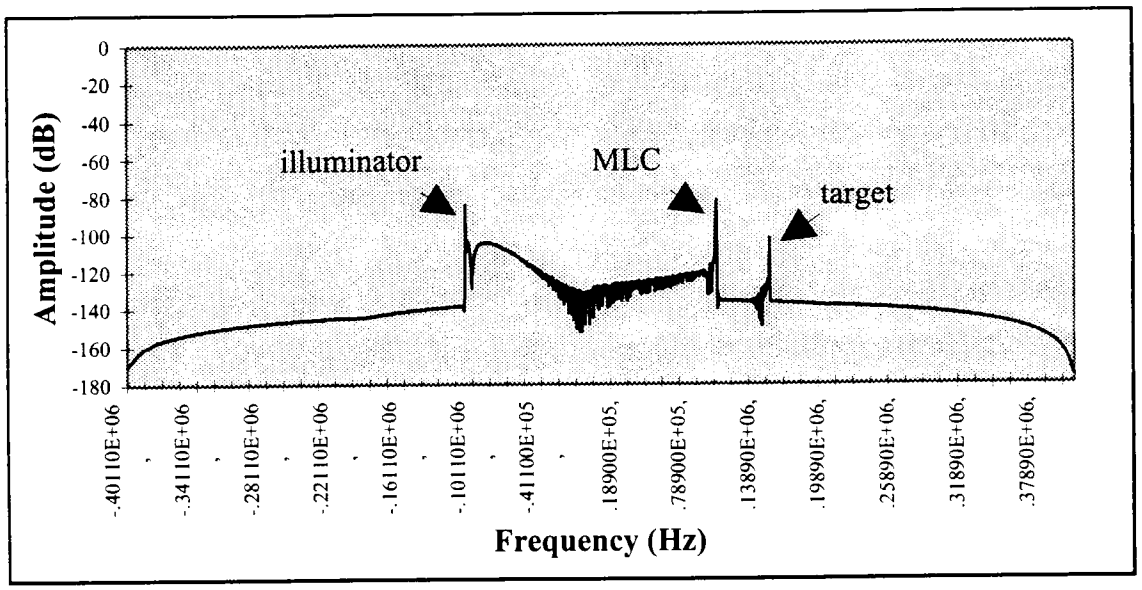


Figure 4-7: Downconverted Front Antenna Spectrum for Low Altitude Condition

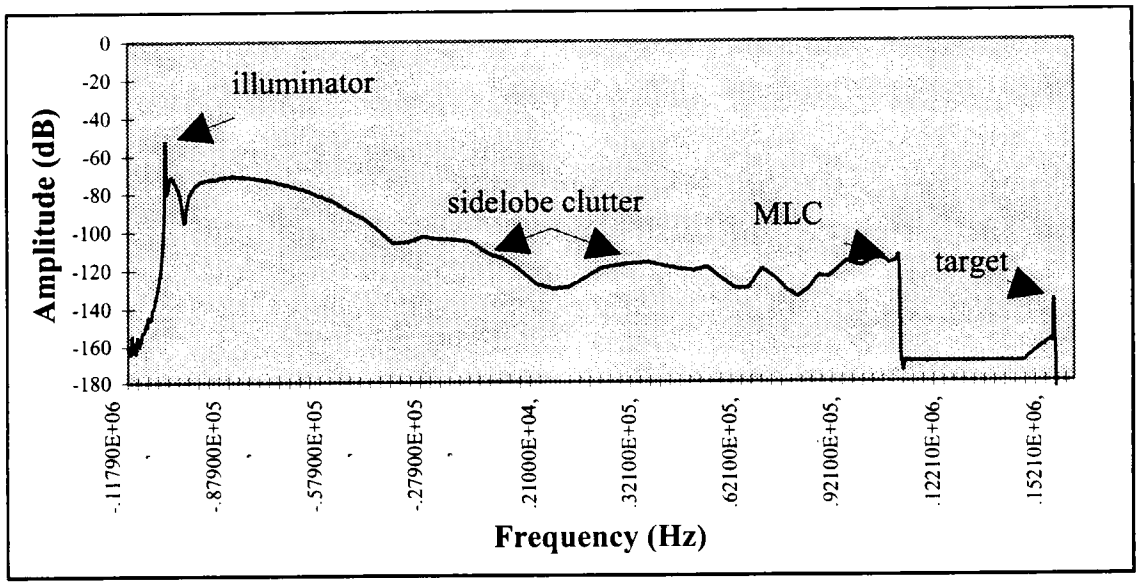


Figure 4-8: Downconverted Rear Reference Spectrum for Low Altitude Condition

Notice that the clutter levels are much higher with the low altitude condition. The reflected illuminator signals and reflected target signals have a much shorter distance to travel, thus, the signal propagation loss is greatly reduced. As stated earlier, ideally these two spectra should be convolved directly. Figure 4-9 shows the pure baseband doppler video spectrum without limiting the rear reference input. The target to clutter ratio is shown to be approximately 30dB. This plot shows what the spectrum should look like.

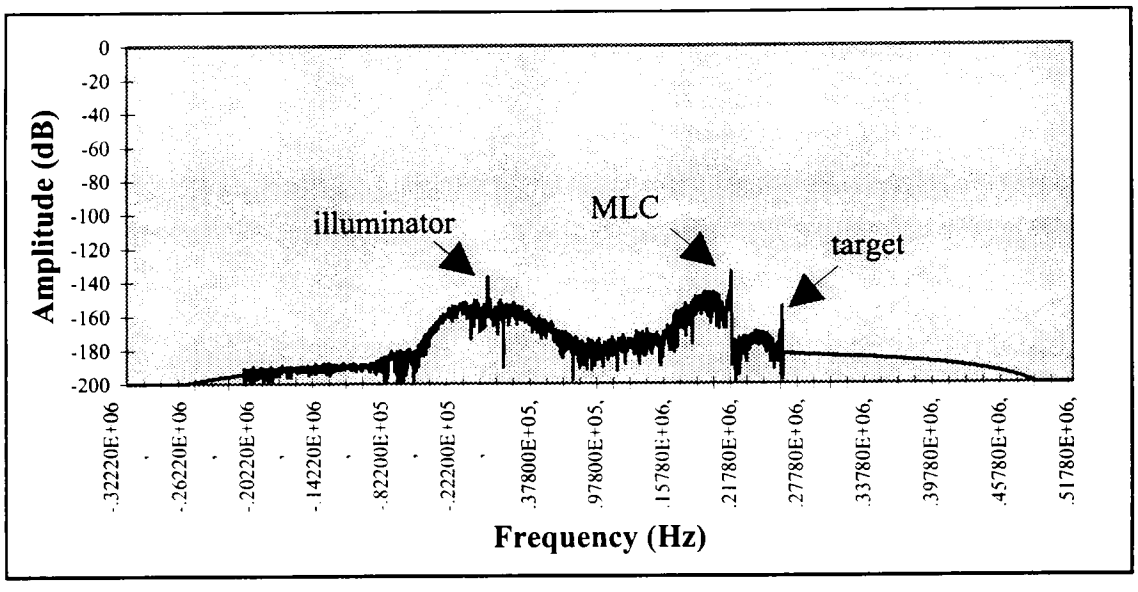


Figure 4-9: Pure Baseband Doppler Video Spectrum for Low Altitude Condition

However, the rear reference must be limited. Figure 4-10 displays the limited rear reference spectrum. The resultant baseband doppler video spectrum is shown in Figure 4-11. Notice that the target has only a 5dB signal to clutter ratio. In most instances, this spike could be considered to be random noise. Additional signal processing would be required to determine if the signal is actually a target or just random noise. In a missile system, time cannot be wasted. It only takes a fraction of a second for the missile to travel out of range of the target, resulting in a missed target.

This receiver model performs well under high altitude conditions with low clutter levels. However, performance is questionable under the low altitude condition with high clutter levels. If the missile system mission statement required low altitude operation, this receiver would not meet the system requirements.

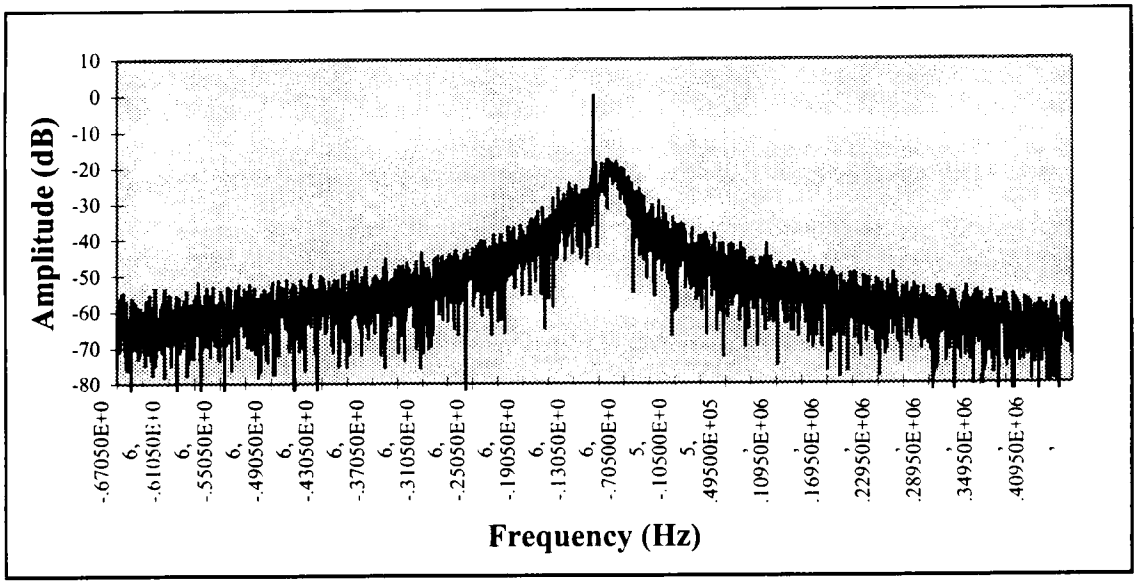


Figure 4-10: Limited Rear Spectrum for Low Altitude Condition

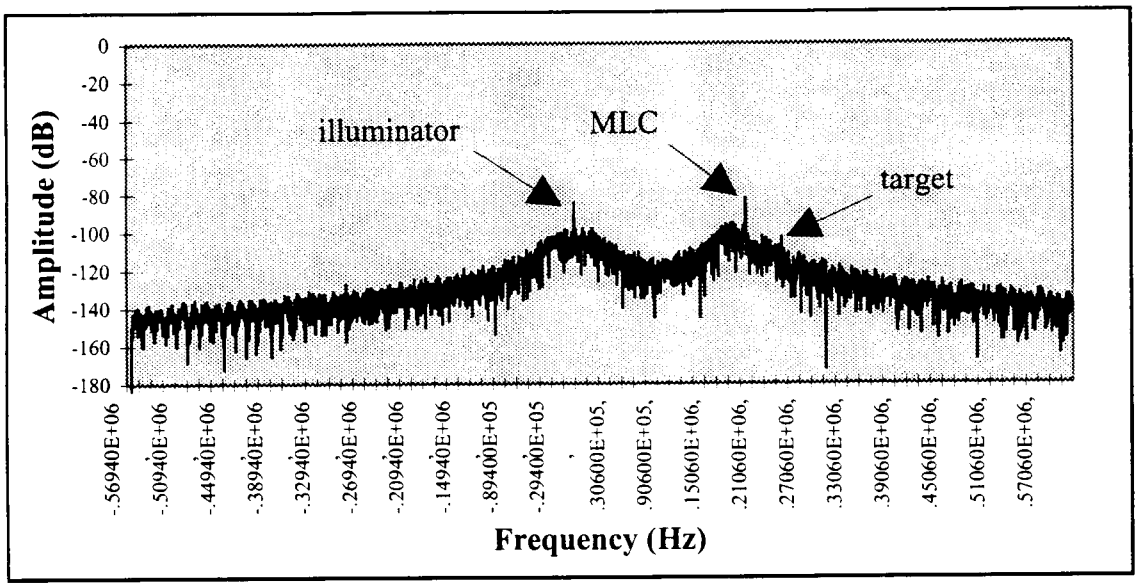


Figure 4-11: Baseband Doppler Video Spectrum for Low Altitude Condition

5. SIMULATION OF AN ALTERNATIVE MISSILE RECEIVER

5.1. Receiver Model Development

An alternative missile receiver was developed to correct the spectral noise problems due to limiting of the rear reference signal. The receiver is identical to the previous receiver in the front antenna signal path, but is quite different in the rear signal path, as can be seen in Figure 5-1.

The rear input signal is first downconverted using the fixed LO block similar to the previous model. However, the rear tuning LO, controlled by the illuminator detection block, is used to position the illuminator frequency in the center of the subsequent bandpass filter. The illuminator detector block not only controls the rear tuning LO, but also determines the absolute illuminator frequency to control the front tuning LO. The front tuning LO is local to the receiver, thus the input signal strength is fixed. This eliminates the necessity for a limiter in the rear signal path. The front tuning LO spectrum for the high altitude condition is shown in Figure 5-2. The front tuning LO spectrum for the low altitude condition is shown in Figure 5-3. The front LO frequency is different for each test condition [16].

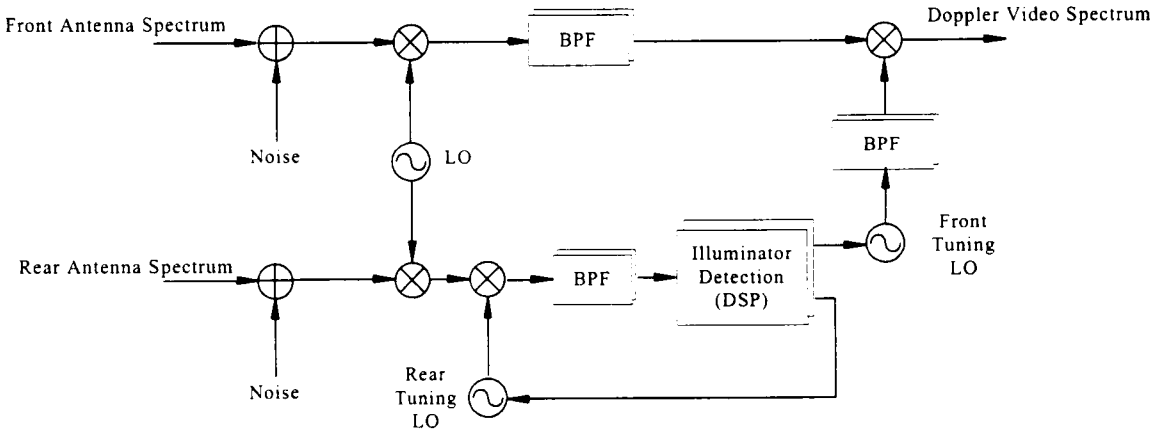


Figure 5-1: Alternative Receiver Model Block Diagram

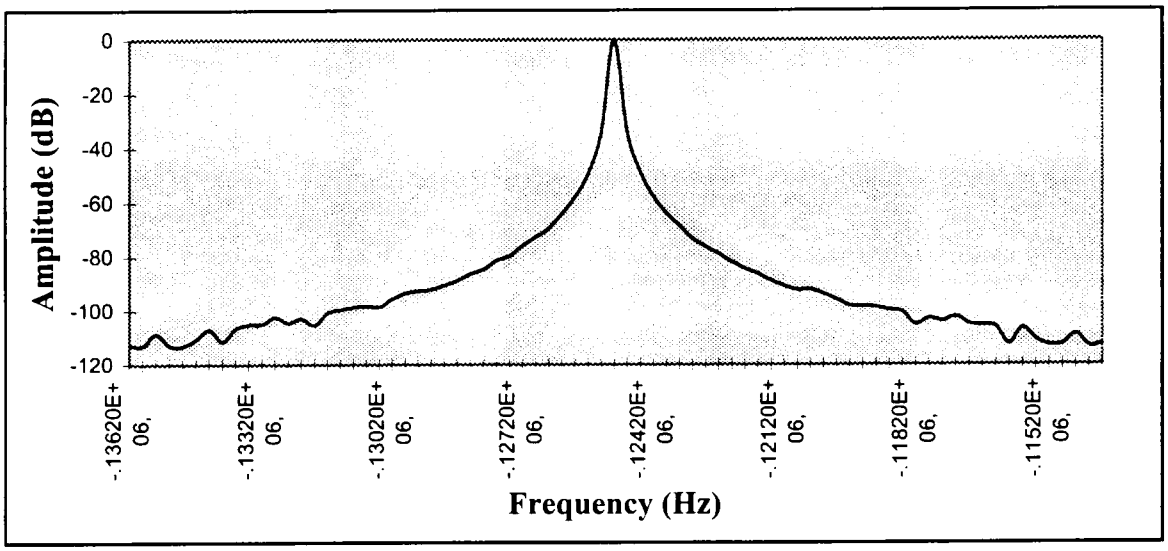


Figure 5-2: Front Tuned LO Spectrum for High Altitude Condition

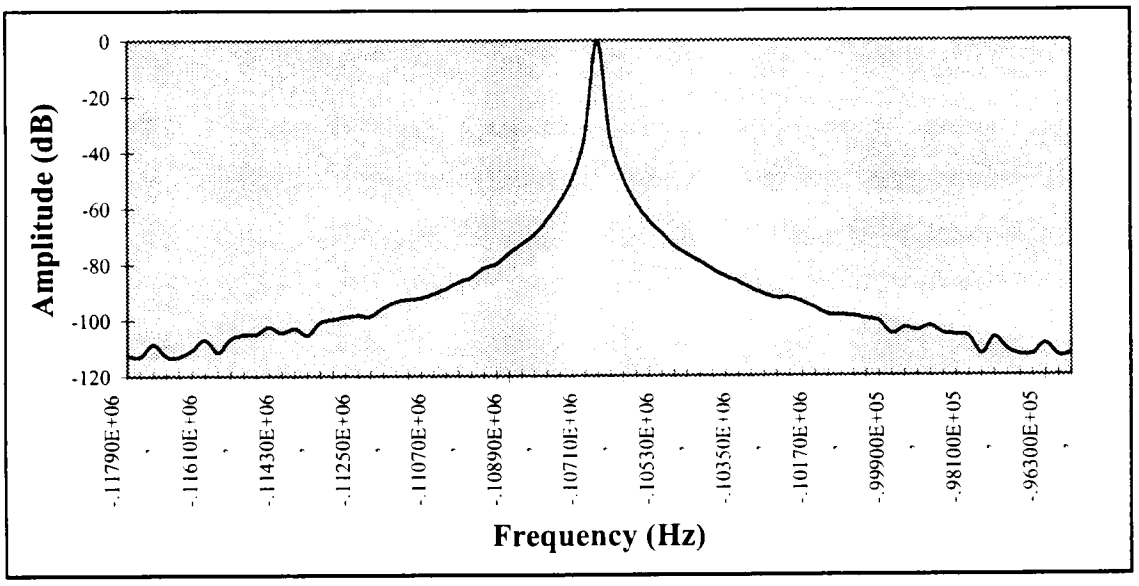


Figure 5-3: Front Tuned LO Spectrum for Low Altitude Condition

The illuminator detection block receives the downconverted rear antenna signal and processes the data in the time domain and the frequency domain to determine the illuminator frequency. The actual processing algorithms are classified, and thus cannot be discussed in detail. In a broad sense, the rear tuning LO sweeps the frequency band until the illuminator is found within the adjacent bandpass filter. Once a signal is found, several processing algorithms are performed to determine if the signal is actually the illuminator. If the signal is found not to be the illuminator, the sweeping algorithm resumes. If the signal is the illuminator, the signal is centered in the bandpass filter and the front tuning LO is adjusted to the desired frequency to produce the doppler frequency band.

Due to the classified nature of the processing algorithms, the receiver model could not include the algorithms. Thus, the actual receiver model, included in Appendix B, detects the center frequency of the rear input spectrum directly without including any of the downconversion processing and creates an LO centered at that frequency. The illuminator detection block conceptually produces a pure LO to drive the last mixer stage for the front antenna signal path. Since the noise of the rear input spectra, the fixed LO, and the rear tuning LO will not be directly injected in the output doppler video spectrum, there was no need, for the purpose of this thesis, to model the detection function in detail.

5.2. Simulation Results

The front antenna signal path is unchanged from the previous receiver to this receiver. The front antenna spectrum for the high altitude low clutter condition is shown Figure 5-4. Notice that this spectrum closely resembles the spectrum in Figure 4-3. The baseband doppler video spectrum is produced by convolving the downconverted front antenna spectrum with the tuned LO. The spectrum for the baseband doppler video signal is shown in Figure 5-5. The target has an approximate 50dB signal to clutter ratio. Notice that this baseband doppler video spectrum also resembles the previous receiver spectrum; however, the noise is limited by the thermal noise floor, rather than received clutter. Actually, the doppler video spectrum is almost identical to the downconverted front antenna spectrum. This is due to the low phase noise of the tuned LO.

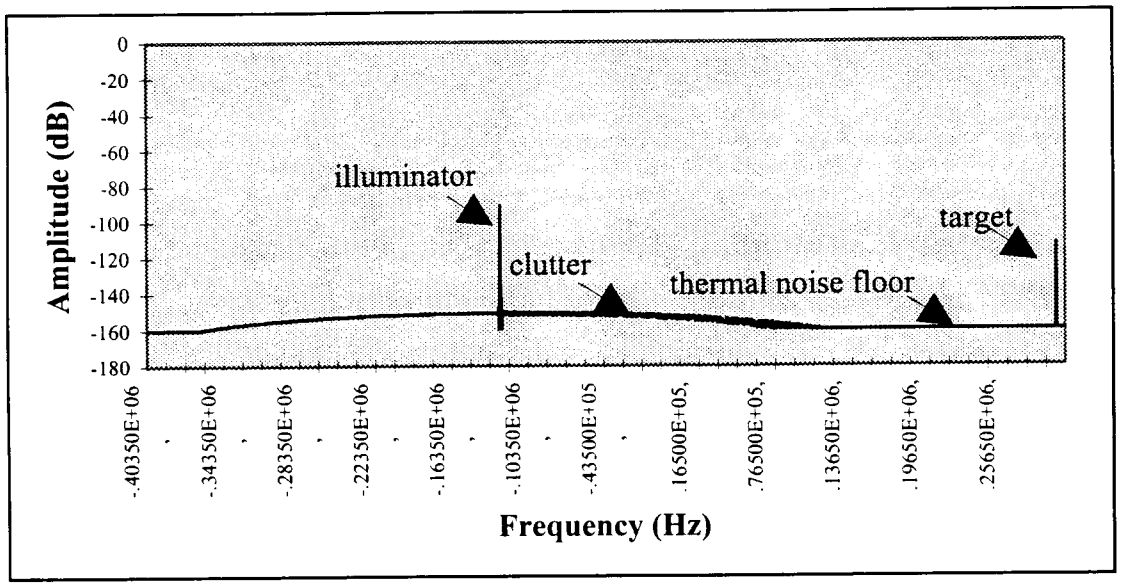


Figure 5-4: Downconverted Front Antenna Spectrum for High Altitude Condition

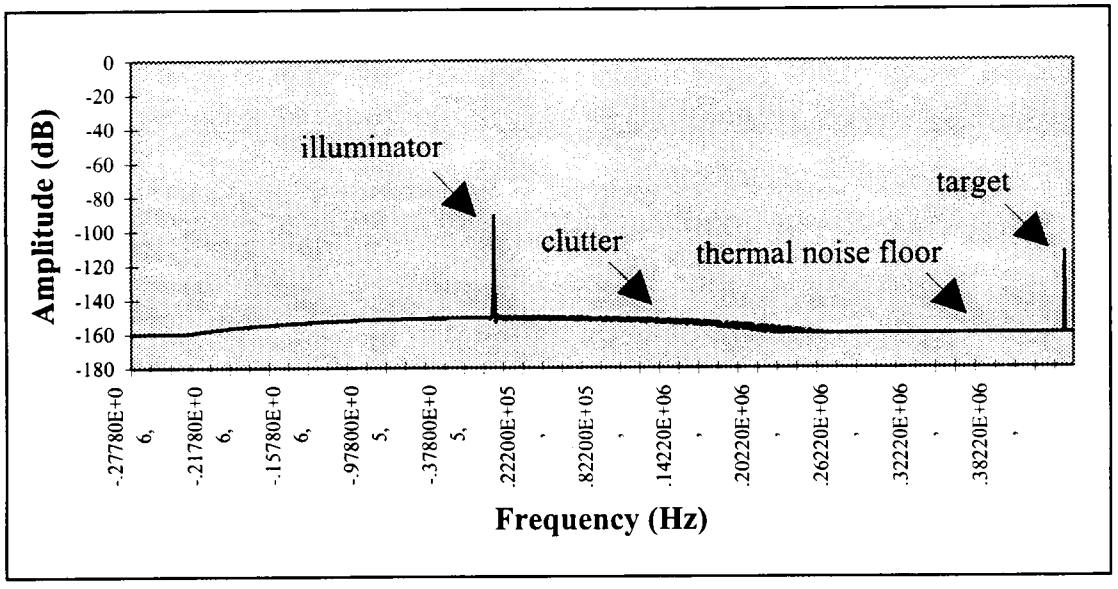


Figure 5-5: Baseband Doppler Video Spectrum for High Altitude Condition

The downconverted front antenna spectrum for the low altitude high clutter condition is shown in Figure 5-6. Again, the spectrum resembles the previous receiver spectrum for the downconverted front antenna signal shown in Figure 4-7. The downconverted front antenna spectrum is then convolved with the tuned LO shown in Figure 5-3, to produce the baseband doppler video spectrum shown in Figure 5-7. The target to clutter noise ratio is approximately 40dB. This is a substantial improvement over the 5dB ratio found with the previous receiver. The guidance algorithm can clearly detect the target, thus, resulting in target destruction.

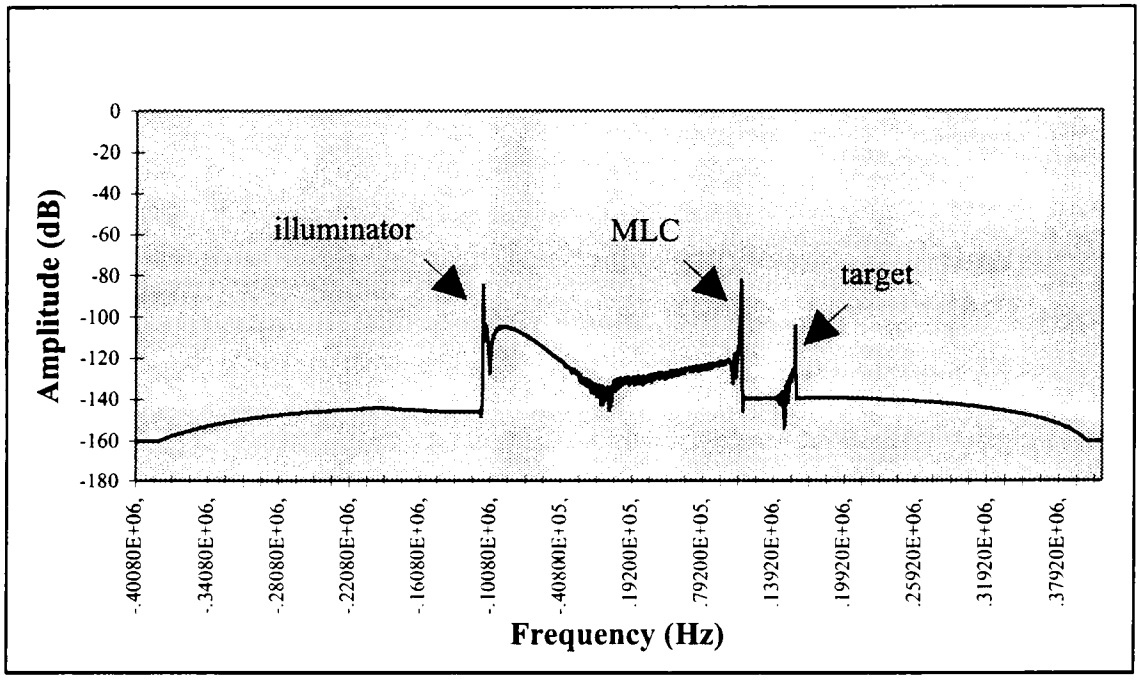


Figure 5-6: Downconverted Front Antenna Spectrum for Low Altitude Condition

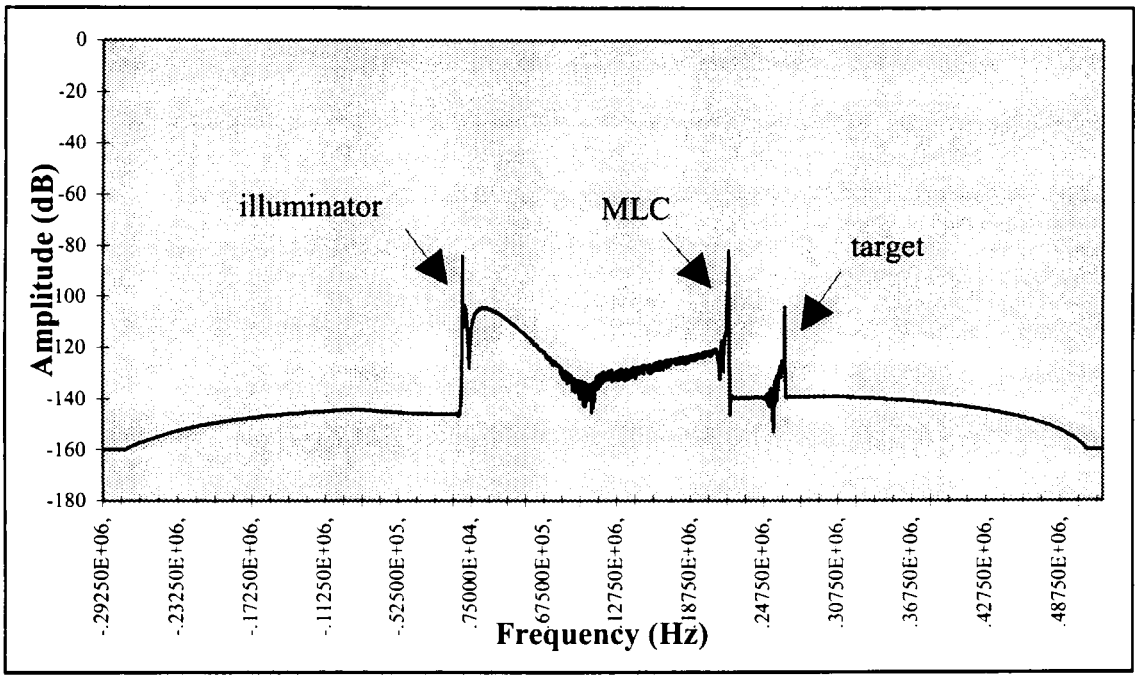


Figure 5-7: Baseband Doppler Video Spectrum for Low Altitude Condition

6. CONCLUSION

This study evaluated the performance of two receiver models for semiactive radar guided missile systems. The models were evaluated by producing two sets of test stimuli. The first set of stimuli was the high altitude condition which would produce a spectrum with very low clutter levels, thus was called the control case. The second set of stimuli was the low altitude condition which would produce a spectrum with very high clutter levels, thus was called the test case. It was expected that the first receiver model would have problems with the low altitude condition. It was also expected that the second receiver model would correct problems with the first model to improve the performance with the low altitude condition. The receiver models performed as expected, thus the simulation was successful.

6.1. Comparison of Receiver Models

The two receiver models discussed in Chapters 4 and 5 function similarly with the exception of the processing of the rear reference signal path. The general receiver model used the rear reference signal to create the baseband doppler video spectrum. The physical mixer required a constant signal strength for the LO input. Thus, the rear signal was amplified and limited to satisfy this requirement. The alternative receiver model only detected the frequency of the rear reference signal and generated an LO with a fixed amplitude to create the baseband doppler video spectrum.

The limiter function in the general receiver model produced dual sideband clutter on the illuminator signal. When the limited rear spectrum was mixed or convolved with the front antenna spectrum, the dual sideband clutter was now embedded with all components of the spectrum, including the MLC. The sideband noise on MLC raised the noise levels around the target to the extent that the target could no longer be detected. The elimination of the limiter function improved the target detection by approximately 35dB. This improvement would allow the guidance algorithm to detect the target and fulfill its mission.

6.2. Future Work

Continued studies are recommended which should include evaluating the performance of a digital receiver under the same test conditions. Also, an evaluation of intermediate clutter levels should be done to find the performance cutoff of each receiver model.

LIST OF REFERENCES

LIST OF REFERENCES

1. Barton, David K., *Modern Radar System Analysis*, Artech House, Massachusetts, 1988.

2. Barton, David K., *Radars*, Artech House, Massachusetts, 1979.

3. Edde, Byron, *Radar Principles Technology Applications*, Prentice Hall, New Jersey, 1993.

4. Hovenessian, S. A., *Radar Detection and Tracking Systems*, Artech House, Massachusetts, 1973.

5. Nathanson, F. E., *Radar Design Principles*, McGraw-Hill, New York, 1969.

6. Ridenour, L. N., *Radar System Engineering*, McGraw-Hill, New York, 1947.

7. Skolnik, M. L., *Introduction to Radar Systems*, McGraw-Hill, New York, 1980.

8. Skolnik, M. L., *Radar Handbook*, McGraw-Hill, New York, 1970.

9. Kraus, J. D., *Antennas*, McGraw-Hill, New York, 1988.

10. Lewis, P. E. and Ward, J. P., *Vector Analysis for Engineers and Scientists*, Addison-Wesley, New York, 1989.

11. Evans, J., "Clutter Environment Computer Simulation and Results," Raytheon Company, Missile Systems Division, GSD-130-82, 26 May 1982.

12. Veitschegger, W. K., "Sub-Clutter Visibility Verification Report," Raytheon Company, Missile Systems Division, SM-2:ID:0322, 4 October 1989.

13. Veitschegger, W. K., "Sub-Clutter Visibility Receiver Model Report," Raytheon Company, Missile Systems Division, SM-2:ID:0320, 4 October 1989.

14. Burrus, C. S. and Parks, T. W., *DFT/FFT and Convolution Algorithms*, Wiley-Interscience, 1985.

15. Kundth, D., *The Art of Computer Programming*, Vol. 2, Addison-Wesley, New York, 1979.

16. Hejazi, S., "Rear Reference Digital Signal Processor Design Description," Raytheon Company, Missile Systems Division, SEO:95:0836, 12 June 1995.

APPENDICES

APPENDIX A

Program Listing of General Receiver Model

```
C     Program: RCVR1
C     Author:  Alan Weddle
C     Date:    4/1/96
C     Update:
C
C
C
C-------------------------Variable definitions-------------------------
C
C          All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-          -DESCRIPTION-
C
C     ISEED   NONE       RANDOM number generator primer
C
C     X()    V~          Real part of data file
C     Y()    V~          Imaginary part of data file
C     DX()   VARIOUS     Data header
C     A()    V~          Real part of a spectrum
C     B()    V~          Imaginary part of a spectrum
C     DA()   VARIOUS     Data header for the spectrum
C     E()    V~          Real part of a spectrum
C     F()    V~          Imaginary part of a spectrum
C     DE()   VARIOUS     Data header for the spectrum
C     G()    V~          Real part of a spectrum
C     H()    V~          Imaginary part of a spectrum
C     DG()   VARIOUS     Data header for the spectrum
C     NAME   NONE        Name of data file
C     NF     dB          Noise Figure
C     CF     Hz          Filter center frequency
C     BW     Hz          Filter 3 dB bandwidth
C     RBW    Hz          Rejection Bandwidth
C     REJ    Hz          Rejection of rejection bandwidth
C     ULT    dB          Filter ultimate rejection
C     TD     sec         Time delay
C     RERUN  BOOLEAN     Flag to start the pass again new NTHRESH
C     NTHRESH NONE       Scaling factor for noise threshold
C
C        V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-----------------------------------------------------------------------
C-------------------------Variable declarations-------------------------
C
```

```
      PROGRAM RCVR1

      IMPLICIT NONE
C
      REAL X(0:8192), Y(0:8192), DX(4)
      REAL A(0:8192), B(0:8192), DA(4)
      REAL C(0:8192), D(0:8192), DC(4)
      REAL E(0:8192), F(0:8192), DE(4)
      REAL RE(0:8192), IM(0:8192), DR(4)
C
      REAL NFF, NFR
      REAL DELTAF
      REAL TARGET
C
      INTEGER SEEDS(100),TIMES,ISEED
      INTEGER I, J, K, PASS
      INTEGER START

C
      CHARACTER*40 NAME

      NFF = .10
      NFR = .10
      TARGET = 0.
C
C-----INPUT PROGRAM OPTIONS DATA---------------------------------
C
      OPEN(UNIT=1,FILE='CLUTTER\INPUTS\SEEDS',STATUS='OLD')
C
      READ(1,20) TIMES
      DO 50 I=1,TIMES
      READ(1,20) SEEDS(I)
50    CONTINUE
C
      CLOSE(UNIT=1)
C
20    FORMAT(I10)
C
C
C
      DR(1) = 0.0
      DR(2) = 0.0
      DR(3) = 0.0
```

```
    DR(4) = 0.0
C
C-----ZERO THE REAL AND IMAGINARY ARRAYS-------------------------
C
    DO 100 I = 0, 8192
      RE(I) = 0.0
      IM(I) = 0.0
100   CONTINUE
C
    DO 1000 PASS=1,TIMES
      PRINT *, PASS, TIMES
C
C-----ENTER REAR ANTENNA SPECTRUM--------------------------------
C
101   ISEED = SEEDS(PASS)
    NAME = 'CLUTTER\DATA\SPECR'
    CALL DATIN(NAME, C, D, DC, ISEED)
C
C-----ADD NOISE TO REAR SPECTRUM---------------------------------
C
    CALL NOISE(DC, NFR)
    CALL REDUCE(C, D, DC)
    NAME = 'CLUTTER\DATA\IREAR.DAT'
    CALL NPSC(NAME, C, D, DC)
C
C-----INCLUDE SSLO NOISE-----------------------------------------
C
    DELTAF = DC(3)
    CALL SN(A, B, DA, E, F, DE, DELTAF, ISEED, 2)
    NAME = 'CLUTTER\DATA\ISSLO.DAT'
    CALL NPSC(NAME, A, B, DA)
C
C-----CONVOLVE THE REAR SPECTRUM WITH THE SSLO--------------------
C
    CALL CONVL(C, D, DC, A, B, DA)
    NAME = 'CLUTTER\DATA\IREARSS.DAT'
    CALL NPSC(NAME, C, D, DC)
C
C-----LIMIT REAR SPECTRUM-----------------------------------------
C
    CALL LIMIT(C, D, DC, X, Y, DX)
    NAME = 'CLUTTER\DATA\RLIM.DAT'
    CALL NPSC(NAME, C, D, DC)
C
```

```
C-----ENTER ANTENNA DATA-----------------------------------------
C
      NAME = 'CLUTTER\DATA\SPECF'
      CALL DATIN(NAME, X, Y, DX, ISEED)
C
C-----ADD NOISE TO FRONT SPECTRUM--------------------------------
C
      CALL NOISE(DX, NFF)
      CALL REDUCE(X, Y, DX)
      NAME = 'CLUTTER\DATA\ISUM.DAT'
      CALL NPSC(NAME, X, Y, DX)
C
C-----CONVOLVE THE SSLO WITH THE FRONT SPECTRUM-------------------
C
      CALL CONVL(X, Y, DX, E, F, DE)
      NAME = 'CLUTTER\DATA\IFRNTSS.DAT'
      CALL NPSC(NAME, X, Y, DX)
C
C-----CONVOLVE FRONT AND REAR SPECTRA-----------------------------
C
      CALL CONVL(X, Y, DX, C, D, DC)
      NAME = 'CLUTTER\DATA\ICONV.DAT'
      CALL NPSC(NAME, X, Y, DX)
C
C-----CALCULATE TEN PASS AVERAGE----------------------------------
C
    IF(PASS .EQ. 1)THEN
      DR(1) = DX(1)
      DR(2) = DX(2)
      DR(3) = DX(3)
      DR(4) = DX(4)
    END IF
    IF(DX(2).GT.DR(2))THEN
      J = INT((DX(2) - DR(2))/DX(3))
      DR(2) = DX(2)
      K = INT(DR(1))
      DR(1) = DR(1) - REAL(J)
      DO 110 I=J,K
          RE(I-J) = RE(I)
          IM(I-J) = IM(I)
110     CONTINUE
    ELSE
      J = INT((DR(2) - DX(2))/DX(3))
      DX(2) = DR(2)
```

```
      K = INT(DX(1))
      DX(1) = DX(1) - REAL(J)
      DO 120 I=J,K
         X(I-J) = X(I)
         Y(I-J) = Y(I)
120     CONTINUE
    END IF
C
    IF(DX(1).GT.DR(1))THEN
      DX(1) = DR(1)
    ELSE
      DR(1) = DX(1)
    END IF
C
    K = REAL(DR(1))
    DO 130 I=0,K
      RE(I) = RE(I) + X(I)**2 + Y(I)**2
130  CONTINUE
C
1000  CONTINUE

    IF (DR(1) .GT. 5000) THEN
      IF (DR(2) .LT. 0.) THEN
        START = INT( TARGET - DR(2)/DR(3) )
        START = AMAX0(START, 0)
      ELSE
        START = 0
      END IF
      DO 133, I = START, DR(1)
        RE(I-START) = RE(I)
        IM(I-START) = IM(I)
133     CONTINUE
      DR(1) = DR(1) - START
      K = RE(DR(1))
    END IF

    DO 140 I=0,K
      RE(I) = SQRT(RE(I)/REAL(TIMES))
      IM(I) = 0.0D0
140  CONTINUE
C
    NAME = 'CLUTTER\DATA\AVE.DAT'
    CALL NPSC(NAME,RE,IM,DR)
C
```

```
C-----End program-----------------------------------------------
C
    STOP

    END

C
C**************************SUBROUTINE
LIMIT*****************************
C
      SUBROUTINE LIMIT(X,Y,DX,A,B,DA)
C
C     This program determines the output of a limiter given an input
C     spectrum.
C
C-------------------------Variable definitions-------------------------
C
C           All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE       Integer counter
C     J      NONE       Integer counter
C     M      NONE       Integer counter
C     K      NONE       Integer counter
C     DX(1)  NONE       Total number resolution bandwidths
C     NUMB1  NONE       Total number of input data points created
C     NUMB2  NONE       Total number of output data points
C     NUM    NONE       2**M data space used in processing
C     NUM1   NONE       2**M processing number minus 1
C     DX(2)  Hz         Starting frequency
C     DX(3)  Hz         Step frequency or resolution bandwidth
C     DX(4)  V~         Input noise floor
C     X()    V~         Real part of the data
C     Y()    V~         Imaginary part of the data
C     THRESH V~         Limiter threshold level
C     THETA  RADIANS    Phase of the input signal
C     NRATIO V~/V~      Peak to noise ratio
C
C       V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-----------------------------------------------------------------------
C-------------------------Variable declarations------------------------
```

```
C
      IMPLICIT NONE
C
      INTEGER I, J, M, NUM, NUM1, K
      INTEGER NUMB1
C
      REAL T1, T2, MAG
      REAL X(0:*), Y(0:*), DX(*), POUT, PIN
      REAL A(0:*), B(0:*), DA(*)
C
C----------------------------Define constants----------------------------
C
      NUMB1 = INT(DX(1))
      M = 13
      NUM = 2**M
      NUM1 = NUM-1
C
C----------------------------Prepare input data----------------------------
C
      PIN = 0.0
      T2  = 0.0
      K   = 0
      DO 150 I=0,NUMB1
            T1 = X(I)**2 + Y(I)**2
            PIN = PIN + T1
            IF(T1.GT.T2) THEN
                  K = I
                  T2 = T1
            END IF
150    CONTINUE
C
      J = NUM/2 - K
      DO 200 I=J,J+NUMB1
            A(I) = X(I-J)
            B(I) = Y(I-J)
200    CONTINUE
C
      DO 250 I=0,J-1
            A(I) = 0.0
            B(I) = 0.0
250    CONTINUE
C
      J = J+NUMB1+1
      DO 300 I=J, NUM1
```

```
          A(I) = 0.0
          B(I) = 0.0
300     CONTINUE
C
      DO 350 I=0,NUM1
          X(I) = A(I)
          Y(I) = B(I)
350     CONTINUE
C
C------------------------Calculate time domain input----------------------
C
      CALL DFFT(X,Y,NUM,M,-1)
C
C------------------------Calculate time domain output---------------------
C
      DO 500 I=0,NUM1
              MAG =  SQRT(X(I)**2 + Y(I)**2)
              X(I) = X(I)/MAG
              Y(I) = Y(I)/MAG
500     CONTINUE
C
C-------------------Convert to frequency and truncate data-----------------
C
      CALL DFFT(X,Y,NUM,M,1)
C
      POUT = 0.0
      T2 = 0.0
      DO 600 I=0,NUM1
          T1 = X(I)**2 + Y(I)**2
          POUT = POUT + T1
          T2 = AMAX1(T1,T2)
600     CONTINUE
C
      T2 = SQRT(T2)
      DO 700 I=0,NUM1
          X(I) = X(I)/T2
          Y(I) = Y(I)/T2
700     CONTINUE
C
      DX(4) = DX(4) * SQRT(POUT/PIN) / T2
C
C-----------------------Generate output frequency data--------------------
C
      DX(1) =  REAL(NUM1)
```

```
      DX(2) =  DX(2) + DX(3) * REAL(K - NUM/2)
C
C
C
C       T1 = DX(3)*DX(1) - 376000.0
      T1 = DX(3)*DX(1) - 1128000.0
C
      IF (T1 .GT. 0.0) THEN
          T1 = T1/DX(3)/2.0
          J = NINT(T1)+1
          DO 111 I=J,NUM1-J
              X(I-J) = X(I)
              Y(I-J) = Y(I)
111         CONTINUE
          DX(1) = REAL(NUM1-2*J)
          DX(2) = DX(2) + DX(3) * REAL(J)
      END IF
C
C-----------------Reduce limitation to significant data only----------------
C
      CALL REDUCE(X,Y,DX)
C
C-------------------------------End program-----------------------------
C
      RETURN
      END
     SUBROUTINE LPF(F,MAG,PHI)
C
     IMPLICIT NONE
C
C
C
     REAL GR, GI, DR, DI, NR, NI, F, PHI, TPI, MAG
     REAL S1, S2, S3, S4
C
C
C
     TPI = 2.0 * ACOS(-1.0)
     PHI = TPI * F
     S1 = PHI
     S2 = PHI**2
     S3 = PHI**3
     S4 = PHI**4
C
```

```
C
C
      NR = 1.0
      NI = 2.08E-3 * S1
C
      DR=1.0D0 - 2.16E-6*S2 + 4.01E-13*S4
      DI=2.08E-3*S1 - 1.32E-9*S3
C
      MAG = SQRT(DR**2 + DI**2)
      PHI = ATAN2(DI,DR)
C
      MAG = 1.0D0/MAG
      PHI = -PHI
      DR = MAG * COS(PHI)
      DI = MAG * SIN(PHI)
C
      GR = NR * DR - NI * DI
      GI = NR * DI + NI * DR
C
      MAG = SQRT(GR**2 + GI**2)
      PHI = ATAN2(GI,GR)
C
C
C
      RETURN
      END
C
C**************************SUBROUTINE
NOISE*****************************
C
      SUBROUTINE NOISE(DX,NF)
C
C     This program adds noise to a complex unit spectrum.
C
C-------------------------Variable definitions----------------------------
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE    Integer counter
C     DX(1)  NONE    Number of resolution bandwidths
C     DX(2)  Hz      Starting frequency of spectrum
C     DX(3)  Hz      Step frequency or resoution bandwidth
C     DX(4)  V~      Noise floor
C     K      J/K     Boltsman's constant
```

```
C     TE     K       Effective temperature
C     NF     dB      Noise figure
C     NOIS   Watts   Noise power level
C
C        V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------------
C-------------------------Variable declarations---------------------------
C
      IMPLICIT NONE
C
      REAL DX(*), K, TE, NF
      REAL NOIS
C
C-------------------------Calculate and add noise-------------------------
C
      NF = 10.0**(NF / 10.0)
C
      K = 1.38E-23
C
      TE = (NF - 1.0) * 290.0
C
      NOIS = DX(3) * K * TE * 1000.0
C
      DX(4) = SQRT(DX(4)**2 + NOIS)
C
C-------------------------------End Program-------------------------------
C
      RETURN
      END
     SUBROUTINE NPSC(NAME,X,Y,DX)
C
C     Program: Power Spectrum Converter (PSC)
C
C     This program converts an amplitude spectrum into
C     a power spectrum.
C
C-------------------------Variable definitions----------------------------
C
C    -UNITS-  -NAME-              -DESCRIPTION-
C
C     I,J,K   NONE        Integer counters
C     NUMB    NONE        Number of data points
C     X()     V~          Real part of the input spectrum
```

```
C     Y()   V~        Imaginary part of input spectrum
C     DX(1)  NONE      Total number of data points
C     DX(2)  Hz        Start frequency
C     DX(3)  Hz        Resolution bandwidth
C     DX(4)  V~        Input noise
C     FX     Hz        Frequency
C     INPUT  NONE      Input spectrum name
C
C       V~ = VOLTS / SQUARE ROOTS OF OHMS
C
C-------------------------------------------------------------------------
C-------------------------Variable declarations---------------------------
C
      IMPLICIT NONE
C
      INTEGER NUMB, I
C
      REAL X(0:*), Y(0:*), DX(*), FX, T1, T2
C
      CHARACTER*40 NAME
C
      NUMB = INT(DX(1))
C
      NUMB = MIN0(NUMB, 8188)
C
C
      OPEN (UNIT=1,FILE=NAME,STATUS='UNKNOWN')
      REWIND(1)
C
      WRITE(1,30) DX(1)
      WRITE(1,30) DX(2)
      WRITE(1,30) DX(3)
C
C-----------------------Calculate power spectrum-------------------------
C
      IF (DX(4) .LE. 1.0E-15) THEN
           T1 = 20.0 * ALOG10(1.0E-15)
      ELSE
           T1 = 20.0 * ALOG10(DX(4))
      END IF
      WRITE(1,30) T1
C
C
C
```

```
      DO 200 I = 0,NUMB
           T2 = X(I)**2 + Y(I)**2
C
           IF(T2 .LT. 1.0E-20) THEN
                T2 = 1.0E-20
           END IF
C
           T2 = 10.0 * ALOG10(T2)
C
           FX = DX(2) + DX(3) * REAL(I)
           WRITE(1,50) FX, T2
200    CONTINUE
C
      ENDFILE(1)
      CLOSE (UNIT = 1)
C
C-------------------------------Formats---------------------------------
C
30     FORMAT(1X,E12.5)
50     FORMAT(1X,E12.5,1H,,E12.5)
C
C------------------------------End program-------------------------------
C
      RETURN
      END
C
C*****************************SUBROUTINE
RANDOM*****************************
C
      SUBROUTINE RANDOM (X,SEED)
C
C     This program was taken from 'THE ART OF COMPUTER PROGRAMING'
C     Vol. 2, D. Knuth, Addison-Wesley
C
C     This program generates random numbers based on the input Seed
C
C-------------------------Variable definitions--------------------------
C
C     X      Random number
C     SEED   Seed
C     MULT   Seed multiplier
C     BASE   Normalization base
C
C-------------------------Variable declarations-------------------------
```

```
C
      IMPLICIT NONE
C
      REAL X
C
      INTEGER SEED
C
C-------------------------Generate random number-------------------------
C
      SEED = MOD(25211 * SEED,32768)
      X = REAL(SEED) / 32768.
C
C-----------------------------End program------------------------------
C
      RETURN
      END
C
C****************************SUBROUTINE
REDUCE*****************************
C
      SUBROUTINE REDUCE(X,Y,DX)
C
C     This program reduces a data file to data significantly above
C     the input noise floor.
C
C--------------------------Variable definitions-------------------------
C
C           All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE        Integer counter
C     J      NONE        Integer counter
C     NUMB   NONE        Total number of data points
C     DX(1)  NONE        Total number of frequency steps
C     DX(2)  Hz          Start frequency
C     DX(3)  Hz          Step frequency or resolution bandwidth
C     DX(4)  V~          Input noise floor
C     NOISE  WATTS       Input noise floor
C     SIGNAL WATTS       Signal power of interest
C     X()    V~          Real part of the data
C     Y()    V~          Imaginary part of the data
C
```

```
C         V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------------
C----------------------------Variable declarations-------------------------
C
      IMPLICIT NONE
C
      INTEGER I, J, K, NUMB
C
      REAL DX(*), X(0:*), Y(0:*)
      REAL NOISE, SIGNAL
C
      NUMB = INT(DX(1))
C
C--------------------------------Reduce Data-----------------------------
C
      NOISE = DX(4)**2
C
      DO 200 I=0,NUMB
C
            SIGNAL = X(I)**2 +Y(I)**2
C
C----------SIGNAL MUST BE HIGHER THAN THE AVERAGE NOISE

            IF (SIGNAL .GT. NOISE) THEN
                  IF (I .GT. 19) THEN
                        K = I - 20
                  ELSE
                        K = 0
                  END IF
                  DX(2) = DX(2)+FLOAT(K)*DX(3)
                  DO 400 J=K,NUMB
                        X(J-K) = X(J)
                        Y(J-K) = Y(J)
400                 CONTINUE
                  NUMB = NUMB - K
                  GO TO 3000
            END IF
200      CONTINUE
C
3000     CONTINUE
C
      DO 500 I=0,NUMB
C
```

```
          SIGNAL = X(NUMB-I)**2 + Y(NUMB-I)**2
C
           IF (SIGNAL .GT. NOISE) THEN
                IF (I .GT. 19) THEN
                     K = I - 20
                ELSE
                     K = 0
                END IF
                NUMB = NUMB - K
                GO TO 2000
           END IF
500    CONTINUE
C
2000   CONTINUE
C
      DX(1) = NUMB
C
C------------------------------End program---------------------------------
C
      RETURN
      END


     SUBROUTINE SN(X ,Y, DX, A, B, DA, RB, SEED, OPTION)
C
     IMPLICIT NONE
C
     INTEGER I, M, NUM, NUM1, HNUM1, SEED, OPTION
C
     REAL R, RB, F
     REAL SCALE, TAU, LASTX, LASTY, SAMPRATE
C
     REAL X(0:*), Y(0:*), NOISE, A(0:*), B(0:*)
     REAL TPI, T, DX(*), DA(*)
C
C-----Set constants-------------------------------------------------
C
     TAU = 50.E-9

     M = NINT(ALOG10(220000.0/REAL(RB))/ALOG10(2.0))
     M = M + 1
     NUM = 2**(M)
     NUM1 = NUM - 1
     HNUM1 = NUM1/2
```

```
C
      SAMPRATE = 1. / 220.E3
C
      DX(1) = REAL(NUM1)
      DX(2) = REAL(-NUM/2) * RB
      DX(3) = RB
C
C-----COPY SPECTRUM 1 PARAMETERS TO SPECTRUM 2 PARAMETERS
C
      DA(1) = DX(1)
      DA(2) = DX(2)
      DA(3) = DX(3)
      DA(4) = DX(4)
C
      TPI = 2.0 * ACOS(-1.0)
C
C-----Create the reference oscillator signal----------------------
C
      DO 100 I=0,NUM1
        X(I) = 0.0
        Y(I) = 0.0
100   CONTINUE
C
C
      IF (OPTION .EQ. 1) THEN
C
        SCALE = SQRT( RB / 1000. )

        DX(4) = 10.**((-9. + ALOG10(RB))/2.0)
        DA(4) = DX(4)
C
        DO 200 I=1,HNUM1
          F= RB * REAL(I)
          IF (F .LT. 1000.0) THEN
            Y(I) = REAL(0.17782794) * SCALE
          ELSE IF (F .GE. 1000. .AND. F .LT. 10000.) THEN
            Y(I) = REAL(0.17782794) * SCALE /
     .           ( F/1000. )
          ELSE IF (F .GE. 10000 .AND. F .LT. 1.2E5) THEN
            Y(I) = 3.758374E-3 * SCALE /
     .           ( F/10000. )
          ELSE IF (F .GT. 1.2E5 .AND. F .LT. 2.6E5) THEN
            Y(I) = 10.**(( -9. + ALOG10(RB) ) / 2.) * SCALE
          END IF
```

```
200     CONTINUE
C
    ELSE
      DX(4) = 10.**((-12.2 + ALOG10(RB))/2.0)
      DO 250 I=0,HNUM1
        F = RB * REAL(I)
        Y(I) = -85. - 49.8 * ALOG10(F/1.0E3 + 1.0E-16)
        Y(I) = Y(I) + 10. * ALOG10(RB)
        Y(I) = 10.**(Y(I)/20.)
250     CONTINUE
        Y(0) = 0.0
C
    END IF
C
C-----ADD PHASE NOISE AND MAKE SURE THE TIME DOMAIN WILL BE REAL
(THE
C       UPPER HALF OF THE FFT IS THE COMPLEX CONJUGATE OF THE
LOWER.)
C
    DO 500 I=1,HNUM1
      CALL RANDOM(R,SEED)
      NOISE = Y(I)
      X(I) = NOISE * SIN(TPI * R)
      Y(I) = NOISE * COS(TPI * R)
      X(NUM-I) =  X(I)
      Y(NUM-I) = -Y(I)
500   CONTINUE
C
    CALL DFFT(X,Y,NUM,M,-1)
C
C-----Phase modulate frequency centered source----------
C
    DO 700 I=0,NUM1
      T = X(I) + REAL(I)*TPI/2.0
      X(I) = COS(T)
      Y(I) = SIN(T)
C
C--------MAKE THE SECOND SSLO SLIGHTLY TIME SHIFTED FROM THE FIRST
BY
C         USING LINEAR INTERPOLATION BETWEEN TWO POINTS.
C
      IF (I .GT. 1) THEN
        A(I-1) = LASTX + TAU * (X(I)-LASTX) / SAMPRATE
        B(I-1) = LASTY + TAU * (Y(I)-LASTY) / SAMPRATE
```

```
      END IF
      LASTX = X(I)
      LASTY = Y(I)

700   CONTINUE
C
     A(0) = 0.
     B(0) = 0.
     A(NUM1) = 0.
     B(NUM1) = 0.
C
C-----Convert to frequency domain---------------------------------
C

     CALL DFFT(X,Y,NUM,M,1)
     CALL REDUCE(X,Y,DX)
     CALL DFFT(A,B,NUM,M,1)
     CALL REDUCE(A,B,DA)
C
C-----End Program-------------------------------------------------
C
     RETURN
     END
C
C**************************SUBROUTINE
CONVOLVE*****************************
C
      SUBROUTINE CONVL(X,Y,DX,A,B,DA)
C
C     This program convolves two spectra producing down mix product
C
C--------------------------Variable definitions--------------------------
C
C           All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE      Integer counter
C     J      NONE      Integer counter
C     K       NONE      Integer counter
C     DX(1)  NONE      Number of frequency steps in the X spectrum
C     NUMX   NONE      Total number of data points in the X spectrum
C     DX(2)  Hz        Start frequency of the X spectrum
```

```
C     DX(3)  Hz       Resolution bandwidth of the X spectrum
C     DX(4)  V~       Noise floor of the X spectrum
C     X()    V~       Real data for the X spectrum
C     Y()    V~       Imaginary data for the X spectrum
C     DA(1)  NONE     Number of frequency steps in the A spectrum
C     NUMA   NONE     Total number of data points in the A spectrum
C     DA(2)  NONE     Start frequency of the A spectrum
C     DA(3)  Hz       Resolution bandwidth of the A spectrum
C     DA(4)  V~       Noise floor of the A spectrum
C     A()    V~       Real data for the A spectrum
C     B()    V~       Imaginary data for the A spectrum
C     NUMo   NONE     Number of data points in the output spectrum
C     M      NONE     FFT radix exponent
C     NUM    NONE     FFT array length
C     NUM1   NONE     FFT array length minus one
C     W,Z    NONE     Temporary data storage
C     NI()   V~       Imaginary part of the noise calcualtion
C     NR()   V~       Real part of the noise calculation
C     NOISEX dB       Noise power caused by the X spectrum
C     NOISEA dB       Noise power caused by the A spectrum
C     TPI    RADIANS  2.0 * 3.14159.....
C     P      NONE     Random number
C
C       V~ = VOLTS/SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------
C----------------------------Variable declarations-------------------------
C
      IMPLICIT NONE
C
      INTEGER NUMX, NUMA, NUMo, NUM, NUM1,M
      INTEGER I, J
C
      REAL TPI, NOISEX, NOISEA
      REAL X(0:*), Y(0:*), DX(*)
      REAL A(0:*), B(0:*), DA(*)
      REAL W, Z
C
C-----------------------------Define constants----------------------------
C
      TPI = 2.0 * ACOS(-1.0)
C
      NUMX = INT(DX(1))
      NUMA = INT(DA(1))
```

```
C
C-----------------------Determine data compatability----------------------
C
      IF (DX(3) .NE. DA(3)) THEN
           PRINT *,'DX(3)=',DX(3),'DA(3)=',DA(3)
           PRINT *, 'FREQUENCY STEP NOT COMPATABLE'
           PAUSE
           GO TO 3000
C
C                *line 3000 is at the end of the program*
C
      END IF
C
C-----------------------------Flip 'A' spectrum---------------------------
C
      J = INT(REAL(NUMA)/2.0)
      DO 200 I=0,J
           W = A(I)
           A(I) = A(NUMA-I)
           A(NUMA-I) = W
C
           W = B(I)
           B(I) = B(NUMA-I)
           B(NUMA-I) = W
200     CONTINUE
C
      DO 250 I=0,NUMA
           B(I) = -B(I)
250     CONTINUE
C
C-----------------------Calculate Spectrum 'X' Noise----------------------
C
           NOISEX = 0.0
           W = DA(4)**2
           DO 400 I=0,NUMX
             NOISEX = W * (X(I)**2 + Y(I)**2) + NOISEX
400          CONTINUE
C
C-----------------------Calculate Spectrum 'A' Noise----------------------
C
           NOISEA = 0.0
           W = DX(4)**2
           DO 700 I=0,NUMA
             NOISEA = W * (A(I)**2 + B(I)**2) + NOISEA
```

```
700          CONTINUE
C
C----------------------------Add spectrum noise----------------------------
C
      IF ( (NOISEA + NOISEX) .GT. 1.E-38) THEN
        DX(4) = SQRT(NOISEA + NOISEX)
      ELSE
        DX(4) = 1.E-19
      END IF
C
C-------------------------Calculate 2**M divisions-------------------------
C
      NUMo =INT(DX(1) + DA(1))
C
      M = NINT(ALOG(FLOAT(NUMo))/ALOG(2.0))
      NUM = 2.0**M
      NUM1 = NUM - 1
C
      IF (NUM.LT.NUMo) THEN
           M = M + 1
           NUM = 2.0**M
           NUM1 = NUM - 1
      END IF
C
C-------------------------Add zeros to spectrums-------------------------
C
      DO 900 I = NUMX+1,NUM1
           X(I) = 0.0
           Y(I) = 0.0
900    CONTINUE
C
      DO 1000 I = NUMA+1,NUM1
           A(I) = 0.0
           B(I) = 0.0
1000   CONTINUE
C
C----------------------------Calculate Convolution-------------------------
C
      CALL DFFT(X,Y,NUM,M,-1)
      CALL DFFT(A,B,NUM,M,-1)
C
      DO 1100 I=0,NUM1
           W = X(I) * A(I) - Y(I) * B(I)
           Z = X(I) * B(I) + Y(I) * A(I)
```

```
            X(I) = W
            Y(I) = Z
1100    CONTINUE
C
      CALL DFFT(X,Y,NUM,M,1)
C
C--------------------------Generate frequencies-------------------------
C
      DX(2) = DX(2) - (DA(2) + DA(3) * DA(1))
      DX(1) = NUMo
C
C---------------Reduce convolution to significant data only--------------
C
      CALL REDUCE(X,Y,DX)
C
C-----------------------------End program-----------------------------
C
3000    CONTINUE
C
      RETURN
      END
C
C*************************SUBROUTINE
INPUT*****************************
C
      SUBROUTINE DATIN(NAME,X,Y,DX,SEED)
C
C     This program converts the input signal from dBm to
C     Volts / square root of Ohms, adds a random phase shift to
C     each frequency bin, and includes the affects of illuminator
C     noise sidebands.
C
C--------------------------Variable definition-------------------------
C
C           All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE        Integer Counter
C     J      NONE        Integer counter
C     DX(1)  NONE          Number of spectrum data points
C     DX(3)  Hz          Resolution bandwidth
C     DX(2)  Hz          Starting frequency of spectrum
```

```
C     DX(4)  V~          Noise floor
C     X()    dBm, V~     Spectrum power, Real part
C     Y()    V~          Imaginary part of the spectrum
C     NUMB   NONE        Integer counter based on DX(1)
C     STOP   Hz          Stoping frequency
C     TDFT   sec         Feedthrough time delay
C     FTFRQ  sec         Feedthrough frequency
C     FTHR   dBm         Feedthrough power
C     RFTHR  V~          Real feedthrough
C     IFTHR  V~          Imaginary feedthrough
C     TDTG   sec         Target time delay
C     TGFRQ  Hz          Target frequency
C     TARG   dBm         Target power
C     RTARG  V~          Real target
C     ITARG  V~          Imaginary target
C     FREQ   Hz          Place holding frequency in the input array
C     NOISE  dBm         Input noise floor
C     DA(1)  NONE        Number of data points in the spectrum
C     DA(3)  Hz          Step Frequency of spectrum
C     DA(2)  Hz          Starting frequency of spectrum
C     DA(4)  V~          Noise floor
C     A()    V~          Real part of the spectrum
C     B()    V~          Imaginary part of the spectrum
C     NAME   NONE        Name of the power spectrum file
C     SEED   NONE        RANDOM number generator primer
C     P      NONE        Random variable for determining phase
C     TPI    NONE        2.0 * 3.14159.....
C
C       V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------------
C-------------------------Variable declarations--------------------------
C
      IMPLICIT NONE
C
      INTEGER I, J, NUMB, SEED
C
      REAL NUMB1, START, DF, STOP
      REAL FTHR, FTFRQ, TGFRQ, TARG
      REAL FREQ, NOISE, TDFT, TDTG, TDCL
      REAL RFTHR, IFTHR, RTARG, ITARG
      REAL P
      REAL TPI
C
```

```
      REAL X(0:*), Y(0:*), DX(*), FX
      CHARACTER test
C
      CHARACTER*40 NAME
C
C-------------------------Define constants-------------------------
C
      TPI = 2.0 * ACOS(-1.0)
C
C-------------------------Open input data file---------------------
C
      OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
C
C-------------------------Read data format-------------------------
C
      READ(1,1000) NUMB1,test,START
      READ(1,1000) DF,test,STOP
      READ(1,1000) FTFRQ,test,FTHR
      READ(1,1000) TGFRQ,test,TARG
      READ(1,1500) TDFT,TDTG
      READ(1,3000) TDCL,NOISE
      READ(1,1000) FREQ,test,NOISE
      NUMB = INT(NUMB1)

C
C-------------------Increase spectrum width for target-------------------
C
      IF(TGFRQ .GT. STOP)THEN
          J = INT((TGFRQ - START)/DF)
          NUMB1 = REAL(J)
      END IF
C
      DO 100 I=0,J
          X(I) = 0.0
          Y(I) = 0.0
100   CONTINUE
C
C-------------------------Read spectrum data-----------------------
C
      DO 200 I=0,NUMB
           READ(1,1000) FX,test,X(I)
200   CONTINUE
C
      CLOSE (UNIT = 1)
```

```
C
      DX(1) = NUMB1
      DX(2) = START
      DX(3) = DF
      DX(4) = 0.0
C
C
C----------------------Convert to volts/sqrt(ohms)-----------------------
C
      FTHR = 10.0**(FTHR/20.0)
C
      TARG = 10.0**(TARG/20.0)
C
      DO 300 I=0,NUMB
           X(I) = 10.0**(X(I)/20.0)
300    CONTINUE
C
C--------------------Add random phase and make complex--------------------
C
      P = 1.0/8.0
           RFTHR = FTHR*COS(P*TPI)
           IFTHR = FTHR*SIN(P*TPI)
C
           RTARG = TARG*COS(P*TPI)
           ITARG = TARG*SIN(P*TPI)
C
      DO 400 I=0,NUMB
           CALL RANDOM(P,SEED)
           Y(I) = X(I) * SIN(P*TPI)
           X(I) = X(I) * COS(P*TPI)
400    CONTINUE
C
C----------------------------Create Target data----------------------------
C
      I = INT((TGFRQ-DX(2))/DX(3))
C
      X(I) = X(I) + RTARG
      Y(I) = Y(I) + ITARG
C
C-------------------------Create Feedthrough data-------------------------
C
      I = INT((FTFRQ-DX(2))/DX(3))
C
      X(I) = X(I) + RFTHR
```

```
      Y(I) = Y(I) + IFTHR
C
C-----------------------------Formats-----------------------------------
C
10     FORMAT (A)
1000   FORMAT (F15.2,A1,F15.2)
1500   FORMAT (E15.5,E15.5)
3000   FORMAT (E15.5,F15.5)
C
C-----------------------------End program-------------------------------
C

      RETURN
      END
C
C     A Cooley-Tukey Radix-2 dif FFT program complex input data
C     arrays X and Y table look-up of W values.
C
C           C.S. Burrus, Rice University, Sept 1983
C
C--------------------------Variable declarations-------------------------
C
      SUBROUTINE DFFT(X,Y,N,M,FLAG)
C
      IMPLICIT NONE
C
      INTEGER N, M, N1, N2, IE, IA, FLAG
      INTEGER I, J, K, L
C
      REAL X(*), Y(*)
      REAL C, S, XT, YT
      REAL P

C
C---------------------------Define constants-----------------------------
C
      P= 2.0 * ACOS(-1.0)/REAL(N)
C
C-------------------------------Main FFT loops---------------------------
C
      N2 = N
      DO 300 K = 1, M
            N1 = N2
            N2 = N2/2
```

```
            IE = N/N1
            IA = 1
            DO 400 J = 1, N2
                  C = COS(P * REAL(IA-1))
                  S = REAL(FLAG) * SIN(P * REAL(IA-1))
                  IA = IA + IE
                  DO 500 I = J, N, N1
                        L = I + N2
                        XT = X(I) - X(L)
                        X(I) = X(I) + X(L)
                        YT = Y(I) - Y(L)
                        Y(I) = Y(I) + Y(L)
                        X(L) = C*XT + S*YT
                        Y(L) = C*YT - S*XT
500               CONTINUE
400         CONTINUE
300   CONTINUE
C
C-----------------------------Digit reverse counter-----------------------
C
      J = 1
      N1 = N -1
C
      DO 600 I=1,N1
C
            IF (I .GE. J) THEN
                  K = N/2
            ELSE
                  XT = X(J)
                  X(J) = X(I)
                  X(I) = XT
                  YT = Y(J)
                  Y(J) = Y(I)
                  Y(I) = YT
                  K = N/2
            END IF
C
700         IF (K .GE. J) THEN
                  J = J + K
            ELSE
                  J = J - K
                  K = K/2
                  GOTO 700
            END IF
```

```
600    CONTINUE
C
C-------------------------Scale data for FFT-----------------------------
C
      IF(FLAG .EQ. 1)THEN
           DO 800 I=1,N
                X(I) = X(I)/REAL(N)
                Y(I) = Y(I)/REAL(N)
800          CONTINUE
      END IF
C
C----------------------------End Program-------------------------------
C
      RETURN
      END
```

APPENDIX B

Program Listing of Alternate Receiver Model

```
C     Program: RCVR2
C     Author:  Alan Weddle
C     Date:    4/1/96
C     Update:
C
C
C
C-------------------------Variable definitions-------------------------
C
C          All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     ISEED   NONE       RANDOM number generator primer
C
C     X()    V~          Real part of data file
C     Y()    V~          Imaginary part of data file
C     DX()   VARIOUS     Data header
C     A()    V~          Real part of a spectrum
C     B()    V~          Imaginary part of a spectrum
C     DA()   VARIOUS     Data header for the spectrum
C     E()    V~          Real part of a spectrum
C     F()    V~          Imaginary part of a spectrum
C     DE()   VARIOUS     Data header for the spectrum
C     G()    V~          Real part of a spectrum
C     H()    V~          Imaginary part of a spectrum
C     DG()   VARIOUS     Data header for the spectrum
C     NAME   NONE        Name of data file
C     NF     dB          Noise Figure
C     CF     Hz          Filter center frequency
C     BW     Hz          Filter 3 dB bandwidth
C     RBW    Hz          Rejection Bandwidth
C     REJ    Hz          Rejection of rejection bandwidth
C     ULT    dB          Filter ultimate rejection
C     TD     sec         Time delay
C     RERUN  BOOLEAN     Flag to start the pass again new NTHRESH
C     NTHRESH NONE       Scaling factor for noise threshold
C
C        V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-----------------------------------------------------------------------
C-------------------------Variable declarations-------------------------
C
```

```
      PROGRAM RCVR2

      IMPLICIT NONE
C
      REAL X(0:8192), Y(0:8192), DX(4)
      REAL A(0:8192), B(0:8192), DA(4)
      REAL C(0:8192), D(0:8192), DC(4)
      REAL E(0:8192), F(0:8192), DE(4)
      REAL RE(0:8192), IM(0:8192), DR(4)
C
      REAL NFF, NFR
      REAL DELTAF
      REAL TARGET
C
      INTEGER SEEDS(100),TIMES,ISEED
      INTEGER I, J, K, PASS
      INTEGER START

C
      CHARACTER*40 NAME

      NFF = .10
      NFR = .10
      TARGET = 0.
C
C-----INPUT PROGRAM OPTIONS DATA---------------------------------
C
      OPEN(UNIT=1,FILE='CLUTTER\INPUTS\SEEDS',STATUS='OLD')
C
      READ(1,20) TIMES
      DO 50 I=1,TIMES
      READ(1,20) SEEDS(I)
50    CONTINUE
C
      CLOSE(UNIT=1)
C
20    FORMAT(I10)
C
C
C
      DR(1) = 0.0
      DR(2) = 0.0
      DR(3) = 0.0
```

```
    DR(4) = 0.0
C
C-----ZERO THE REAL AND IMAGINARY ARRAYS-------------------------
C
    DO 100 I = 0, 8192
      RE(I) = 0.0
      IM(I) = 0.0
100   CONTINUE
C
    DO 1000 PASS=1,TIMES
      PRINT *, PASS, TIMES
C
C-----ENTER REAR ANTENNA SPECTRUM--------------------------------
C
101   ISEED = SEEDS(PASS)
    NAME = 'CLUTTER\DATA\SPECR'
    CALL REARIN(NAME, DC)
    DELTAF = DC(3)
    CALL DVTO(C, D, DC, DELTAF, ISEED, 2)
    NAME = 'CLUTTER\DATA\DVTO.DAT'
    CALL NPSC(NAME, C, D, DC)
C
C-----CALCULATE SSLO NOISE---------------------------------------
C
    DELTAF = DC(3)
    CALL SN(A, B, DA, E, F, DE, DELTAF, ISEED, 2)
    NAME = 'CLUTTER\DATA\ISSLO.DAT'
    CALL NPSC(NAME, A, B, DA)
C
C-----ENTER ANTENNA DATA-----------------------------------------
C
      NAME = 'CLUTTER\DATA\SPECF'
      CALL DATIN(NAME, X, Y, DX, ISEED)
C
C-----ADD NOISE TO FRONT SPECTRUM--------------------------------
C
      CALL NOISE(DX, NFF)
      CALL REDUCE(X, Y, DX)
      NAME = 'CLUTTER\DATA\ISUM.DAT'
      CALL NPSC(NAME, X, Y, DX)
C
C-----CONVOLVE THE SSLO WITH THE FRONT SPECTRUM-------------------
C
      CALL CONVL(X, Y, DX, E, F, DE)
```

```
      NAME = 'CLUTTER\DATA\IFRNTSS.DAT'
      CALL NPSC(NAME, X, Y, DX)
C
C-----CONVOLVE FRONT AND DVTO SPECTRA----------------------------
C
      CALL CONVL(X, Y, DX, C, D, DC)
      NAME = 'CLUTTER\DATA\ICONV.DAT'
      CALL NPSC(NAME, X, Y, DX)
C
C-----CALCULATE TEN PASS AVERAGE---------------------------------
C
    IF(PASS .EQ. 1)THEN
      DR(1) = DX(1)
      DR(2) = DX(2)
      DR(3) = DX(3)
      DR(4) = DX(4)
    END IF
    IF(DX(2).GT.DR(2))THEN
      J = INT((DX(2) - DR(2))/DX(3))
      DR(2) = DX(2)
      K = INT(DR(1))
      DR(1) = DR(1) - REAL(J)
      DO 110 I=J,K
          RE(I-J) = RE(I)
          IM(I-J) = IM(I)
110     CONTINUE
    ELSE
      J = INT((DR(2) - DX(2))/DX(3))
      DX(2) = DR(2)
      K = INT(DX(1))
      DX(1) = DX(1) - REAL(J)
      DO 120 I=J,K
          X(I-J) = X(I)
          Y(I-J) = Y(I)
120     CONTINUE
    END IF
C
    IF(DX(1).GT.DR(1))THEN
      DX(1) = DR(1)
    ELSE
      DR(1) = DX(1)
    END IF
C
    K = REAL(DR(1))
```

```
      DO 130 I=0,K
        RE(I) = RE(I) + X(I)**2 + Y(I)**2
130   CONTINUE
C
1000  CONTINUE

      IF (DR(1) .GT. 5000) THEN
        IF (DR(2) .LT. 0.) THEN
          START = INT( TARGET - DR(2)/DR(3) )
          START = AMAX0(START, 0)
        ELSE
          START = 0
        END IF
        DO 133, I = START, DR(1)
          RE(I-START) = RE(I)
          IM(I-START) = IM(I)
133       CONTINUE
        DR(1) = DR(1) - START
        K = RE(DR(1))
      END IF

      DO 140 I=0,K
        RE(I) = SQRT(RE(I)/REAL(TIMES))
        IM(I) = 0.0D0
140   CONTINUE
C
      NAME = 'CLUTTER\DATA\AVE.DAT'
      CALL NPSC(NAME,RE,IM,DR)
C
C-----End program--------------------------------------------------
C
      STOP

      END

C
C*****************************SUBROUTINE
LIMIT*******************************
C
       SUBROUTINE LIMIT(X,Y,DX,A,B,DA)
C
C      This program determines the output of a limiter given an input
C      spectrum.
C
```

```
C--------------------------Variable definitions--------------------------
C
C          All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE        Integer counter
C     J      NONE        Integer counter
C     M      NONE        Integer counter
C     K      NONE        Integer counter
C     DX(1)  NONE        Total number resolution bandwidths
C     NUMB1  NONE        Total number of input data points created
C     NUMB2  NONE        Total number of output data points
C     NUM    NONE        2**M data space used in processing
C     NUM1   NONE        2**M processing number minus 1
C     DX(2)  Hz          Starting frequency
C     DX(3)  Hz          Step frequency or resolution bandwidth
C     DX(4)  V~          Input noise floor
C     X()    V~          Real part of the data
C     Y()    V~          Imaginary part of the data
C     THRESH V~          Limiter threshold level
C     THETA  RADIANS     Phase of the input signal
C     NRATIO V~/V~       Peak to noise ratio
C
C       V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------------
C--------------------------Variable declarations--------------------------
C
      IMPLICIT NONE
C
      INTEGER I, J, M, NUM, NUM1, K
      INTEGER NUMB1
C
      REAL T1, T2, MAG
      REAL X(0:*), Y(0:*), DX(*), POUT, PIN
      REAL A(0:*), B(0:*), DA(*)
C
C----------------------------Define constants-----------------------------
C
      NUMB1 = INT(DX(1))
      M = 13
      NUM = 2**M
```

```
      NUM1 = NUM-1
C
C---------------------------Prepare input data--------------------------
C
      PIN = 0.0
      T2  = 0.0
      K   = 0
      DO 150 I=0,NUMB1
           T1 = X(I)**2 + Y(I)**2
           PIN = PIN + T1
           IF(T1.GT.T2) THEN
                K = I
                T2 = T1
           END IF
150   CONTINUE
C
      J = NUM/2 - K
      DO 200 I=J,J+NUMB1
           A(I) = X(I-J)
           B(I) = Y(I-J)
200   CONTINUE
C
      DO 250 I=0,J-1
           A(I) = 0.0
           B(I) = 0.0
250   CONTINUE
C
      J = J+NUMB1+1
      DO 300 I=J, NUM1
           A(I) = 0.0
           B(I) = 0.0
300   CONTINUE
C
      DO 350 I=0,NUM1
          X(I) = A(I)
          Y(I) = B(I)
350   CONTINUE
C
C-------------------------Calculate time domain input--------------------
C
      CALL DFFT(X,Y,NUM,M,-1)
C
C-------------------------Calculate time domain output-------------------
C
```

```
      DO 500 I=0,NUM1
            MAG =  SQRT(X(I)**2 + Y(I)**2)
            X(I) = X(I)/MAG
            Y(I) = Y(I)/MAG
500    CONTINUE
C
C------------------Convert to frequency and truncate data------------------
C
      CALL DFFT(X,Y,NUM,M,1)
C
      POUT = 0.0
      T2 = 0.0
      DO 600 I=0,NUM1
         T1 = X(I)**2 + Y(I)**2
         POUT = POUT + T1
         T2 = AMAX1(T1,T2)
600    CONTINUE
C
      T2 = SQRT(T2)
      DO 700 I=0,NUM1
         X(I) = X(I)/T2
         Y(I) = Y(I)/T2
700    CONTINUE
C
      DX(4) = DX(4) * SQRT(POUT/PIN) / T2
C
C-----------------------Generate output frequency data---------------------
C
      DX(1) =  REAL(NUM1)
      DX(2) =  DX(2) + DX(3) * REAL(K - NUM/2)
C
C
C
      T1 = DX(3)*DX(1) - 376000.0
C
      IF (T1 .GT. 0.0) THEN
         T1 = T1/DX(3)/2.0
         J = NINT(T1)+1
         DO 111 I=J,NUM1-J
            X(I-J) = X(I)
            Y(I-J) = Y(I)
111        CONTINUE
         DX(1) = REAL(NUM1-2*J)
         DX(2) = DX(2) + DX(3) * REAL(J)
```

```
      END IF
C
C-----------------Reduce limitation to significant data only----------
C
      CALL REDUCE(X,Y,DX)
C
C-------------------------------End program-----------------------------
C
      RETURN
      END
     SUBROUTINE LPF(F,MAG,PHI)
C
     IMPLICIT NONE
C
C
C
     REAL GR, GI, DR, DI, NR, NI, F, PHI, TPI, MAG
     REAL S1, S2, S3, S4
C
C
C
     TPI = 2.0 * ACOS(-1.0)
     PHI = TPI * F
     S1 = PHI
     S2 = PHI**2
     S3 = PHI**3
     S4 = PHI**4
C
C
C
     NR = 1.0
     NI = 2.08E-3 * S1
C
     DR=1.0D0 - 2.16E-6*S2 + 4.01E-13*S4
     DI=2.08E-3*S1 - 1.32E-9*S3
C
     MAG = SQRT(DR**2 + DI**2)
     PHI = ATAN2(DI,DR)
C
     MAG = 1.0D0/MAG
     PHI = -PHI
     DR = MAG * COS(PHI)
     DI = MAG * SIN(PHI)
C
```

```
      GR = NR * DR - NI * DI
      GI = NR * DI + NI * DR
C
      MAG = SQRT(GR**2 + GI**2)
      PHI = ATAN2(GI,GR)
C
C
C
      RETURN
      END
C
C**************************SUBROUTINE
NOISE*******************************
C
      SUBROUTINE NOISE(DX,NF)
C
C     This program adds noise to a complex unit spectrum.
C
C-------------------------Variable definitions----------------------------
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE     Integer counter
C     DX(1)  NONE     Number of resolution bandwidths
C     DX(2)  Hz       Starting frequency of spectrum
C     DX(3)  Hz       Step frequency or resoution bandwidth
C     DX(4)  V~       Noise floor
C     K      J/K      Boltsman's constant
C     TE     K        Effective temperature
C     NF     dB       Noise figure
C     NOIS   Watts    Noise power level
C
C        V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-----------------------------------------------------------------------
C-------------------------Variable declarations---------------------------
C
      IMPLICIT NONE
C
      REAL DX(*), K, TE, NF
      REAL NOIS
C
C-------------------------Calculate and add noise-------------------------
C
```

```
      NF = 10.0**(NF / 10.0)
C
      K = 1.38E-23
C
      TE = (NF - 1.0) * 290.0
C
      NOIS = DX(3) * K * TE * 1000.0
C
      DX(4) = SQRT(DX(4)**2 + NOIS)
C
C--------------------------------End Program-----------------------------
C
      RETURN
      END
     SUBROUTINE NPSC(NAME,X,Y,DX)
C
C     Program: Power Spectrum Converter (PSC)
C
C     This program converts an amplitude spectrum into
C     a power spectrum.
C
C---------------------------Variable definitions-------------------------
C
C    -UNITS-  -NAME-              -DESCRIPTION-
C
C     I,J,K   NONE        Integer counters
C     NUMB    NONE        Number of data points
C     X()     V~          Real part of the input spectrum
C     Y()     V~          Imaginary part of input spectrum
C     DX(1)   NONE        Total number of data points
C     DX(2)   Hz          Start frequency
C     DX(3)   Hz          Resolution bandwidth
C     DX(4)   V~          Input noise
C     FX      Hz          Frequency
C     INPUT   NONE        Input spectrum name
C
C        V~ = VOLTS / SQUARE ROOTS OF OHMS
C
C-----------------------------------------------------------------------
C-------------------------Variable declarations-------------------------
C
      IMPLICIT NONE
C
      INTEGER NUMB, I
```

```
C
      REAL X(0:*), Y(0:*), DX(*), FX, T1, T2
C
      CHARACTER*40 NAME
C
      NUMB = INT(DX(1))
C
      NUMB = MIN0(NUMB, 8188)
C
C
      OPEN (UNIT=1,FILE=NAME,STATUS='UNKNOWN')
      REWIND(1)
C
      WRITE(1,30) DX(1)
      WRITE(1,30) DX(2)
      WRITE(1,30) DX(3)
C
C------------------------Calculate power spectrum--------------------------
C
      IF (DX(4) .LE. 1.0E-15) THEN
           T1 = 20.0 * ALOG10(1.0E-15)
      ELSE
           T1 = 20.0 * ALOG10(DX(4))
      END IF
      WRITE(1,30) T1
C
C
C
      DO 200 I = 0,NUMB
           T2 = X(I)**2 + Y(I)**2
C
           IF(T2 .LT. 1.0E-16) THEN
                T2 = 1.0E-16
           END IF
C
           T2 = 10.0 * ALOG10(T2)
C
           FX = DX(2) + DX(3) * REAL(I)
           WRITE(1,50) FX, T2
200    CONTINUE
C
      ENDFILE(1)
      CLOSE (UNIT = 1)
C
```

```
C-------------------------------Formats--------------------------------
C
30    FORMAT(1X,E12.5)
50    FORMAT(1X,E12.5,1H,,E12.5)
C
C------------------------------End program-----------------------------
C
      RETURN
      END
C
C*****************************SUBROUTINE
RANDOM*****************************
C
      SUBROUTINE RANDOM (X,SEED)
C
C     This program was taken from 'THE ART OF COMPUTER PROGRAMING'
C     Vol. 2, D. Knuth, Addison-Wesley
C
C     This program generates random numbers based on the input Seed
C
C-------------------------Variable definitions-------------------------
C
C     X      Random number
C     SEED   Seed
C     MULT   Seed multiplier
C     BASE   Normalization base
C
C-------------------------Variable declarations------------------------
C
      IMPLICIT NONE
C
      REAL X
C
      INTEGER SEED
C
C-------------------------Generate random number-----------------------
C
      SEED = MOD(25211 * SEED,32768)
      X = REAL(SEED) / 32768.
C
C------------------------------End program-----------------------------
C
      RETURN
      END
```

```
C
C*****************************SUBROUTINE
REDUCE*****************************
C
      SUBROUTINE REDUCE(X,Y,DX)
C
C     This program reduces a data file to data significantly above
C     the input noise floor.
C
C----------------------------Variable definitions-------------------------
C
C           All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE        Integer counter
C     J      NONE        Integer counter
C     NUMB   NONE          Total number of data points
C     DX(1)  NONE         Total number of frequency steps
C     DX(2)  Hz           Start frequency
C     DX(3)  Hz           Step frequency or resolution bandwidth
C     DX(4)  V~           Input noise floor
C     NOISE  WATTS         Input noise floor
C     SIGNAL WATTS         Signal power of interest
C     X()    V~           Real part of the data
C     Y()    V~           Imaginary part of the data
C
C        V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-----------------------------------------------------------------------
C----------------------------Variable declarations------------------------
C
      IMPLICIT NONE
C
      INTEGER I, J, K, NUMB
C
      REAL DX(*), X(0:*), Y(0:*)
      REAL NOISE, SIGNAL
C
      NUMB = INT(DX(1))
C
C--------------------------------Reduce Data-------------------------------
C
```

```
      NOISE = DX(4)**2
C
      DO 200 I=0,NUMB
C
            SIGNAL = X(I)**2 +Y(I)**2
C
C----------SIGNAL MUST BE HIGHER THAN THE AVERAGE NOISE

            IF (SIGNAL .GT. NOISE) THEN
                  IF (I .GT. 19) THEN
                        K = I - 20
                  ELSE
                        K = 0
                  END IF
                  DX(2) = DX(2)+FLOAT(K)*DX(3)
                  DO 400 J=K,NUMB
                        X(J-K) = X(J)
                        Y(J-K) = Y(J)
400                 CONTINUE
                  NUMB = NUMB - K
                  GO TO 3000
            END IF
200      CONTINUE
C
3000     CONTINUE
C
      DO 500 I=0,NUMB
C
            SIGNAL = X(NUMB-I)**2 + Y(NUMB-I)**2
C
            IF (SIGNAL .GT. NOISE) THEN
                  IF (I .GT. 19) THEN
                        K = I - 20
                  ELSE
                        K = 0
                  END IF
                  NUMB = NUMB - K
                  GO TO 2000
            END IF
500      CONTINUE
C
2000     CONTINUE
C
      DX(1) = NUMB
```

```
C
C-----------------------------End program--------------------------------
C
      RETURN
      END


      SUBROUTINE SN(X ,Y, DX, A, B, DA, RB, SEED, OPTION)
C
      IMPLICIT NONE
C
      INTEGER I, M, NUM, NUM1, HNUM1, SEED, OPTION
C
      REAL R, RB, F
      REAL SCALE, TAU, LASTX, LASTY, SAMPRATE
C
      REAL X(0:*), Y(0:*), NOISE, A(0:*), B(0:*)
      REAL TPI, T, DX(*), DA(*)
C
C-----Set constants-------------------------------------------------
C
      TAU = 50.E-9

      M = NINT(ALOG10(220000.0/REAL(RB))/ALOG10(2.0))
      M = M + 1
      NUM = 2**(M)
      NUM1 = NUM - 1
      HNUM1 = NUM1/2
C
      SAMPRATE = 1. / 220.E3
C
      DX(1) = REAL(NUM1)
      DX(2) = REAL(-NUM/2) * RB
      DX(3) = RB
C
C-----COPY SPECTRUM 1 PARAMETERS TO SPECTRUM 2 PARAMETERS
C
      DA(1) = DX(1)
      DA(2) = DX(2)
      DA(3) = DX(3)
      DA(4) = DX(4)
C
      TPI = 2.0 * ACOS(-1.0)
C
```

```
C-----Create the reference oscillator signal----------------------
C
    DO 100 I=0,NUM1
      X(I) = 0.0
      Y(I) = 0.0
100   CONTINUE
C
C
    IF (OPTION .EQ. 1) THEN
C
      SCALE = SQRT( RB / 1000. )

      DX(4) = 10.**((-9. + ALOG10(RB))/2.0)
      DA(4) = DX(4)
C
      DO 200 I=1,HNUM1
        F= RB * REAL(I)
        IF (F .LT. 1000.0) THEN
          Y(I) = REAL(0.17782794) * SCALE
        ELSE IF (F .GE. 1000. .AND. F .LT. 10000.) THEN
          Y(I) = REAL(0.17782794) * SCALE /
     .          ( F/1000. )
        ELSE IF (F .GE. 10000 .AND. F .LT. 1.2E5) THEN
          Y(I) = 3.758374E-3 * SCALE /
     .          ( F/10000. )
        ELSE IF (F .GT. 1.2E5 .AND. F .LT. 2.6E5) THEN
          Y(I) = 10.**(( -9. + ALOG10(RB) ) / 2.) * SCALE
        END IF
200     CONTINUE
C
    ELSE
      DX(4) = 10.**((-12.2 + ALOG10(RB))/2.0)
      DO 250 I=0,HNUM1
        F = RB * REAL(I)
        Y(I) = -85. - 49.8 * ALOG10(F/1.0E3 + 1.0E-16)
        Y(I) = Y(I) + 10. * ALOG10(RB)
        Y(I) = 10.**(Y(I)/20.)
250     CONTINUE
        Y(0) = 0.0
C
    END IF
C
C-----ADD PHASE NOISE AND MAKE SURE THE TIME DOMAIN WILL BE REAL
(THE
```

```
C       UPPER HALF OF THE FFT IS THE COMPLEX CONJUGATE OF THE
LOWER.)
C
    DO 500 I=1,HNUM1
      CALL RANDOM(R,SEED)
      NOISE = Y(I)
      X(I) = NOISE * SIN(TPI * R)
      Y(I) = NOISE * COS(TPI * R)
      X(NUM-I) =  X(I)
      Y(NUM-I) = -Y(I)
500   CONTINUE
C
    CALL DFFT(X,Y,NUM,M,-1)
C
C-----Phase modulate frequency centered source----------------------
C
    DO 700 I=0,NUM1
      T = X(I) + REAL(I)*TPI/2.0
      X(I) = COS(T)
      Y(I) = SIN(T)
C
C--------MAKE THE SECOND SSLO SLIGHTLY TIME SHIFTED FROM THE FIRST
BY
C         USING LINEAR INTERPOLATION BETWEEN TWO POINTS.
C
      IF (I .GT. 1) THEN
        A(I-1) = LASTX + TAU * (X(I)-LASTX) / SAMPRATE
        B(I-1) = LASTY + TAU * (Y(I)-LASTY) / SAMPRATE
      END IF
      LASTX = X(I)
      LASTY = Y(I)

700   CONTINUE
C
    A(0) = 0.
    B(0) = 0.
    A(NUM1) = 0.
    B(NUM1) = 0.
C
C-----Convert to frequency domain----------------------------------
C

    CALL DFFT(X,Y,NUM,M,1)
    CALL REDUCE(X,Y,DX)
```

```
      CALL DFFT(A,B,NUM,M,1)
      CALL REDUCE(A,B,DA)
C
C-----End Program------------------------------------------------
C
      RETURN
      END
C
C*************************SUBROUTINE
CONVOLVE*****************************
C
      SUBROUTINE CONVL(X,Y,DX,A,B,DA)
C
C     This program convolves two spectra producing down mix product
C
C-------------------------Variable definitions-------------------------
C
C          All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-            -DESCRIPTION-
C
C     I      NONE      Integer counter
C     J      NONE      Integer counter
C     K      NONE      Integer counter
C     DX(1)  NONE      Number of frequency steps in the X spectrum
C     NUMX   NONE      Total number of data points in the X spectrum
C     DX(2)  Hz        Start frequency of the X spectrum
C     DX(3)  Hz        Resolution bandwidth of the X spectrum
C     DX(4)  V~        Noise floor of the X spectrum
C     X()    V~        Real data for the X spectrum
C     Y()    V~        Imaginary data for the X spectrum
C     DA(1)  NONE      Number of frequency steps in the A spectrum
C     NUMA   NONE      Total number of data points in the A spectrum
C     DA(2)  NONE      Start frequency of the A spectrum
C     DA(3)  Hz        Resolution bandwidth of the A spectrum
C     DA(4)  V~        Noise floor of the A spectrum
C     A()    V~        Real data for the A spectrum
C     B()    V~        Imaginary data for the A spectrum
C     NUMo   NONE      Number of data points in the output spectrum
C     M      NONE      FFT radix exponent
C     NUM    NONE      FFT array length
C     NUM1   NONE      FFT array length minus one
C     W,Z    NONE      Temporary data storage
```

```
C     NI()   V~        Imaginary part of the noise calcualtion
C     NR()   V~        Real part of the noise calculation
C     NOISEX  dB       Noise power caused by the X spectrum
C     NOISEA  dB       Noise power caused by the A spectrum
C     TPI    RADIANS   2.0 * 3.14159.....
C     P      NONE      Random number
C
C       V~ = VOLTS/SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------------
C----------------------------Variable declarations-------------------------
C
      IMPLICIT NONE
C
      INTEGER NUMX, NUMA, NUMo, NUM, NUM1,M
      INTEGER I, J
C
      REAL TPI, NOISEX, NOISEA
      REAL X(0:*), Y(0:*), DX(*)
      REAL A(0:*), B(0:*), DA(*)
      REAL W, Z
C
C-----------------------------Define constants-----------------------------
C
      TPI = 2.0 * ACOS(-1.0)
C
      NUMX = INT(DX(1))
      NUMA = INT(DA(1))
C
C-----------------------Determine data compatability----------------------
C
      IF (DX(3) .NE. DA(3)) THEN
          PRINT *,'DX(3)=',DX(3),'DA(3)=',DA(3)
          PRINT *, 'FREQUENCY STEP NOT COMPATABLE'
          PAUSE
          GO TO 3000
C
C                 *line 3000 is at the end of the program*
C
      END IF
C
C----------------------------Flip 'A' spectrum----------------------------
C
      J = INT(REAL(NUMA)/2.0)
```

```
      DO 200 I=0,J
           W = A(I)
           A(I) = A(NUMA-I)
           A(NUMA-I) = W
C
           W = B(I)
           B(I) = B(NUMA-I)
           B(NUMA-I) = W
200    CONTINUE
C
      DO 250 I=0,NUMA
           B(I) = -B(I)
250    CONTINUE
C
C-----------------------Calculate Spectrum 'X' Noise----------------------
C
           NOISEX = 0.0
           W = DA(4)**2
           DO 400 I=0,NUMX
             NOISEX = W * (X(I)**2 + Y(I)**2) + NOISEX
400          CONTINUE
C
C-----------------------Calculate Spectrum 'A' Noise----------------------
C
           NOISEA = 0.0
           W = DX(4)**2
           DO 700 I=0,NUMA
             NOISEA = W * (A(I)**2 + B(I)**2) + NOISEA
700          CONTINUE
C
C----------------------------Add spectrum noise---------------------------
C
      IF ( (NOISEA + NOISEX) .GT. 1.E-38) THEN
        DX(4) = SQRT(NOISEA + NOISEX)
      ELSE
        DX(4) = 1.E-19
      END IF
C
C-------------------------Calculate 2**M divisions-----------------------
C
      NUMo =INT(DX(1) + DA(1))
C
      M = NINT(ALOG(FLOAT(NUMo))/ALOG(2.0))
      NUM = 2.0**M
```

```
      NUM1 = NUM - 1
C
      IF (NUM.LT.NUMo) THEN
           M = M + 1
           NUM = 2.0**M
           NUM1 = NUM - 1
      END IF
C
C-------------------------Add zeros to spectrums-------------------------
C
      DO 900 I = NUMX+1,NUM1
           X(I) = 0.0
           Y(I) = 0.0
900    CONTINUE
C
      DO 1000 I = NUMA+1,NUM1
           A(I) = 0.0
           B(I) = 0.0
1000   CONTINUE
C
C-------------------------Calculate Convolution-------------------------
C
      CALL DFFT(X,Y,NUM,M,-1)
      CALL DFFT(A,B,NUM,M,-1)
C
      DO 1100 I=0,NUM1
           W = X(I) * A(I) - Y(I) * B(I)
           Z = X(I) * B(I) + Y(I) * A(I)
           X(I) = W
           Y(I) = Z
1100   CONTINUE
C
      CALL DFFT(X,Y,NUM,M,1)
C
C-------------------------Generate frequencies-------------------------
C
      DX(2) = DX(2) - (DA(2) + DA(3) * DA(1))
      DX(1) = NUMo
C
C---------------Reduce convolution to significant data only---------------
C
      CALL REDUCE(X,Y,DX)
C
C-----------------------------End program-----------------------------
```

```
C
3000   CONTINUE
C
      RETURN
      END
C
C**************************SUBROUTINE
INPUT*****************************
C
      SUBROUTINE DATIN(NAME,X,Y,DX,SEED)
C
C     This program converts the input signal from dBm to
C     Volts / square root of Ohms, adds a random phase shift to
C     each frequency bin, and includes the affects of illuminator
C     noise sidebands.
C
C---------------------------Variable definition--------------------------
C
C           All variables used in this program are defined by type.
C     Note the use of IMPLICIT NONE.
C
C    -NAME-  -UNITS-          -DESCRIPTION-
C
C     I      NONE        Integer Counter
C     J      NONE        Integer counter
C     DX(1)  NONE          Number of spectrum data points
C     DX(3)  Hz          Resolution bandwidth
C     DX(2)  Hz          Starting frequency of spectrum
C     DX(4)  V~          Noise floor
C     X()    dBm, V~       Spectrum power, Real part
C     Y()    V~         Imaginary part of the spectrum
C     NUMB   NONE          Integer counter based on DX(1)
C     STOP   Hz          Stoping frequency
C     TDFT   sec         Feedthrough time delay
C     FTFRQ  sec          Feedthrough frequency
C     FTHR   dBm           Feedthrough power
C     RFTHR  V~            Real feedthrough
C     IFTHR  V~           Imaginary feedthrough
C     TDTG   sec         Target time delay
C     TGFRQ  Hz           Target frequency
C     TARG   dBm           Target power
C     RTARG  V~            Real target
C     ITARG  V~           Imaginary target
C     FREQ   Hz          Place holding frequency in the input array
```

```
C     NOISE  dBm        Input noise floor
C     DA(1)  NONE       Number of data points in the spectrum
C     DA(3)  Hz         Step Frequency of spectrum
C     DA(2)  Hz         Starting frequency of spectrum
C     DA(4)  V~         Noise floor
C     A()    V~         Real part of the spectrum
C     B()    V~         Imaginary part of the spectrum
C     NAME   NONE       Name of the power spectrum file
C     SEED   NONE       RANDOM number generator primer
C     P      NONE       Random variable for determining phase
C     TPI    NONE       2.0 * 3.14159.....
C
C       V~ = VOLTS / SQUARE ROOT OF OHMS
C
C-------------------------------------------------------------------
C-------------------------Variable declarations-----------------------
C
      IMPLICIT NONE
C
      INTEGER I, J, NUMB, SEED
C
      REAL NUMB1, START, DF, STOP
      REAL FTHR, FTFRQ, TGFRQ, TARG
      REAL FREQ, NOISE, TDFT, TDTG, TDCL
      REAL RFTHR, IFTHR, RTARG, ITARG
      REAL P
      REAL TPI
C
      REAL X(0:*), Y(0:*), DX(*), FX
      CHARACTER test
C
      CHARACTER*40 NAME
C
C--------------------------Define constants---------------------------
C
      TPI = 2.0 * ACOS(-1.0)
C
C-------------------------Open input data file-------------------------
C
      OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
C
C-------------------------Read data format-----------------------------
C
      READ(1,1000) NUMB1,test,START
```

```
      READ(1,1000) DF,test,STOP
      READ(1,1000) FTFRQ,test,FTHR
      READ(1,1000) TGFRQ,test,TARG
      READ(1,1500) TDFT,TDTG
      READ(1,3000) TDCL,NOISE
      READ(1,1000) FREQ,test,NOISE
      NUMB = INT(NUMB1)

C
C--------------------Increase spectrum width for target--------------------
C
      IF(TGFRQ .GT. STOP)THEN
          J = INT((TGFRQ - START)/DF)
          NUMB1 = REAL(J)
      END IF
C
      DO 100 I=0,J
          X(I) = 0.0
          Y(I) = 0.0
100    CONTINUE
C
C---------------------------Read spectrum data---------------------------
C
      DO 200 I=0,NUMB
           READ(1,1000) FX,test,X(I)
200    CONTINUE
C
      CLOSE (UNIT = 1)
C
      DX(1) = NUMB1
      DX(2) = START
      DX(3) = DF
      DX(4) = 0.0
C
C
C----------------------Convert to volts/sqrt(ohms)----------------------
C
      FTHR = 10.0**(FTHR/20.0)
C
      TARG = 10.0**(TARG/20.0)
C
      DO 300 I=0,NUMB
           X(I) = 10.0**(X(I)/20.0)
300    CONTINUE
```

```
C
C--------------------Add random phase and make complex----------------------
C
      P = 1.0/8.0
            RFTHR = FTHR*COS(P*TPI)
            IFTHR = FTHR*SIN(P*TPI)
C
            RTARG = TARG*COS(P*TPI)
            ITARG = TARG*SIN(P*TPI)
C
      DO 400 I=0,NUMB
            CALL RANDOM(P,SEED)
            Y(I) = X(I) * SIN(P*TPI)
            X(I) = X(I) * COS(P*TPI)
400     CONTINUE
C
C---------------------------Create Target data---------------------------
C
      I = INT((TGFRQ-DX(2))/DX(3))
C
      X(I) = X(I) + RTARG
      Y(I) = Y(I) + ITARG
C
C-------------------------Create Feedthrough data-------------------------
C
      I = INT((FTFRQ-DX(2))/DX(3))
C
      X(I) = X(I) + RFTHR
      Y(I) = Y(I) + IFTHR
C
C------------------------------Formats----------------------------------
C
10     FORMAT (A)
1000   FORMAT (F15.2,A1,F15.2)
1500   FORMAT (E15.5,E15.5)
3000   FORMAT (E15.5,F15.5)
C
C---------------------------End program-------------------------------
C

      RETURN
      END
C
C     A Cooley-Tukey Radix-2 dif FFT program complex input data
```

```
C     arrays X and Y table look-up of W values.
C
C          C.S. Burrus, Rice University, Sept 1983
C
C---------------------------Variable declarations---------------------------
C
      SUBROUTINE DFFT(X,Y,N,M,FLAG)
C
      IMPLICIT NONE
C
      INTEGER N, M, N1, N2, IE, IA, FLAG
      INTEGER I, J, K, L
C
      REAL X(*), Y(*)
      REAL C, S, XT, YT
      REAL P

C
C----------------------------Define constants-----------------------------
C
      P= 2.0 * ACOS(-1.0)/REAL(N)
C
C-----------------------------Main FFT loops------------------------------
C
      N2 = N
      DO 300 K = 1, M
           N1 = N2
           N2 = N2/2
           IE = N/N1
           IA = 1
           DO 400 J = 1, N2
                C = COS(P * REAL(IA-1))
                S = REAL(FLAG) * SIN(P * REAL(IA-1))
                IA = IA + IE
                DO 500 I = J, N, N1
                     L = I + N2
                     XT = X(I) - X(L)
                     X(I) = X(I) + X(L)
                     YT = Y(I) - Y(L)
                     Y(I) = Y(I) + Y(L)
                     X(L) = C*XT + S*YT
                     Y(L) = C*YT - S*XT
500             CONTINUE
400        CONTINUE
```

```
300    CONTINUE
C
C----------Digit reverse counter----------
C
      J = 1
      N1 = N -1
C
      DO 600 I=1,N1
C
            IF (I .GE. J) THEN
                  K = N/2
            ELSE
                  XT = X(J)
                  X(J) = X(I)
                  X(I) = XT
                  YT = Y(J)
                  Y(J) = Y(I)
                  Y(I) = YT
                  K = N/2
            END IF
C
700           IF (K .GE. J) THEN
                  J = J + K
            ELSE
                  J = J - K
                  K = K/2
                  GOTO 700
            END IF
600    CONTINUE
C
C----------Scale data for FFT----------
C
      IF(FLAG .EQ. 1)THEN
            DO 800 I=1,N
                  X(I) = X(I)/REAL(N)
                  Y(I) = Y(I)/REAL(N)
800           CONTINUE
      END IF
C
C----------End Program----------
C
      RETURN
      END
C
```

```
C*****************************SUBROUTINE
REARIN*******************************
C
      SUBROUTINE REARIN(NAME,DX)
C
C     This program reads the rear input power spectrum received for the
C     Point in Flight clutter model.  This routine reads the frequecy of
C     the illuminator doppler and returns that value back to the main
C     program.
C
C-------------------------------------------------------------------------
C--------------------------Variable declarations--------------------------
C
      IMPLICIT NONE
C
      INTEGER NUMB
C
      REAL NUMB1, START, DF, STOP
      REAL FTHR, FTFRQ, TGFRQ, TARG
      REAL FREQ, NOISE, TDFT, TDTG, TDCL
      REAL TPI
C
      REAL DX(*)
      CHARACTER test
C
      CHARACTER*40 NAME
C
C---------------------------Define constants-----------------------------
C
      TPI = 2.0 * ACOS(-1.0)
C
C-------------------------Open input data file---------------------------
C
      OPEN(UNIT=1,FILE=NAME,STATUS='OLD')
C
C---------------------------Read data format-----------------------------
C
      READ(1,1000) NUMB1,test,START
      READ(1,1000) DF,test,STOP
      READ(1,1000) FTFRQ,test,FTHR
      READ(1,1000) TGFRQ,test,TARG
      READ(1,1500) TDFT,TDTG
      READ(1,3000) TDCL,NOISE
      READ(1,1000) FREQ,test,NOISE
```

```
      NUMB = INT(NUMB1)
      DX(1) = NUMB1
      DX(2) = FTFRQ
      DX(3) = DF
      DX(4) = 0.0
C
C-------------------------------Formats-----------------------------------
C
10     FORMAT (A)
1000   FORMAT (F15.2,A1,F15.2)
1500   FORMAT (E15.5,E15.5)
3000   FORMAT (E15.5,F15.5)
C
C-----------------------------End program---------------------------------
C

      RETURN
      END
C
C
C
     SUBROUTINE DVTO(X ,Y, DX, RB, SEED, OPTION)
C
     IMPLICIT NONE
C
     INTEGER I, M, NUM, NUM1, HNUM1, SEED, OPTION
C
     REAL R, RB, F
     REAL SCALE, TAU, SAMPRATE
C
     REAL X(0:*), Y(0:*), NOISE
     REAL TPI, T, DX(*)
C
C-----Set constants-------------------------------------------------
C
     TAU = 50.E-9

     M = NINT(ALOG10(220000.0/REAL(RB))/ALOG10(2.0))
     M = M + 1
     NUM = 2**(M)
     NUM1 = NUM - 1
     HNUM1 = NUM1/2
C
     SAMPRATE = 1. / 220.E3
```

```
C
      DX(1) = REAL(NUM1)
      DX(2) = REAL(-NUM/2) * RB + DX(2)

      TPI = 2.0 * ACOS(-1.0)
C
C-----Create the reference oscillator signal----------------------
C
      DO 100 I=0,NUM1
        X(I) = 0.0
        Y(I) = 0.0
100   CONTINUE
C
C
      IF (OPTION .EQ. 1) THEN
C
        SCALE = SQRT( RB / 1000. )

        DX(4) = 10.**((-9. + ALOG10(RB))/2.0)
        DO 200 I=1,HNUM1
          F= RB * REAL(I)
          IF (F .LT. 1000.0) THEN
            Y(I) = REAL(0.17782794) * SCALE
          ELSE IF (F .GE. 1000. .AND. F .LT. 10000.) THEN
            Y(I) = REAL(0.17782794) * SCALE /
     .           ( F/1000. )
          ELSE IF (F .GE. 10000 .AND. F .LT. 1.2E5) THEN
            Y(I) = 3.758374E-3 * SCALE /
     .           ( F/10000. )
          ELSE IF (F .GT. 1.2E5 .AND. F .LT. 2.6E5) THEN
            Y(I) = 10.**(( -9. + ALOG10(RB) ) / 2.) * SCALE
          END IF
200     CONTINUE
C
      ELSE
        DX(4) = 10.**((-12.2 + ALOG10(RB))/2.0)
        DO 250 I=0,HNUM1
          F = RB * REAL(I)
          Y(I) = -85. - 49.8 * ALOG10(F/1.0E3 + 1.0E-16)
          Y(I) = Y(I) + 10. * ALOG10(RB)
          Y(I) = 10.**(Y(I)/20.)
250     CONTINUE
          Y(0) = 0.0
C
```

```
    END IF
C
C-----ADD PHASE NOISE AND MAKE SURE THE TIME DOMAIN WILL BE REAL
(THE
C       UPPER HALF OF THE FFT IS THE COMPLEX CONJUGATE OF THE
LOWER.)
C
    DO 500 I=1,HNUM1
      CALL RANDOM(R,SEED)
      NOISE = Y(I)
      X(I) = NOISE * SIN(TPI * R)
      Y(I) = NOISE * COS(TPI * R)
      X(NUM-I) =  X(I)
      Y(NUM-I) = -Y(I)
500   CONTINUE
C
    CALL DFFT(X,Y,NUM,M,-1)
C
C-----Phase modulate frequency centered source----------------------
C
    DO 700 I=0,NUM1
      T = X(I) + REAL(I)*TPI/2.0
      X(I) = COS(T)
      Y(I) = SIN(T)
700   CONTINUE
C
C-----Convert to frequency domain----------------------------------
C
    CALL DFFT(X,Y,NUM,M,1)
    CALL REDUCE(X,Y,DX)
C
C-----End Program--------------------------------------------------
C
    RETURN
    END
```

VITA

VITA

Robert Alan Weddle was born in Radford, Virginia on October 22, 1965. He attended elementary schools in Roanoke Rapids, North Carolina, Greenville, South Carolina, and Elizabethton, Tennessee and graduated from high school in June of 1984 from Elizabethton High School. He received the degree of Bachelor of Science in Electrical Engineering in 1989 from Tennessee Technological University in Cookeville, Tennessee. In August of 1990 he entered the University of Tennessee, Knoxville and in December of 1996 received a Master of Science degree in Electrical Engineering.

At the beginning of this thesis, the author was employed at Raytheon Company, Missile Systems Division, in Bristol, Tennessee, where he worked as a Digital Design Engineer. The author is presently employed as a Research and Development Engineer with Extended Technologies Inc. in Elizabethton, Tennessee.