DSP Analysis of Digital Vector Slope Gauge Data Produced by Ocean Wave Simulation

Prof. Earl Schweppe
EECS 803
Introduction to Research
December 9, 2003

Evan L. Bryson
KUID 618093

In memory of my dad,
who earnestly wished to see me complete this work.

"Although no one can start over and make a new beginning,
Anyone can start from now and make a new ending."

- Carl Bard

Acknowledgment

I acknowledge first of all my heavenly Father. Without His assistance I could not have finished this race. I also acknowledge my sister and her family for providing me a place of refuge during an extremely difficult time. My wife, Kathleen; my parents; Jim Hill; Pieter Willems; Paul Taylor; Steve Butler; Woody Davis; and Joyce Meyer have provided invaluable encouragement through this time. My professor for this class, Earl Schweppe; the principal of the Ocean Project, Prof. Richard Moore; my advisor, David Petr; Assistant Dean Mulinazzi; and my other professors, Glenn Prescott, John Gauch, and Jim Stiles have provided timely words of wisdom, and assistance when I needed it most. Many thanks go to Carmela Sibley for her kind words and long conversations, and to Prof. Victor Wallace for providing my entrance into this program. All these have continued to believe in me, even when I did not. Thanks also go to Mike Hulett, Torry Akins, and Justin Legarsky for their last-minute assistance in compiling information for this project.

TABLE OF CONTENTS

1. INTRODUCTION

The Vector Slope Gauge (VSG) is a linearly-swept FM radar designed for the purpose of measuring the varying slopes of ocean wave surfaces at close ranges. Motivations for the work include, but are not limited to 1) verification of SAR images collected by aircraft and spacecraft and 2) eventual design modification as a shipboard radar for measuring real-time local ocean wave behavior. Design and analysis of the first two versions of the VSG are well documented.

The first two VSG versions averaged radar returns internally and stored text results to hard disk. The third version departed from this method by digitally sampling the IF returns and storing the samples and beam data to binary files on hard disk. This paper concerns simulation and analysis of the third version of the VSG. Since the third version has not been tested on the ocean, this paper will divide into two main topics: 1) generation of simulated ocean wave data, and 2) DSP analysis of the simulated data.

Simulated ocean wave data is generated in the same form as the data output from the VSG. In addition to Gaussian noise present in previous VSG measurements, data from the digital VSG will also contain some quantum (sampling) noise. It is expected that the advantages of digital signal processing will outweigh losses attributed to quantum noise.

Inherent errors associated with the VSG collection method are incorporated into the generation of simulated ocean wave data with some

approximations. Previous analyses exposed three sources of error: 1) skewing of slope measurement due to sequential measurement method, 2) slope range measurement error due to wave motion at non-vertical incidence angles, and 3) slope modulation of radar return energy due to varying aspect of waves with respect to radar position.

Experiments with a flat, still surface are compared to results obtained from simulated inputs. Results from simple single wavefronts are compared to simulated inputs. Doppler shift added to these analyses verify proper function of program modules.

1.1 Motivation

A number of motivations drive the design and testing of the VSG as described in [Hesany, 1994 and Evans, 1994]. The VSG can determine wave energy distribution, wave frequency, and predominant wave direction. The close-range data may be compared to spaceborne SAR images and used to determine how winds affect ocean waves. Ocean wave effects on shipping and offshore structures is another motivation, as are coastal sediment migration and wave diffraction and refraction from the shoreline. For remote sensing, VSG results help the understanding of tilt and hydrodynamic modulation on radar returns.

1.2 Three versions of VSG

The versions will be designated VSG-1, -2, and -3 for brevity. VSG-1 and VSG-2 processed the resulting IF within the analog circuitry of the VSG and stored the processed ASCII data on disk. All versions switched between the antenna horns sequentially.

VSG-1 was used in the SAXON-FPN experiment from the Nordsee oil platform in the North Sea during November 1990. It switched between beams at 30 Hz, with a 20ms delay after switching [Hesany, 1994]. This radar had a range error of 10 cm.

VSG-2 was tested at the U.S. Army Field Research Facility in Duck, NC in December 1995 [Legarsky, 1996]. Duck Pier extends about 1000 meters into the ocean, avoiding some of the shoreline effects on ocean waves. This version switched beams at a 300 Hz rate, with a 2.3 ms delay at switching. Faster switching allowed the radar a 1 cm range error, which came out to less than 1% slope measurement error.

VSG-3 departs from the other two versions in that it samples the IF and stores the samples in binary files, rather than processing the data internally. This process lends itself to DSP analysis. This version was never tested on the ocean, so only rudimentary data exists; simple synthesized ocean data is used for this analysis.

2. Basic Ocean Wave Behavior

The study of ocean waves is a complex subject. One can make a highly detailed analysis, yet not have a complete description of their behavior. Tides, gravity waves, tsunamis, ship waves, and capillary waves are all ocean waves. This paper will focus primarily on simplified gravity waves. Entire books are written to describe the complex shapes and behaviors of ocean waves. The purpose of this project was to present simplified wave shapes to the MATLAB® DSP modules, rather than to reproduce the complex wave shapes which occur in nature.

As with other waves occurring in nature, ocean waves can be described by a sum of sinusoids, especially in the deep ocean. It is there that the basic sinusoidal shape is better preserved. Most people do not witness this wave shape unless they have been aboard a ship in the open ocean. As waves near a shore, the water becomes shallower, which alters the shapes of the waves. Energy once concealed beneath the surface is constrained by the rising bottom to appear increasingly above the surface. Eventually the waves peak, pitch forward, and break on the shoreline, as most people are accustomed to seeing them. Waves begin their decline in the region where the water depth decreases to half a wavelength.

2.1 Wave Superposition

A superposition of sinusoids of varying wavelengths, heights, directions, and velocities make up a "sea" (Figure 2.1). A single wavefront may be defined by the equation

$$h(x, y, t) = A\cos(\sigma t + \kappa x \sin\alpha + \kappa y \cos\alpha + \phi)$$ (Kinsman, 1965)

where *h* is wave height relative to mean water level, *A* is amplitude, *κ* is the wave number, *x* and *y* are axes, *φ* is the phase angle, and *α* is the angle of approach, with 0° corresponding to the y direction and increasing clockwise. The variables σ and κ are defined as $\sigma = \frac{2\pi}{T}$ and $\kappa = \frac{2\pi}{L}$.

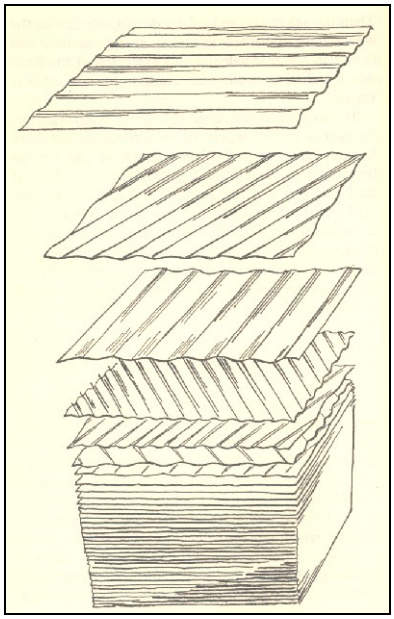

Figure 2.1.
Superposition of Wavefronts (Bascom, 1964).

A developed sea has a period *T* ranging from five to twelve seconds. The wavelength *L* is then $L = \frac{g}{2\pi}T^2$ where g = 9.8 m/sec^2. Wave velocity *C* in deep

ocean is $C = \sqrt{\frac{gL}{2\pi}}$. When water depth decreases to less than *L*/2, velocity decreases and a new term become significant in the equation. Then *C* becomes

$$C = \sqrt{\frac{gL}{2\pi}\tanh\frac{2\pi d}{L}}.$$

Ocean wave energy spectrum is shown in Figure 2.2. A portion of the spectrum is shown in Figure 2.3 in graphical form with wave periods. Of interest to this project are waves with periods between 5 and 12 seconds, which fall in the classifications of sea and swell. Also of interest are capillary waves, or ripples, which are discussed below.

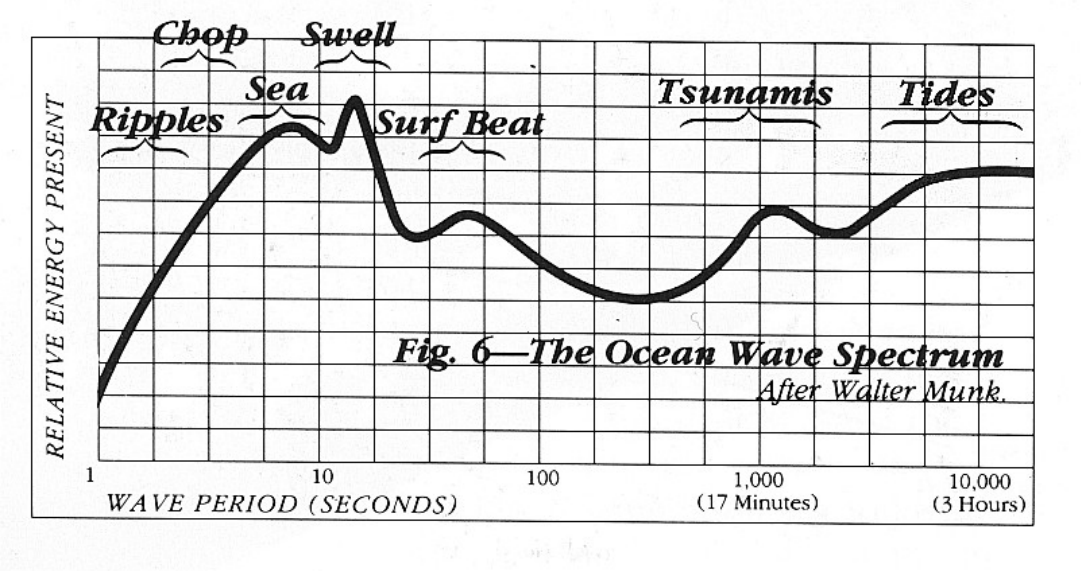

Figure 2.2. Ocean Wave Energy Spectrum
[Kampion, 1997]

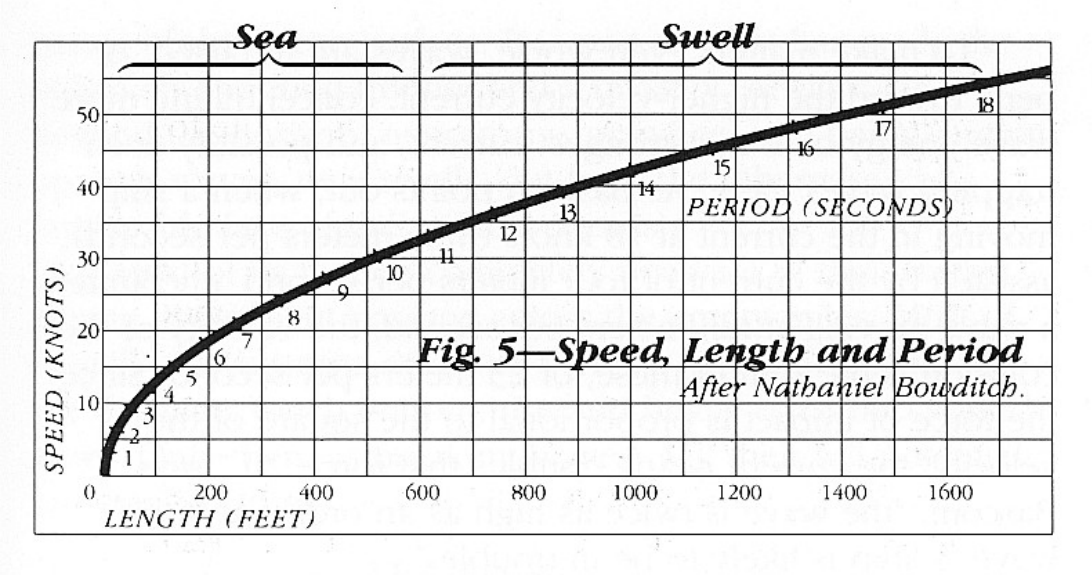

Figure 2.3. Wave Speed vs. Wave Length with Period.
[Kampion, 1997]

2.2 Orbital Motion

Orbital motion is the motion of the water particles within the wave. (See Figure 2.4.) As the wave moves through its cycle, the water near the surface moves in an elliptical pattern. At the end of each cycle of the wave, net displacement of the water particles is near zero. Further from the surface, the water particles move in smaller ellipses.

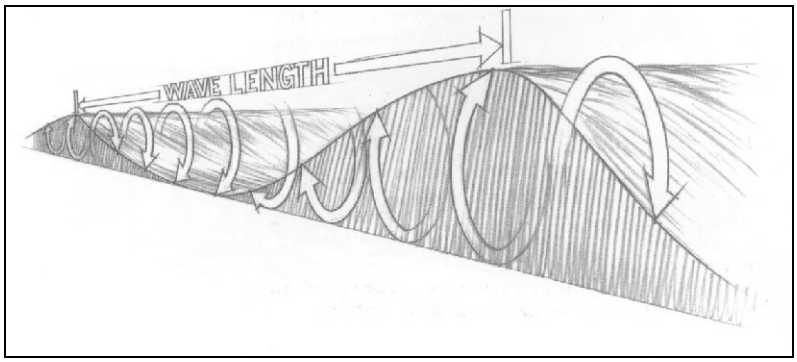

Figure 2.4. Orbital Motion in Approaching Wave [Kampion, 1997]

2.3 Capillary Waves

Capillary waves, or ripples, are small waves occurring on the surface of the gravity waves. These are generated by wind currents working against the water surface. Wavelengths of capillary waves are in the millimeter range; periods are in the range of fractions of a second.

During World War II in the early days of radar, operating frequencies were much lower than they are today. The desired targets over the English Channel were German war planes. One undesired radar return was clutter from the water surface. After the war, scientists discovered a symmetry to the clutter and determined that it was due to Bragg scattering. Whenever ocean wavelength corresponded to half the radar operating wavelength, large radar returns occurred.

Capillary waves make it possible to receive sizable radar returns at the VSG operating frequency of 34.6 to 34.9 GHz. Since the wavelength at this frequency range is about 8.7 mm, capillary waves often produce Bragg scattering. VSG radar returns come primarily from this phenomenon rather than the smooth ocean surface.

3. VSG Function

The Vector Slope Gauge (VSG) is a 35 GHz linearly-swept FM radar used to measure the vector slope of an ocean surface. Function of the VSG is fully discussed in [Haimov and Moore, 1993; Hesany, 1994; Evans, 1994; and Legarsky, 1995 and 1996]. A brief summary of VSG function is in order here.

The VSG transmits FM radar energy in sequence to three spots arranged in an orthogonal pattern on the ocean surface (Figure 3.1). Returns from the three spots each contain frequency information proportional to the range to each spot. Trigonometric computations with the known physical configuration and three range measurements are used to compute a nearly instantaneous estimate of vector slope of the surface. Vector slope is defined as a vector perpendicular to a plane defined by the x, y and z positions of the three radar spots.

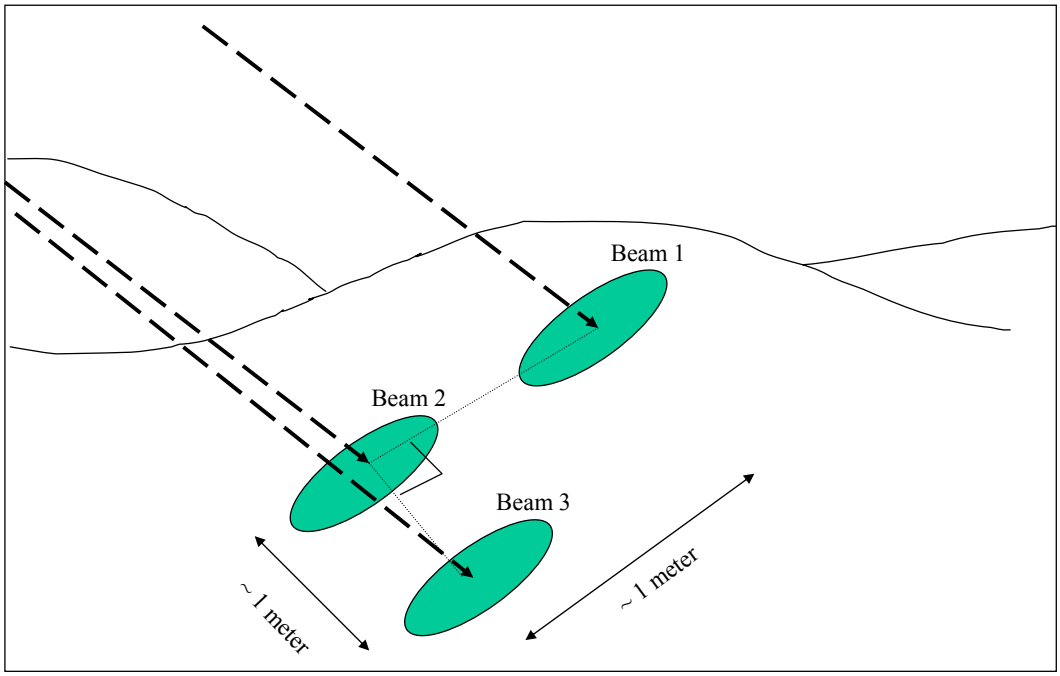

Figure 3.1. Beam Spot Arrangement on Ocean Surface

The VSG routes radar energy to Beam 1, Beam 2, and Beam 3 sequentially. Each beam experiences one upsweep and one downsweep (Figure 3.2). Careful aiming of these beams creates the orthogonal beam pattern on the surface. Returned energy mixes with the transmit frequency to produce three difference frequencies (IF's). These IF's are the frequencies corresponding to the ranges to each spot.

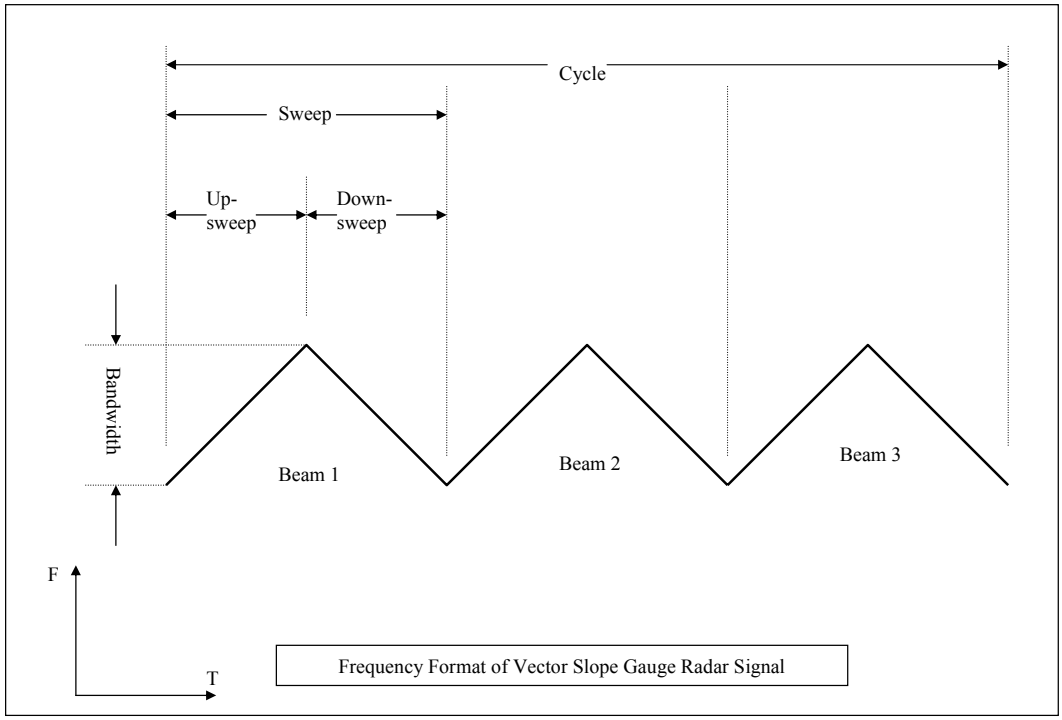

Figure 3.2. Frequency Format of VSG Radar Transmitted Signal

Figure 3.3 illustrates frequency format of the VSG returns from a point target. Obviously, the ocean surface is not a point target, so frequency spreading will occur in the actual radar return. Furthermore, the figure illustrates return from a *stationary* point target. Ocean waves are not stationary targets; therefore, Doppler shift will be present in the return, as shown in Figure 3.4.

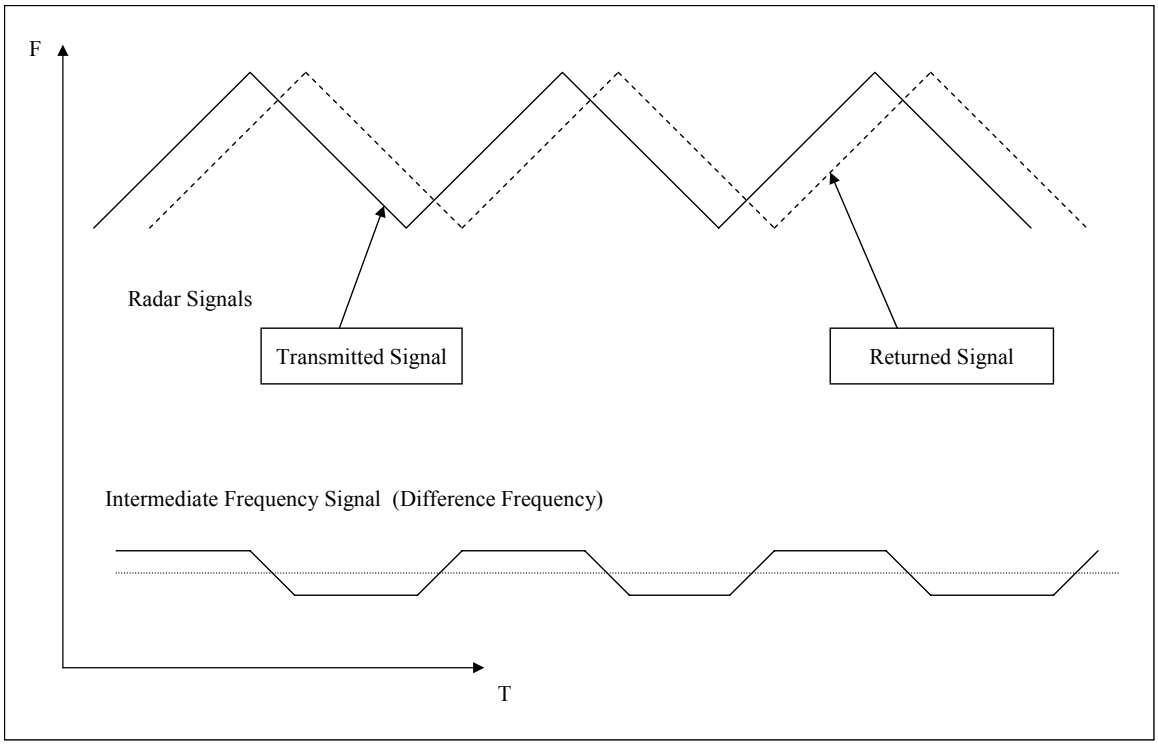

Figure 3.3. FM Radar Return from a Point Target
[Adapted from Ulaby et al, 1986]

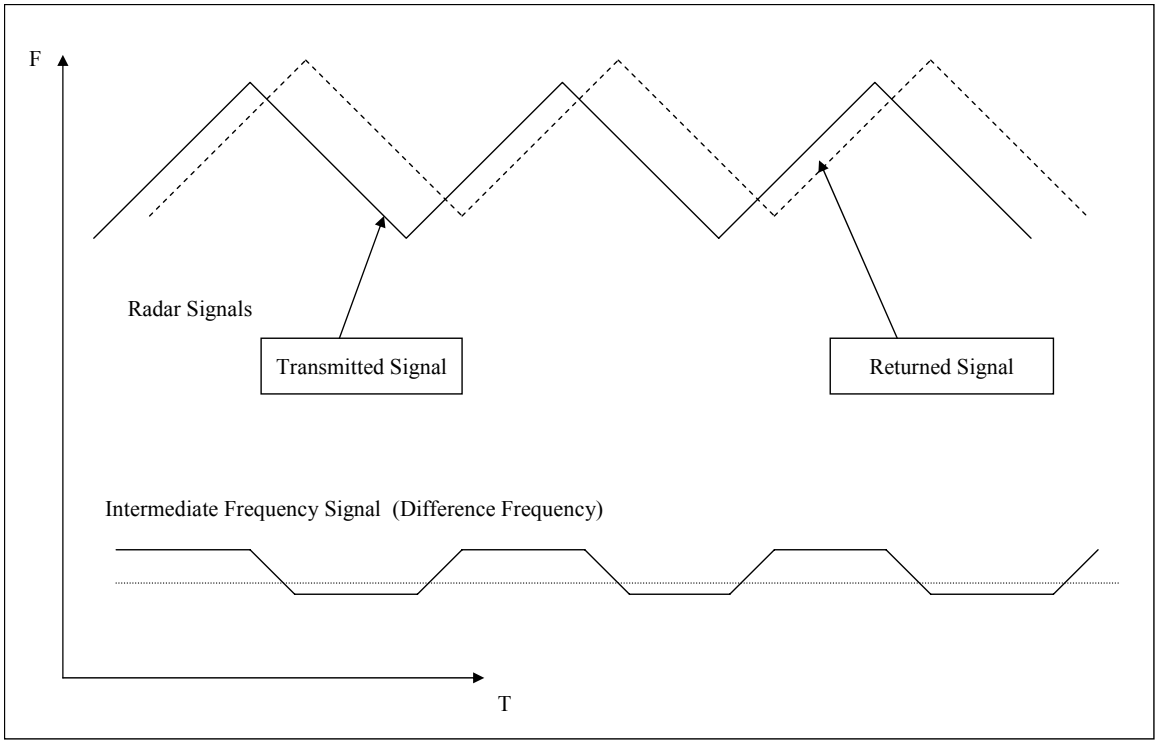

Figure 3.4. FM Radar Return from a Point Target with Doppler Shift Added
[Adapted from Ulaby et al, 1986]

Note that while upsweep and downsweep IF's are equal for a stationary target, the IF's from a moving target are shifted due to added Doppler shift. Curiously, no Doppler shift appears due to the forward wave velocity. Doppler shift does occur, however, due to the orbital velocity of the water within the wave [Ulaby, et al, 1988].

Bandwidth and sweep rate are selected to produce a mixing frequency corresponding to the desired range window, according to the equation:

$$f_s = \Delta f_R(T) = \frac{2BT}{T_R} = \frac{4BR}{cT_R}$$ [Ulaby et al, p. 512]

where: f_s = Signal frequency (IF)
B = Bandwidth of sweep
T = Time for signal to travel distance 2*R
T_R = Repetition period
R = Range to target
c = Speed of light

Example

Let B = 600 MHz, R = 14.14 meters, desired f_s = IF = 450 kHz (typical VSG values). Desired repetition period, T_R, then becomes approximately 250 ms, or $f_R \approx 4$ kHz. It is useful when working with the VSG to obtain the following numbers for a given configuration:

Since $c = 3*10^{10}$ cm/sec, the signal travels 1 cm in $\frac{1\,\text{cm}}{3*10^{10}\,\text{cm/s}} = 3.33*10^{-3}\,\mu\text{s}$. The time for one upsweep or one downsweep in frequency is 125 ms, then

$$\frac{\Delta B}{\Delta T_R} = \frac{600\,\text{Mhz}}{125.7\,\mu\text{s}} = 4.77\,\text{MHz}/\mu\text{s}.$$

Translating from time to range:

$$4.77\ \text{MHz}/\mu\text{s} * 3.33 * 10 - 3\ \mu\text{s} = 159.1\ \text{Hz}/\text{cm}$$

Since the energy must traverse the distance twice:

$$R = \frac{f_{IF}}{2\ *\ 159.1\,\text{Hz}/\text{cm}} = \frac{450\ \text{khz}}{318.2\ \text{Hz}/\text{cm}} = 1414\,\text{cm}$$

Any f_{IF} substituted into this equation will produce the range in cm for this configuration and VSG setting. Of course, f_{IF} must lie within the bandwidth of the IF filter.

For DSP sampling in VSG-3, the sample rate should be at least the Nyquist rate plus about 10%:

$F_s = 450 \cdot 10^3\ \text{Hz} * 2 * 1.1 = 1\,\text{MHz}$ and $T_s = 1\ \mu\text{sec}$

Since $F_{sweep} = 4000\,\text{Hz},\ T_{sweep} = 250\ \mu\text{sec}$, then

N = 250 samples/sweep or *N* = 125 samples per up/down sweep.

4. VSG Output Data Simulation

4.1 VSG-3 Output Data Format

Each binary file begins with 24 bytes of header information and continues with an unspecified length of 2-byte data items. Data recording will nearly always begin during a cycle. Similarly, the ending cycle will also be incomplete. Therefore, the first and final [partial] cycles must be discarded.

4.1.1 Header Format

Each binary file begins with 24 bytes of header information in this format:

Variable Name	Size (bytes)	Data Type	Description
Original	4	uint	File number
Time 1	1	uint	Month
Time 2	1	uint	Day
Time 3	1	uint	Hour
Time 4	1	uint	Minute
FBeam	4 x 3	float	3 sweep freqs
Fs	4	float	Sample freq

Table 4.1

For long data sequences, the VSG may break up the binary data into a series of shorter files. The *Original* variable will be a “1” for the first data file. It may be zero for remaining files in the sequence, or it may contain “2,” “3,” etc. to indicate the proper sequence of the files. The main concern was to mark the first file in the sequence with a “1,” hence the name *Original*.

The *Time* variables are self explanatory. They indicate the time each data sequence was generated.

FBeam contains three sweep frequencies, for the up/down sweep on each of the three beams. Although these variables are recorded three times, it would be very unusual to have one beam's sweep frequency differ from another's. This amounts to the frequency of sampling of the ocean wave.

Fs is the DSP sample frequency, or the rate of digital sampling of the IF return within VSG-3. Fs is always higher than FBeam. From Fs and FBeam, one can determine the average number of DSP samples in each sweep.

4.1.2 Data Format

After the header information, each binary file contains a long data series. The data format produced by the VSG contains three pieces of information: The beam number (2 bits), up or down sweep direction (1 bit), and unsigned integer data (12 bits). Arrangement of the bits in one datum is shown in Figure 4.1.

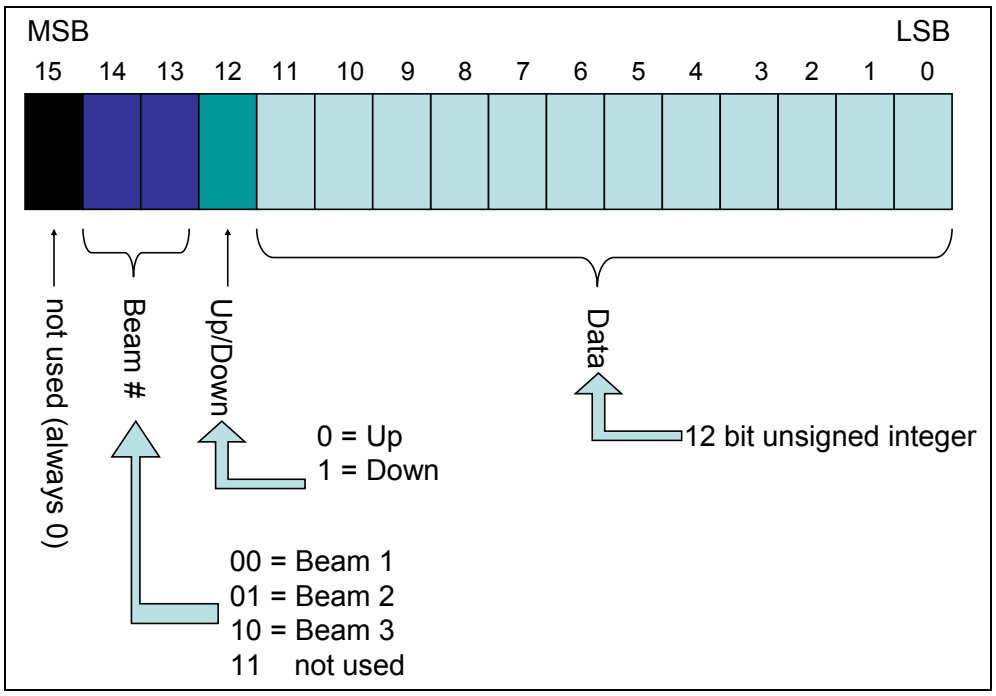

Figure 4.1. Digital VSG Data Output Structure

The unsigned integer has a range from 0 to 2^{12} – 1 or 4095. In the analysis phase, 2048 is subtracted from the data to bring it back to zero mean.

4.2 Wave Simulation Algorithm

No ocean data or Clinton Lake data was taken on VSG-3. Rudimentary data was taken in a hallway at the Remote Sensing Laboratory. VSG-2 data taken at Duck Pier and Clinton Lake was available, but was not usable since it was created and stored in an entirely different format. Therefore, it became necessary to generate simple ocean wave data in VSG-3 format.

It was not necessary to recreate a complex ocean surfaces such as those occurring in nature. The *Wavesim7.m* module creates data for flat surfaces and simple sinusoidal wavefronts. It is capable of injecting Doppler shift and noise into the return signal. These capabilities are sufficient for testing the basic function of the DSP modules.

The programmer supplies VSG-3 configuration and desired ocean wave parameters in the command line of *Wavesim7.m*. The block diagram in Figure 4.2 contains three modules based upon (Evans, 1994). (See Appendix B for complete code.) Modules *slrangemod.m* and *slopepoimod.m* are slight modifications of Evans' modules. The modifications allow the data to fit the VSG-3 format. Module *slopeab2.m* is an exact recreation of Evans' module.

The top row of Figure 4.2 computes and stores the slopes of the simulated ocean wave, as well as providing ranges to the three beam spots on the surface. Module slrangemod.m requires a range estimate for input, which is computed by

RangeEst.m. It then performs an iterative process on two nonlinear simultaneous equations according to Newton's method (see Errors section below).

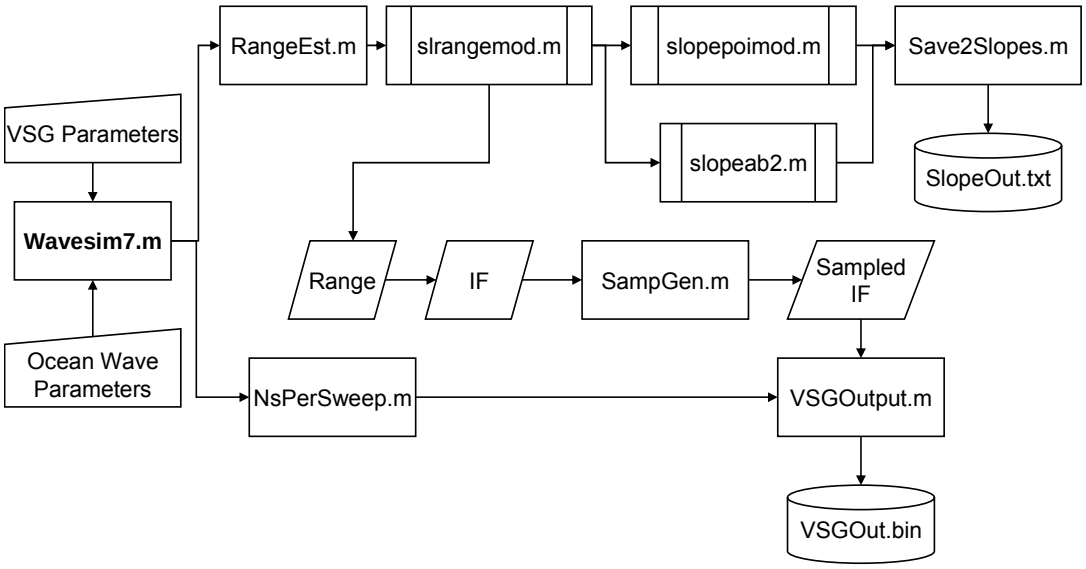

Figure 4.2. Wavesim7.m Block Diagram

Module *slopepoimod.m* uses range data to calculate a time series of exact x and y slopes (in degrees) based upon the derivative of the waveshape. Module *slopeab.m* uses x, y, and z data for the three radar spots to approximate x and y wave slope (in degrees) by extending a plane through the three points. Exact and approximated slope series are saved in a text file by *Save2Slopes.m*. These are for later comparison to processed data.

The middle row of Figure BB shows each range converted to an IF. The IF is sampled at *Fs* by module *SampGen.m* and submitted to *VSGOutput.m*.

Since *Fs* is usually not a multiple of *FBeam*, the number of samples in an up or down sweep has some variability. For example, if sweep frequency is 3000 Hz and sample frequency is 1 MHz, the average number of samples in an

up/down sweep is 166.67. This means that some sweeps will have 167 samples and others will have 166. Module *NsPerSweep.m* computes an M x 3 matrix containing the number of samples in each sweep of the data block to be created. VSGOutput.m uses this information and the sampled IF data to create a binary file just as VSG-3 would create it.

4.3 Errors

4.3.1 Built-in errors

4.3.1.1 Quantization Noise

The digital VSG introduces quantization noise to the signal during the IF sampling process. This was not an issue in VSG-1 and VSG-2. Storage of data in the exact configuration as VSG-3 output assures the same noise will be present in the simulation. Theoretical quantization noise for 12-bit uniform quantization is defined by:

$$\sigma_N^2 = \frac{\Delta V^2}{12}$$

where (Frerking, 1994)

$$\Delta V = \frac{V_{p-p}}{2^b}$$

Since the stored VSG data is an unsigned integer, ΔV may be regarded as unity and the variance as 1/12.

4.3.1.2 Slant Range Measurement Error

[Evans, 1994] deals with slant range measurement error in terms of phase error in the ocean wave measurement. Generally, measurements are not taken from the vertical, or θ = 0° incidence angle. Larger incidence angles produce

larger phase errors, since the VSG measures the change in length along the radar beam, not the amplitude of the wave. (See Figure 4.3.) Wave propagation aligning with the RLD produces the largest phase error, whereas propagation transverse to the RLD produces no error, regardless of the incidence angle. Evans computes the largest phase error (in degrees) in this way:

$$\varepsilon_{\text{phase}}(\max) = \frac{A \tan\theta \ \cos\phi \ 360°}{\lambda}$$

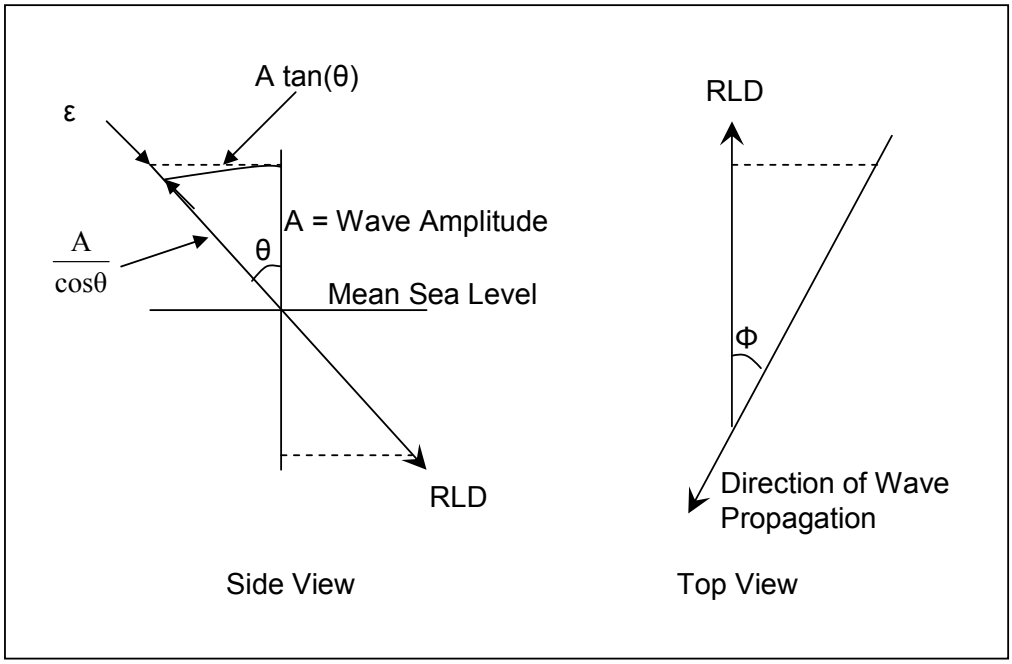

Figure 4.3. Geometry used to calculate phase and range error in a single beam due to slant range measurement. Adapted from (Evans, 1994).

In this simulation, it is necessary to compute the actual distance along the RLD rather than the phase error relative to the ocean wave. For this purpose, an approximation may be used:

$$\Delta R_{\text{slant}} = \frac{h(t) \ \cos\phi}{\cos\theta}$$

where ΔR_{slant} = Distance along RLD from mean sea level
$h(t)$ = Height of wave at time t

The approximation leaves a small error ε, as shown in Figure 4.3. This error is the difference between the horizontal at the RLD point and the height of the wave at the RLD point. Error ε is zero for a calm surface and increases with wave amplitude.

For a more precise solution, [Evans, 1994] wrote *slrange.m* to compute an iterative solution to these two equations. This module uses Newton's method to approximate the solution to the two equations to a very small error of 10^{-13}:

$$r_{n+1} = r_n \frac{f(r_n)}{f'(r_n)}$$

The module *slrange.m* is called in *wavesim7.m.*

4.3.1.3 Error resulting from sequential measurements

This error, treated in [Evans, 1994], was more pronounced in earlier versions. The error results from the fact that the radar beams are switched, and returns measured, sequentially. From one measurement time to the next, an ocean wave will travel a small distance. This will skew the slope a small amount depending on the speed with which samples are taken.

4.3.2 Extractable Errors

4.3.2.1 Gaussian Noise

Gaussian noise is extractable only in the data simulation sense, in that it may be included in a simulation or not, so that other phenomenon may be isolated. Gaussian noise is present in all versions of the VSG, as it is in any real-world measuring device. It is added to the simulation in the final analysis of

program module function. It is excluded in other simulations in order to isolate the effects of other error modes.

4.3.2.2 Doppler Shift

Doppler shift is not a true error. However, if not considered and dealt with appropriately, it will introduce errors into the range measurements. Doppler shift is extractable in the simulation sense, and the DSP analysis attempts to isolate it for measurement. It may be included in a simulation to compare processed results with original Doppler shift input.

4.3.3 Errors not included

4.3.3.1 Slope modulation

Generally we analyze the radar always looking in the y direction. In this case, the slope variation in the x direction will have no effect on radar return. Then variation in y slope varies the radar return in this proportion:

$$\sigma^\circ \propto \left[\frac{1 + s_y \tan\theta_p}{\tan\theta_p - s_y} \right]^4 \qquad \text{[Ulaby et al. p.1694]}$$

where σ° = Relative radar power return

θ_p = Radar pointing direction (incidence angle)

s_y = y component of water surface slope

4.3.3.2 Frequency Spreading

The ocean surface is not a point target and the radar beam width is not infinitesimally small. This results in a continuum of ranges to a radar spot on the surface, which in turn results in a range of frequencies in the IF return. The

frequency return may be approximated from an integration of the ranges to the area of a spot, out to the -3dB points. This frequency spreading is discussed in [Ulaby et al, 1988]. At angles departing from perpendicular incidence, the frequency spreading increases, as may be inferred from Figure 4.4.

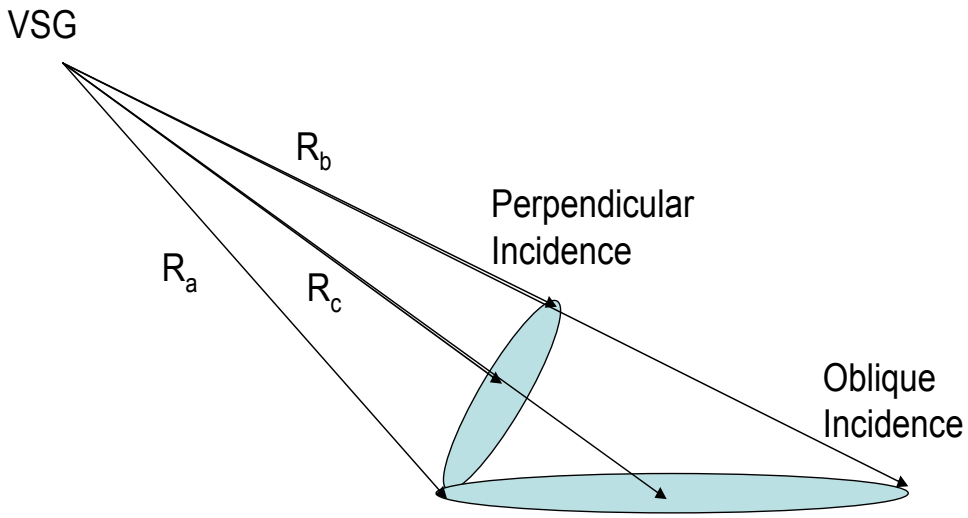

Figure 4.4. Illustration of Range/Frequency Spreading
In an FM Radar

Inclusion of frequency spreading will provide a more realistic simulation of radar returns from the VSG.

Variation of returned signal power in the simulation according to this proportionality approximates actual ocean data return. Gaussian noise will remain statistically constant, while signal return strength varies. Maximum return occurs when the wave face is perpendicular to the RLD. This is practically never the case, as the slope relative to the RLD constantly changes. Inclusion of this return variation will create a more realistic simulation of received power.

5. DSP Analysis

5.1 DSP Algorithm

A DSP algorithm, developed by [Hamilton, 1998] in five stages, is found in summary form in Appendix C. Figure C.1 is a block diagram of this algorithm. At the time of writing, the algorithm was incomplete. Portions of Hamilton's algorithm are incorporated into this analysis.

Figure 5.1 contains a block diagram of DSP2.m; MATLAB® code is found in Appendix B. Modules *FindPSD.m* and *FindEffFreq.m* are used directly as

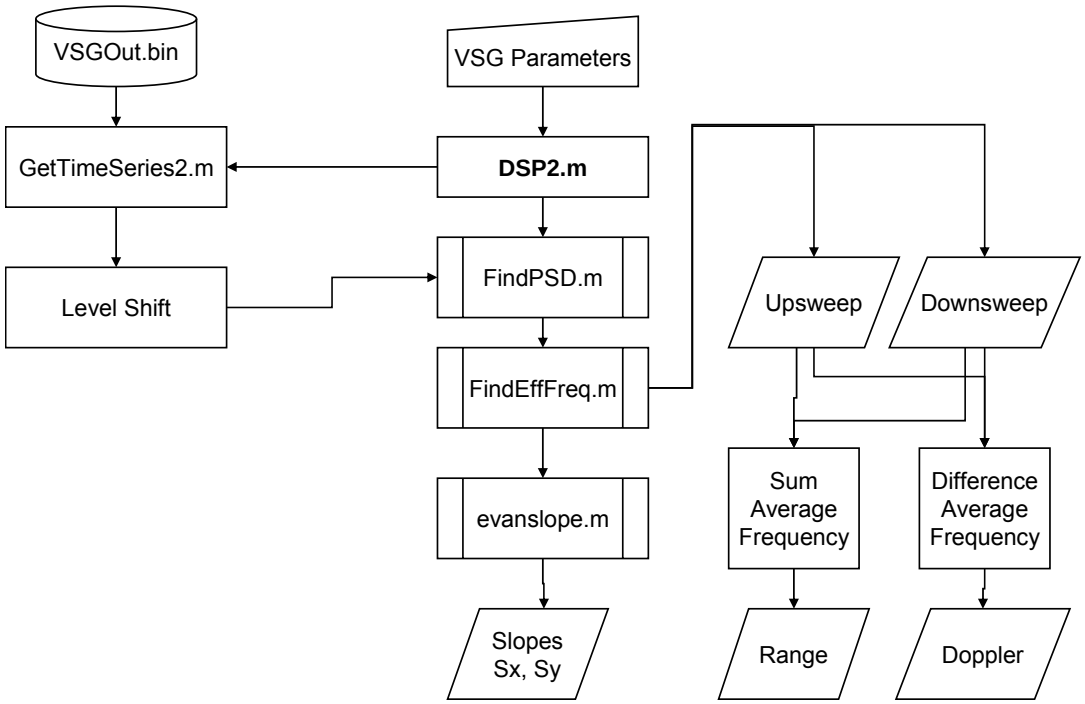

Figure 5.1. DSP2.m Block Diagram

written by [Hamilton, 1998]. Module *evanslope.m* is derived from trigonometry and coordinate transformations in [Evans, 1994].

VSG binary data, stored by module *WaveSim7.m* in the file *VSGOut.bin*, is accessed by *GetTimeSeries2.m*. The programmer specifies the time offset (*Toffset*) and number of cycles (*M*) desired to average (usually *M* = 4). GetTimeSeries2.m moves a pointer into the binary file corresponding to the time offset, and finds the beginning of the cycle nearest to the pointer. This becomes the beginning file pointer. The module then looks *M* cycles later to find the end file pointer. Binary search for the file pointers is unnecessary, since close approximate locations may be calculated prior to search. Once a file pointer is placed, the direction of search is determined by the beam number and up/down indicator of the current datum. The program searches the data sequentially backwards or forwards. It usually has only to step once or twice to find the starting/ending pointer. Once pointers are established, the module reads in the data between the pointers and places it in an *M* x 6 x N_{max} matrix, where N_{max} is the maximum DSP sample size, M is the number of cycles read in, and 6 is the number of up/down sweeps for all three beams.

Application of *FindPSD.m* and *FindEffFreq.m* to each “column” of DSP samples results in an M x 6 matrix of effective frequencies. Simple multiplication results in an M x 6 matrix of ranges. Separation of the ranges into three upsweep and three downsweep ranges enables computation of sum and difference average frequencies, which enables computation of range and Doppler values, respectively (see Appendix 1 for a description of these computations). Module *evanslope.m* computes slopes in the x and y directions for comparison with generated data.

5.2 Frequency Estimation

Accurate estimation of frequency is key to obtaining the best slope measurements. The desire is that DSP sampling and analysis will produce more accurate estimates of the frequencies and associated ranges than the previous analog processing technique.

Frequency is defined as the time derivative of phase:

$$f = \frac{d\phi}{dt}$$

We cannot estimate a frequency from one data point, but we may attempt to estimate frequency from small piece-wise segments of a signal. Small segments of a signal may be relatively stationary, whereas the entire signal may not be. This is the case with the radar return from ocean waves. Since the waves are in motion, the ranges to the waves are constantly changing, which means the radar's IF is constantly changing. Overall, then, the returns from ocean waves are not stationary. But taken in short enough time segments, these returns become quasi-stationary. In this case, most stationary frequency estimation techniques may be applied.

For VSG-2, the averaging of range frequencies over four cycles was considered the optimum method. At 3.33 ms per sweep, this amounted to averaging the returns over a period of 40 ms. A gravity wave travels between 5 and 12 m/sec. In 40 ms a wave with velocity 8 m/sec will move 32 cm (if the wave travel is along the RLD).

Using values from the previous example for VSG-3:

Example

F_{beam} = 4000 Hz, B = 600 MHz, 8 m/s wave velocity

$$125 \cdot 10^{-6} \text{ sec} * 8 \text{ m/s} = 1 \cdot 10^{-3} \frac{\text{m}}{\text{(up/down) sweep}}$$

In 4 cycles the wave will move this distance:

$$1 \cdot 10^{-3} \frac{\text{m}}{\text{(up/down) sweep}} * 2 \frac{\text{(up/down) sweeps}}{\text{sweep}} * 3 \frac{\text{sweeps}}{\text{cycle}} * 4 \text{ cycles} = 24 \cdot 10^{-3} \text{ m} = 2.4 \text{ cm}$$

The frequency excursion for 4 cycles will be:

$$1 \cdot 10^{-3} * 24 * 159.1 \frac{\text{Hz}}{\text{cm}} * 100 \frac{\text{cm}}{\text{m}} = 440 \text{ Hz}$$

which amounts to only

$$\frac{440 \text{ Hz}}{350 \cdot 10^{3} \text{ Hz}} = 0.125 \% \text{ shift}$$

Let M be the number of cycles over which to average. An engineering tradeoff must be made in choosing the value of M. Increasing M will increase the estimation accuracy, but decrease the stationarity of the estimated frequencies. Decreasing M will increase stationarity while decreasing the accuracy of the estimation. Clearly, 4 cycles of averaging will have better stationarity for VSG-3 than it did for VSG-2.

In *DSP2.m*, we use a discrete version of Rice's expected number of zero crossings to determine effective frequency [Hamilton, 1998]:

$$f_{eff} = \sqrt{\frac{\sum_{F_l}^{F_u} k(i)2 \cdot PSD(i) \cdot \Delta k}{\sum_{F_l}^{F_u} PSD(i) \cdot \Delta k}}$$

where

$PSD(i)$ = power spectral density approximation vector [1 x N/2 + 1]

$k(i)$ = Discrete frequency vector [1 x N/2 + 1]

Δk = Discrete frequency vector spacing

F_u = Upper frequency limit

F_l = Lower frequency limit

Module *FindEffFreq.m* performs an FFT on each sweep, resulting in a discrete frequency spectrum. Choosing the frequency of the largest element would be inaccurate, since there exist surrounding frequency elements of lesser strength which affect the result. The desired effective frequency will, in all likelihood, reside between two discrete frequency elements. Rice's method provides an interpolation of the FFT elements and a much more precise estimate of single frequency content.

5.3 Module Verification

All modules were verified for function and accuracy during synthesis without added Gaussian noise. Ranges and IF's were checked during synthesis of the modules to assure freedom from errors. Slopes extracted by *DSP2.m* matched slopes produced by *slopeab2.m*. The slopes closely resembled the slopes produced by *slopepoimod.m* with slight error, as expected (See Section 4.3.1.2 on slant range error.) This was true for incidence angles up to about 65 degrees, which was the design limit of the VSG [Legarsky, 1995]. Higher

incidence angles applied to these modules generated difficulties in *slantrangemod.m* module, where iteration by Newton's method took place. Velocities injected into a flat surface in *Wavesim7.m* module were extracted by *DSP2.m* in the correct values and directions.

The only known VSG-3 data was taken against a vertical wall at a range of about 5.75 meters. This data was treated as return from a flat surface at zero degrees' incidence. Application of *DSP2.m* to this data produced near expected results. Slope values in the *y* direction were about 2 degrees, regardless of the number of averaged cycles. Slopes in the *x* direction were near zero. The wall being a stationary object, error due to sequential measurement did not affect accuracy, and cycle averaging did not improve repeatability. Since the wall was flat and angle of incidence was zero, slant range error was not a consideration. The 2-degree slope discrepancy in the *y* direction was probably due to pointing error. However, no precise physical measurements corresponding to this data were recorded.

6. CONCLUSION

This paper serves as documentation of simulated input and subsequent DSP analysis programs for VSG-3, the digital vector slope gauge. Simple wave simulations incorporate most known measurement errors and anomalies from previous versions, as well as from VSG-3. Known errors included skewing of slope measurements due to sequential measurement method and slope range measurement error due to non-vertical angles of observation of vertical wave motion. Modules are capable of introducing Gaussian noise and Doppler shift into the data. Simulations create binary data in the format produced by VSG-3, thereby embedding quantization noise in the data.

Not included in the simulation was slope modulation of radar return energy due to varying aspect of waves with respect to radar position. Frequency spreading due to finite width of the radar beam is also not included in the simulation.

Simulations of simple flat surfaces and single sinusoidal wavefronts enable primary evaluation of DSP modules. Expected output of the VSG with flat surface and single wavefront, with and without Doppler, is verified. No Gaussian noise is included in these evaluations. Evaluation of the only known VSG-3 data reveal expected measurements from a vertical wall in a hallway at a range of about 5.75 meters. This data shows little frequency spreading, since the angle of incidence is near zero.

DSP modules extract VSG-3 data from binary data files at operator designated points and will process one to 75 cycles of data, as desired. The modules perform PSD on sampled IF's and estimate the sampled frequencies for each of the three beams. From these estimates the modules compute *x* and *y* slopes, ranges to each beam, and Doppler shifts.

6.1 Future Work

Further evaluation of DSP modules should include performance capability with Gaussian noise added. We also need to interpret orbital velocity measurements. Frequency spreading and slope modulation may or may not be necessary in the simulation stage.

The next step will be to take measurements on actual ocean waves from Duck Pier or an oil platform for accurate comparison to previous data from VSG-1 and VSG-2. This would be followed by real time processing using a TI-30 or similar DSP board. Finally, depending on the success of the preceding steps, we will need to integrate the VSG with a shipboard environment.

Appendix A

Glossary

Beam - One of the three lobes of microwave energy which strike the ocean surface and enable measurement of vector slope. Beams are numbered 1, 2, and 3. Each beam has a frequency upsweep and downsweep.

Cycle - A series of six vectors corresponding to one upsweep and one downsweep for each of the three beams.

Doppler – An apparent frequency shift as perceived by an observer, due to a physical motion of a frequency source either toward or away from the observer.

Downsweep – A linear frequency excursion of the VSG in which the frequency changes from high to low, or the data collected from a downsweep.

FPN – *Forschungsplattforn Nordsee*. A German oil drilling platform in the North Sea. Test site of the first version of the VSG.

Group Velocity – Speed of ocean wave movement as a whole.

Orbital Velocity – Speed of water movement within the surface of a wave.

Returned Power – That portion of the transmitted power scattered toward and received by the radar antenna.

RLD - *Radar Look Direction*. For VSG purposes, RLD is usually in the positive y direction.

SAXON – *Synthetic Aperture Radar and X-band Ocean Nonlinearities*. An experiment to compare close-range mm radar results with spaceborne SAR remote sensing results.

Sweep – A contiguous upsweep and downsweep from a single radar beam, or the data collected from such a sweep.

Vector Slope - The instantaneous slope of a wave surface coupled with its direction of travel.

Vector Slope Gauge (VSG) – An FM radar instrument emitting microwave energy in three beams toward the ocean surface. Radar returns from the three beams are processed to obtain the vector slope of the wave.

Upsweep – A linear frequency excursion of the VSG in which the frequency changes from low to high, or the data collected from an upsweep.

APPENDIX B

MATLAB® m-files

Part 1 - Wave Simulation Files

WaveSim7.m

```
function [NSmat,Range,IFreq,IFSamp] =
WaveSim7(BW,Fs,FBeam,HVSG,Thetad,Alphad,T,hwave,PHId,d,TObs);

% [NSmat,Rsamp] =
WaveSim7(BW,Fs,FBeam,HVSG,Thetad,Alphad,T,hwave,PHId,d,TObs);
%   INPUTS
%       Radar Parameters:
%       BW      Bandwidth of frequency sweep (MHz)
%       Fs      DSP sample frequency (Hz)
%       FBeam   Beam sweep frequencies [3] (Hz)
%       Height  Height of VSG from mean ocean surface (m)
%       Thetad  Angle of incidence, from vertical (deg)
%       Alphad  Angle of ocean wave group velocity (deg)
%               [ 0 deg = receding along y axis (away in RLD),
%                 90 deg  = transverse, left to right (along x axis)]
%               wrt RLD
%       Ocean Parameters:
%       T       Period of ocean wave (sec)
%       hwave   Height of ocean wave = 2*Amplitude (m)
%       d       Depth of ocean at measurement area (m)
%       TObs    Total observation time (sec)
%
%   OUTPUTS
%       Binary file containing DSP samples in VSG format
%       Text file containing x and y slopes from two methods

% t = 0 is defined as first transmission time recorded at time of
% arrival at ocean wave.
% Origin is defined as intersection of center beam (2) with
% calm ocean surface.

%   FUNCTION CALLS
%       RangeEst.m      Initial range estimate for slrangemod.m
%       slrangemod.m    Solves nonlinear equations by Newton's method.
%                       Gives ranges to radar spots (C. Evans)
%       slopepoimod.m   Calculates slope at wave by derivative method
%       slopeab2.m      Calculates slope at wave by intersecting plane
%                       method (C. Evans)
%       Save2Slopes.m   Saves results of slopepoi and slopeab2 to text
file
%       NsperSweep2.m   Calculates number of samples in each sweep
%       SampGen.m       Generates DSP samples of IF
%       VSGoutput.m     Saves DSP samples to binary file in VSG format

% FILENAMES
BinFileName = 'D:\VSG\VSGout.bin';
TxtFileName = 'D:\VSG\SlopeOut.txt';

% CONSTANTS
c = 2.998e8;        % propagation velocity (m/s)
dr = pi/180;    % conversion factor, degrees-->radians (rad/deg)
g = 32*12*2.54/100;     % grav const (m/s)

% VSG beams as measured by Justin Legarsky and Torry Akins, Dec. 1995
```

```
% Subtended angles, Beam-to-beam (deg)
Beta12d = 3.53; Beta12 = Beta12d*dr;    % convert to radians
Beta23d = 3.34; Beta23 = Beta23d*dr;
Beta13d = 5.03; Beta13 = Beta13d*dr;
% 3 Db Beam widths (degrees)
ThetaE1d = 1.31; ThetaE1 = ThetaE1d*dr; % convert to radians
ThetaE2d = 1.43; ThetaE2 = ThetaE2d*dr;
ThetaE3d = 1.26; ThetaE3 = ThetaE3d*dr;
ThetaH1d = 1.03; ThetaH1 = ThetaE1d*dr;
ThetaH2d = 1.28; ThetaH2 = ThetaE2d*dr;
ThetaH3d = 0.94; ThetaH3 = ThetaE3d*dr;

Theta = Thetad*dr;
Alpha = Alphad*dr;
PHI = PHId*dr;
Delta = 0;      % No rotation about z' axis

% OCEAN
% Calculate Ocean Surface parameters
F = 1/T;                % frequency of ocean wave
L = g*T^2/2/pi;         % wavelength (m)
Csq = g*L/2/pi;         % deep water velocity squared
Fsh = tanh(2*pi*d/L);   % shallow water factor
C = sqrt(Csq);          % deep water velocity
Csh = sqrt(Csq*Fsh);    % shallow water velocity
Lsh = T*Csh;            % shallow water wavelength
A = hwave/2;               % wave amplitude
if d < L/2
    DepthFlag = 'Shallow water wave';
else
    DepthFlag = 'Deep water wave';
end

% VSG
% Calculate VSG Radar parameters
Ts = 1/Fs;
TBeam = 1/FBeam;
%TSweep = TBeam/2;
dFdTcgs = BW/(TBeam*1e6/2);    % (MHz/us)
dFdT = dFdTcgs*1e12;    % convert dFdT to (Hz/sec)
Ns = TBeam/Ts/2;        % samples per up or down sweep
(samples/sweep/2)
Ns = ceil(Ns);          % max # samples in sweep
Nsweeps = TObs/TBeam;   % # U/D sweeps in total observation time
Nsweeps = fix(2*Nsweeps); % counting each up or down sweep as one
TperM = 1/c;            % Time for signal to travel 1 meter (sec)
IFperM = dFdT*TperM;    % Intermediate frequency as dependent on range
(Hz/m)
NsObs = Fs*60*TObs;     % Total samples observed in TObs

% Display parameters of ocean wave, VSG system

% ********************************************
% * GENERATE OCEAN WAVE BEAM RANGES AND SLOPES *
% ********************************************

% Estimate shortest range to wave surface [1x3] for each beam
[r] = RangeEst(Theta,HVSG,Beta12,Beta23,A);
%r = ones(1,3);
%rless = r;
%r(2) = HVSG/cos(Theta);    rless(2) = A/cos(Theta);
%r(3) = HVSG/cos(Theta+Beta23);     rless(3) = A/cos(Theta+Beta23);
%r(1) = r(2)/cos(Beta12);    rless(1) = rless(2);
%[r; rless; r-rless]
```

```
%r = r - rless;

% calculate time series for ranges and xyz positions
%[Range,x,y,z] = slrange(Theta,Delta,r,Alpha,T,A,HVSG);
[Range,x,y,z] = slrangemod(Theta,Delta,r,Alpha,T,A,HVSG,FBeam,TObs);

% calculate time series for slopes from derivative
%[Sxd,Syd] = slopepoi(Range,Theta,Delta,A,Alpha,F);
[Sxd,Syd] = slopepoimod(Range,Theta,Delta,A,Alpha,T,FBeam);

% calculate time series for slopes from xyz positions
[Syp,Sxp] = slopeab2(x,y,z);

% store slopes to text file
Save2Slopes(TxtFileName,...
    Syd,Sxd,Syp,Sxp,...
    BW,Fs,FBeam,HVSG,Thetad,dFdTcgs,IFperM,Ns,...
    DepthFlag,Alphad,PHId,hwave,A,d,T,F,C,L,Csh,Lsh);

% ****************************************
% * GENERATE FREQUENCIES, SAMPLE, AND STORE *
% ****************************************
% generate matrix of DSP sample sizes for VSG3
[NSmat] = NsperSweep2(TBeam,Ts,TObs);
% NSmat col 1 contains Beam number
%       col 2 contains Up/Down sweep indicator
%       col 3 contains # of DSP samples in sweep

% generate frequency time series from range time series
IFreq = Range*2*IFperM;

% Generate IF, Sample IF at Fs (one up or down sweep)
% add doppler shift
fD = 0;                                 % Doppler ********* TEMP
********
% add slope modulation ? ? ?
IFSamp = SampGen(NSmat,IFreq,Ts,fD);    % IFSamp is [NSweeps x
NSamples]
                                        % These are DSP samples
% add noise

% SCALE SAMPLE VALUES
% to unsigned integers in the range [0, 2^12-1]
MaxSamp = max(max(IFSamp))       % most positive DSP sample
MinSamp = min(min(IFSamp))       % most negative DSP sample
SpanSamp = MaxSamp - MinSamp     % range of DSP samples
IFSamp = (IFSamp - MinSamp)./SpanSamp;    % shift samples to be >= 0
                                          % scale samples to range [0,
1]
IFSamp = fix(IFSamp.*(2^12-1));   % scale samples to range [0, 2^12-1]
                                  % fix to whole numbers

% WRITE DATA TO BINARY FILE
[HdrWrit,DatWrit] = VSGoutput(Fs,FBeam,NSmat,IFSamp,BinFileName);
disp(sprintf('%14i %30s',HdrWrit,' items written to file header'))
disp(sprintf('%14i %30s',DatWrit,' data items written to file'))
```

RangeEst.m

```
function [r] = RangeEst(Theta,HVSG,Beta12,Beta23,A);

% Estimate shortest range to wave surface [1x3] for each beam
```

```
r = ones(1,3);
rless = r;
r(2) = HVSG/cos(Theta);    rless(2) = A/cos(Theta);
r(3) = HVSG/cos(Theta+Beta23);     rless(3) = A/cos(Theta+Beta23);
r(1) = r(2)/cos(Beta12);    rless(1) = rless(2);
%[r; rless; r-rless]
r = r - rless;
```

Save2Slopes.m

```
function Save2Slopes(FileName,...
    Syd,Sxd,Syp,Sxp,...
    BW,Fs,FBeam,HVSG,Thetad,dFdTcgs,IFperM,Ns,...
    DepthFlag,Alphad,Phid,hwave,A,d,T,F,C,L,Csh,Lsh);

% 1st line contains output file name
% 2d line contains slope data
% 3d line contains VSG parameters
% 4th line contains ocean wave parameters
%
% Header contains VSG parameters, ocean wave parameters
% Save slopes in text form in the following format:
%   Derivative slope                Point Estimated slope
%       Sy      Sx                      Sy      Sx
%               .
%               .
%               .                   etc.

[fidOut, Success] = OpenOutFile(FileName);

Form3 = '   %25s %1s %9.4f %7s\n';
Form4 = '   %30s %1s %14.4f %7s\n';

% Output VSG Parameters
fprintf(fidOut,'\n%40s\n','VSG Parameters');
fprintf(fidOut,Form4,'Bandwidth','=',BW,'MHz');
fprintf(fidOut,Form4,'DSP sample freq','=',Fs,'Hz');
fprintf(fidOut,Form4,'Sweep freq, per beam','=',FBeam,'Hz');
fprintf(fidOut,Form4,'VSG height above surface','=',HVSG,'m');
fprintf(fidOut,Form4,'Incidence angle (theta)','=',Thetad,'degrees');
fprintf(fidOut,Form4,'dF/dT','=',dFdTcgs,'MHz/us');
fprintf(fidOut,Form4,'IF per range','=',IFperM,'Hz/m');
fprintf(fidOut,Form4,'Samples per U/D sweep','=',Ns,'samples/sweep');

% Output ocean wave parameters
fprintf(fidOut,'\n%40s\n','Wave Parameters');
fprintf(fidOut,'%40s\n',DepthFlag);
fprintf(fidOut,Form4,'Approach direction (Alpha)','=',Alphad,'CCW
degrees from RLD');
fprintf(fidOut,Form4,'Phase offset (Phi)','=',Phid,'degrees');
fprintf(fidOut,Form4,'Wave height','=',hwave,'meters');
fprintf(fidOut,Form4,'Wave amplitude','=',A,'meters');
fprintf(fidOut,Form4,'Water depth','=',d,'meters');
fprintf(fidOut,Form4,'Wave period','=',T,'sec');
fprintf(fidOut,Form4,'Wave frequency','=',F,'sec');
fprintf(fidOut,Form4,'Wave velocity','=',C,'m/s');
fprintf(fidOut,Form4,'Wave length','=',L,'meters');
fprintf(fidOut,Form4,'Wave velocity (shallow)','=',Csh,'m/s');
fprintf(fidOut,Form4,'Wave length (shallow)','=',Lsh,'meters');
fprintf(fidOut,'%s\n',' ');
fprintf(fidOut,'%s\n','    Syd       Sxd       Syp       Sxp');

% Output heights and slopes to ASCII file
Form5 = '%9.6f %9.6f %9.6f %9.6f\n';
for Cnt = 1:length(Syd)
     fprintf(fidOut,Form5,Syd(Cnt),Sxd(Cnt),Syp(Cnt),Sxp(Cnt));
end

CloseFile(fidOut,FileName);
```

SampGen.m

```
function [IFSamp,NewIF] = SampGen(NSmat,IFreq,Ts,fD);

% [IFSamp] = SampGen(IFSamp,NSmat,IFreq,Ts,fD);
% Fills the IF sample array with sampled values of IF array
frequencies,
% accounting for Doppler shift

%   INPUTS
%       IFSamp  IF sample array [n x m], initially all zeros
%       NSmat   matrix indicating how many DSP samples are in each
sweep
%       IFreq   IF array computed from range values
%       Ts      time between DSP samples
%       fD      Doppler shift (+ = approaching, - = receding)
%   OUTPUTS
%       IFSamp  filled IFSamp array to return


[r,c] = size(IFreq);

% Generate vector of IF +/- fD
NewIF = ones(2*r,1);
J = 1;  % index for NewIF
for K = 1:r
    L = mod(K-1,3) + 1;
    NewIF(J) = IFreq(K,L) + fD;
    J = J + 1;
    NewIF(J) = IFreq(K,L) - fD;
    J = J + 1;
end

r = length(NewIF);

% create DSP sample array
NSweeps = size(NSmat,1);
Nsmax = max(NSmat(:,3));
IFSamp = zeros(NSweeps,Nsmax);
% sample all the sweeps
for Cnt = 1:NSweeps
    t = Ts:Ts:Ts*NSmat(Cnt,3);          % time series
    % U/D multiplier [+1 for Upsweep, -1 for Downsweep]
    UDmult = -sign(NSmat(Cnt,2)-0.5);
    %Beam = NSmat(Cnt,1) + 1;
    TempIF = UDmult*(cos(2.*pi.*NewIF(Cnt).*t)); %+ PhIF + pi));
    IFSamp(Cnt,1:NSmat(Cnt,3)) = TempIF;      % assign to data
end
```

VSGoutput.m

```
function [HdrWrit,DatWrit] =
VSGoutput(Fs,FBeam,NSmat,Rsamp,OutFileName);

% Output simulated data to binary file
% in same format as VSG produces it

% INPUTS
%   Fs      DSP sampling frequency      [1,1]
%   FBeam   Beam sweep frequenc(ies)    [1,1]
%   NSmat   Indicator matrix containing columns [Nsweeps,3]:
%               Beam #    Up/Down sweep     # DSP samples
%   Rsamp   Sample data                 [Nsweeps,Ns]
% OUTPUT
%   data to binary file:
%     HEADER
%       Original        [float32]
%       Month           [uint4]
%       Day             [uint4]
%       Hour            [uint4]
%       Minute          [uint4]
%       FSweep          [float32 X 3]
%       Fs              [float32]
%     DATA
%       Data            [uint16 X NSamples]

Orig = 1;
C = datevec(now);
Time = zeros(1,4);
for Cnt = 1:4
    Time(Cnt) = C(Cnt + 1);
end
FBeam = [FBeam,FBeam,FBeam];

% OPEN BINARY FILE FOR OUTPUT
%OutFileName = 'U:\OCEAN\Wave Simulation\VSGout.bin';
[fidBin, Success] = OpenOutFile(OutFileName);

% WRITE HEADER INFORMATION
% write continuation indicator as unsigned 32 bit int
Numwrit = fwrite(fidBin, Orig, 'uint32');
HdrWrit = Numwrit;

% write 4 Time Codes
for Cnt = 1:4
   Numwrit = fwrite(fidBin, Time(Cnt), 'uint8');
   HdrWrit =HdrWrit + Numwrit;
end

% write 3 Beam Frequencies
for Cnt = 1:3
   Numwrit = fwrite(fidBin, FBeam(Cnt), 'float32');
   HdrWrit = HdrWrit + Numwrit;
end

% write Sampling Frequency
Numwrit = fwrite(fidBin, Fs, 'float32');
HdrWrit = HdrWrit + Numwrit;
DatWrit = 0;

% CREATE DATA
Cnt = 0;
NSweeps = size(NSmat,1);
```

```
for SwCnt = 1:NSweeps
   Beam = bitshift(NSmat(SwCnt,1),13);
   UD = bitshift(NSmat(SwCnt,2),12);
   Word = zeros(1,size(Rsamp,2));
   %Beam = bitshift(Beam,13);
   %UD = bitshift(UD,12);
   Ns = NSmat(SwCnt,3);

   for NsCnt = 1:Ns
      Data = Rsamp(SwCnt,NsCnt);
      Word(NsCnt) = Data + UD + Beam;
      %WordStr = dec2bin(Word(NsCnt),16);
      Cnt = 1 + Cnt;
      %disp(sprintf('%12i %16i %18s',Cnt,Word(NsCnt),WordStr))
   end

   % WRITE DATA
   NumWrit = fwrite(fidBin, Word(1:Ns), 'uint16');
   DatWrit = DatWrit + NumWrit;
end

CloseFile(fidBin, OutFileName);
```

NsperSweep.m

```
function [N] = NsperSweep(TBeam,Ts,TObs);

% create matrix containing vectors:
%   Column 1:   Sweep number [1,2,3]
%   Column 2:   Up/Down [0 = up, 1 = down]
%   Column 3:   Number of DSP samples in each sweep

Ns = ceil(TBeam/2/Ts) + 1;         % max # of samples in a sweep
NSweeps = floor(TObs/TBeam);       % # of complete sweeps
NSamples = floor(TObs/Ts);          % total # of DSP samples

N = zeros(NSweeps,3);
T = Ts;
Beam = 0;
UD = 0;
for Cnt = 1:NSweeps
   N(Cnt,1) = mod(Beam,3);
   N(Cnt,2) = mod(UD,2);
   Temp =  T - Cnt*TBeam/2;        % test value
      while Temp <= 0
        N(Cnt,3) = N(Cnt,3) + 1;    % increment sample count
        T = T + Ts;                 % next DSP sample
        Temp = T - Cnt*TBeam/2;    % test value
      end
   UD = UD + 1;
   if mod(UD,2) == 0
       Beam = Beam + 1;
   end
end
```

Part 2 – DSP Analysis Files

DSP2.m

```
function [SlopeXY,R,V,BeamData,Fs,FBeam,PSD,F] =
DSP2(InFileName,Toffset,M,BW,Theta,HVSG);

%   DSP of Ocean Project data
%
%   INPUTS
%       InFileName  VSG binary filename
%       Toffset     Time position in file to begin reading (sec)
%       M           Number of cycles to average
%       BW          IF bandwidth (MHz)
%       Theta       Incidence angle (deg)
%   OUTPUTS
%       R           Range (m)
%       V           Velocity (m/s)
%       BeamData    DSP sample data for 3 beams, Up/Down, M cycles, N
%                       samples each [M x 6 x N]
%       Fs          DSP sample frequency (Hz)
%       FBeam       Beam sweep repetition frequency (1 beam)(Hz)
%       PSD         power spectral density calculated from samples
%                      (relative units)
%       F           frequency abscissa for PSD (Hz)


%   CONSTANTS
c = 2.998*10^8;     % speed of light
f0 = 34.75e9;       % radar operating frequency
Lambda = c/f0;      % radar operating wavelength
dr = pi/180;        % conversion, deg to radians
Delta = 0;          % rotation angle about z' axis
Theta = Theta*dr;

BW = BW*1e6;

% Read Data to Analyze
[BeamData,Fs,FBeam] = GetTimeSeries2(InFileName,Toffset,M);
% Make BeamData symmetric about zero
BeamData = BeamData - 2^11;

% Compute size of PSD
DataSize = size(BeamData);
%DataSize(3)
Nexpt  = 1;
N = 2^Nexpt;
while N < DataSize(3)
    Nexpt = Nexpt + 1;
    N = 2^Nexpt;
end

% Initialize arrays for PSD results
PSD = zeros(M,6,N/2);   % PSD results
F = zeros(6,N/2);       % PSD frequency scale
FreqBuff = zeros(M,6);  % effective frequency buffer
%x = zeros(DataSize(3),1);
for MCnt = 1:M
    for BUDCnt = 1:6
        x = reshape(BeamData(MCnt,BUDCnt,:),[DataSize(3),1]);
        % Get PSD of one sweep vector
        [y, z] = FindPSD(x, N, Fs);
        PSD(MCnt,BUDCnt,:) = y;
```

```
        F(BUDCnt,:) = z;
        y = FindEffFreq(x, N, Fs);
        FreqBuff(MCnt,BUDCnt) = y;  % frequency buffer [M x 6]
    end
end

% TEMP
PSDSize = size(PSD);
x = reshape(F(6,:),[1,PSDSize(3)]);
y = reshape(PSD(1,6,:),[1,PSDSize(3)]);
stem(x,y)
% TEMP

% Average frequencies for each beam/sweep
AvgFreq = mean(FreqBuff);
FreqBuff
AvgFreq

% Compute frequencies for range and Doppler
FRange = zeros(1,3);
FDoppler = zeros(1,3);
for J = 1:3,
    FRange(J) = (AvgFreq(2*J) + AvgFreq(2*J-1))/2;
    FDoppler(J) =  (AvgFreq(2*J-1) - AvgFreq(2*J))/2;
end

% Convert to range
R = FRange.*(c/(4*BW*FBeam));
% Convert to velocity
V = FDoppler.*(c/2/f0);


%[SlopeXY] = slopexymod(R,Theta,Delta)
[SlopeXY] = evanslope(R,Theta,HVSG)
```

GetTimeSeries2.m

```
function [BeamData,Fs,FB] = GetTimeSeries2(InFileName,Toffset,M);
% Obtain one time series of raw VSG data from input file.
% Sort into array, pad with zeros as needed.
% Pass to analysis program.
%   INPUTS
%       InFileName  Binary source data filename
%       Toffset     Desired Time offset from beginning of file (sec)
%       M           Desired number of [beam1,beam2,beam3] cycles
%                       after offset
%   OUTPUTS
%       BeamData    3-D array in this form: BeamData(Cycle,BUD,Datum)
%                   where   Cycle = which cycle number [1,2,...M]
%                             BUD = Beam/Up/Down number [1,2,...6]
%                           Datum = Individual raw data item
%   FUNCTION CALLS
%       OpenInFile.m
%       GetHeader.m
%       GetFileSize.m
%       FindFirstDatum.m
%       FindLastDatum.m
%       CloseFile.m
%
% CONSTANTS
global MaskBUD MaskData MaskBeam MaskUD It HdrSize
MaskBUD = 28672;        % to extract Beam and Up/Down bits from word
```

```
MaskData = 4095;        % to extract data value from word
MaskBeam = 24576;       % to extract Beam Number from word
MaskUD = 4096;          % to extract Up/Down bit from word
It = 2;                 % search iteration size, bytes/word
%HdrSize = 2;           % header size (bytes)

% ROUTINE
% GET HEADER INFO
% Open input file
[fid, Success] = OpenInFile(InFileName);
% Read header information
[Original, Time, FBeam, Fs] = GetHeader(fid);
HdrSize = ftell(fid);
FB = FBeam(1);

% Initialize beam data array
Ns = ceil(Fs/2/FBeam(1));
BeamData = zeros(M,6,Ns);

% Test time offset against file size, issue warning
FileSize = GetFileSize(fid);       % get file size (bytes)
TFile = FileSize/2 / Fs;           % compute file size (sec)
SamplesOffset = fix(Toffset*Fs);   % # words in Toffset
BytesOffset = It*SamplesOffset;    % set estimate # of bytes
                                   % It = bytes/word
                                   % TotalSamples = samples desired
if BytesOffset >= FileSize
    disp(sprintf('    %s %8.3f sec','Toffset too long: File length
=',TFile))
end

% Estimate Bytes2Read (number of bytes to read) = 2*N
Ns = Fs/2/FBeam(1); %+ 1;         % avg # of samples in a sweep
% set estimated # of bytes to read
Bytes2Read = It*Ns*M*6 %+ mod((It*Ns*M*6),2)
Bytes2Read = 2*round(Bytes2Read/2) % force multiple of 2
                                  % It bytes/word
                                  % Ns words/sweep
                                  % 6 sweeps/cycle
                                  % M cycles desired

% Find beginning and ending file pointers
FPtrBegin = FindFirstDatum1(fid,BytesOffset)
FPtrEnd   = FindLastDatum1(fid,FPtrBegin,Bytes2Read)

N = (FPtrEnd - FPtrBegin + It)/It;   % # of words to read

fseek(fid, FPtrBegin, -1);      % set file to beginning of desired data
[RawVSGData, NumRead] = fread(fid, N, 'uint16');  % read data
disp(sprintf('  %8i Items read out of %8i required\n',NumRead,N))

% Close input file
[Success] = CloseFile(fid, InFileName);

% EXTRACT BEAM/U/D NUMBERS AND DATA, SORT INTO OUTPUT ARRAY
MCnt = 1;
PrevBUD = 1;
%BUDCnt = zeros(1,6);
DataCnt = ones(1,6);
for NCnt = 1:N
    % Read bits 12, 13, & 14, ie. 0111000000000000
    BUD = bitshift(bitand(RawVSGData(NCnt),MaskBUD), -12) + 1;
    %BUDCnt(BUD) = BUDCnt(BUD) + 1;
    %while BUDCnt(BUD) > 6
```

```
    %    BUDCnt(BUD) = BUDCnt(BUD) - 6;
    %end
    % Read bits 0 thru 11 for Data, ie. 0000111111111111
    Data = bitand(RawVSGData(NCnt), MaskData);
    % Beam = bitshift(bitand(Word,MaskBeam),-13) + 1;
    % UD = bitshift(bitand(Word,MaskUD),-12);
    if PrevBUD == 6 & BUD == 1
        MCnt = MCnt + 1;
        DataCnt = ones(1,6);
    end
    BeamData(MCnt,BUD,DataCnt(BUD)) = Data;
    DataCnt(BUD) = DataCnt(BUD) + 1;
    PrevBUD = BUD;
end
```

GetHeader.m

```
function [Original, Time, FBeam, FSample] = GetHeader(fid);
% Get header information and place in output variables.
%     Original:  1 = first file in series
%                0 = continuation file
%     Time (int) X 4 = time code of experiment
%     FBeam (float) X 3 = beam frequencies
%     FSample (float) = sampling frequency

Original = fread(fid, 1, 'uint32');   % read continuation indicator
                                        % as unsigned 32 bit int
Time =  fread(fid, 4, 'uint8');              % read 4 time segments as
                                        % unsigned 8 bit ints
FBeam = fread(fid, 3, 'float32');     % read 3 beam frequencies
                                        % as 32 bit floats
FSample =  fread(fid, 1, 'float32');  % read sampling frequency
```

GetFileSize.m

```
function [FileSize] = GetFileSize(fid);

% go to end of file, get file size, rewind file
Success = fseek(fid,0,1);
if Success ~= 0
    ErrMsg = ferror(fid)
    N = -1;
    return
end
FileSize = ftell(fid);
frewind(fid);
```

FindFirstDatum1.m

```
function [FPtr] = FindFirstDatum1(fid,BytesOffset);

% Finds file position of ending data for M desired cycles of VSG data
%   INPUTS
%       InFileName  Complete path and input file name
%       Toffset     Time into file to begin read (sec)
%       M           Number of cycles required
%                       (to find beginning byte to read)
%   OUTPUTS
%       N           Number of 16-bit integers to read from file

% CONSTANTS
global HdrSize MaskBUD It

%[fid BytesOffset]

% move file pointer to estimated position
%Success =
fseek(fid, HdrSize + BytesOffset, -1);
%if Success ~= 0
%    ErrMsg = ferror(fid)
%    N = -1;
%    return
%end

% test contents, determine Beam/U/D position
[Word, NumRead] = fread(fid, 1, 'uint16');
Vector = bitshift(bitand(Word,MaskBUD), -12) + 1;
Cnt = 0;                    % byte shift counter
%BinDir = sign(Vector-3.5);        % search direction

% Look backward if in Beam 1 Up (BUD = 1)
while Vector == 1
    fseek(fid,-2*It,0);            % back to previous word
    [Word, NumRead] = fread(fid, 1, 'uint16');
    Vector = bitshift(bitand(Word,MaskBUD), -12) + 1;
    FPtr = ftell(fid);      % beginning file pointer
    Cnt = Cnt - It;
end
% Look forward if not in Beam 1 Up (BUD ~= 1)
while Vector ~= 1
    [Word, NumRead] = fread(fid, 1, 'uint16');
    Vector = bitshift(bitand(Word,MaskBUD), -12) + 1;
    FPtr = ftell(fid) - It;      % beginning file pointer
    Cnt = Cnt + It;
end
```

FindLastDatum1.m

```
function [FPtr] = FindLastDatum1(fid,BytesOffset,Bytes2Read);

% CONSTANTS
global HdrSize MaskBUD It

%[fid BytesOffset Bytes2Read HdrSize]

% move file pointer to estimated position
Success = fseek(fid,(BytesOffset+Bytes2Read+HdrSize),-1);

%ftell(fid)

if Success ~= 0
    ErrMsg = ferror(fid)
    N = -1;
    return
end

% test contents, determine Beam/U/D position
[Word, NumRead] = fread(fid, 1, 'uint16');
Vector = bitshift(bitand(Word,MaskBUD), -12) + 1;
Cnt = 2;                    % byte shift counter
%BinDir = sign(Vector-3.5);        % search direction
% Look forward if in Beam 3 Down (BUD = 6)
while Vector == 6 | Vector == 5 | Vector == 4
    [Word, NumRead] = fread(fid, 1, 'uint16');
    Vector = bitshift(bitand(Word,MaskBUD), -12) + 1;
    FPtr = ftell(fid);      % ending file pointer
    Cnt = Cnt + It;
end
% Look backward if not in Beam 3 Down (BUD ~= 6)
while Vector == 1 | Vector == 2 | Vector == 3
    fseek(fid,-2*It,0);           % back to previous word
    [Word, NumRead] = fread(fid, 1, 'uint16');
    Vector = bitshift(bitand(Word,MaskBUD), -12) + 1;
    FPtr = ftell(fid) - It;      % ending file pointer
    Cnt = Cnt - It;
End
```

OpenInFile.m

```
function [fidIn, Success] = OpenInFile(InFileName)

fidIn = fopen(InFileName, 'r');
if fidIn > 2
   Success = 1;
   fprintf('\n\n     File %s Opened Successfully for read\n',
InFileName);
else
   Success = 0;
   fprintf('\n\n     *** UNABLE TO OPEN FILE %s ***\n\n',...
      InFileName);
   return
end
```

CloseFile.m

```
function [Success] = CloseFile(fid, FileName);
% User Inputs:    fid - file identifier of the file to close (int)
%           FileName - Filename of the file to close (string)
% Output:   Success - 1 = successful close (bin int)
%                   0 = unsuccessful close
%
% Module closes the indicated file and gives message as to success of
% close.  In addition, it returns a binary indicator of success or
% failure.

CloseErr = fclose(fid);
if ~CloseErr
   Success = 1;
   fprintf('\n     File %s Closed Successfully\n\n\n', FileName);
else
   Success = 0;
   fprintf('\n\n            *** UNABLE TO CLOSE FILE %s ***\n\n',...
      FileName);
end
```

Part 3 - Other Authors' Routines

FindPSD.m

```
%*********************************************************************
******
%  Program:     FindPSD.m
%  Version:     1.0
%  Type:        MATLAB
%  Date:        5/18/98
%  Programmer:  Gary W. Hamilton II
%
%  Syntax:      [PSD, F] = FindPSD(x, N, Fsampling)
%
%  Function:    To calculate the approximate power spectral density of
a
%               sweep vector using a FFT^2 algorithm
%
%  Input:       Sweep vector, FFT length, and sampling frequency
%
%  Output:      Approximate power spectral density and corresponding
%          discrete frequency vector
%
%  Fun. Calls:  none
%
%  Variable Definition
%  -------------------
%  x            Sweep vector
%  N            FFT length
%  Fsampling    Sampling frequency (in Hz)
%*********************************************************************
******

%  Define function
function [PSD, F] = FindPSD(x, N, Fsampling)

%  Convert PSD sample index to positive continuous frequency domain
F = [0:N/2-1]*Fsampling/N;

%  Calculate FFT of sweep vector
%length(x)   % TEMP
%size(x)     % TEMP

x = x.*hanning(length(x));
X = abs(fft(x, N));
FFT = X(1:N/2)/N;

%  Calculate PSD of sweep vector
PSD = (FFT.^2)';
```

FindEffFreq.m

```
%*********************************************************************
******
%  Program: FindEffFreq.m
%  Version: 3.0
%  Type:    MATLAB
%  Date:    5/19/98
%  Programmer:    Gary W. Hamilton II
%
%  Syntax:  EffFreq = FindEffFreq(x, N, Fsampling)
%
%  Function:      To calculate the effective frequency of the VSG
return
%           signal using a FFT^2 algorithm
%
%  Input:       Sweep vector, FFT length, and sampling frequency
%
%  Output:  Effective frequency of VSG return signal
%
%  Fun. Calls:    FindPSD
%
%  Variable Definition
%  -------------------
%  x        Sweep vector
%  N        FFT length
%  Fsampling      Sampling frequency (in Hz)
%*********************************************************************
******

%  Define function
function EffFreq = FindEffFreq(x, N, Fsampling);

%  Local variable definitions
%  dF       Discrete frequency vector spacing (in Hz)
%  PSD            Power spectrum of sweep vector (in Watts)
%  Numerator      Numerator integral of effective frequency calculation
%  Denominator    Denominator integral of effective frequency
calculation
%  EffFreq  Effective frequency of sweep vector

%  Declare and initialize local variables
Numerator = 0; Denominator = 0; EffFreq = 0; dF = 0; PSD = 0;

%  Calculate approximate power spectral density
[PSD, F] = FindPSD(x, N, Fsampling);

%  Calculate discrete frequency vector spacing
dF = F(2) - F(1);

%  Calculate the effective frequency
Numerator = trapz((F.^2).*PSD) * dF;
Denominator = trapz(PSD)*dF;
EffFreq = sqrt(Numerator / Denominator);
```

slrangemod.m

```
function [ran,x,y,z] = slrangemod(theta,delta,r,phi,T,A,h,FBeam,TObs);
% Format:
% [ran,x,y,z] =
slrangemod(theta,delta,beta12,beta23,gamma,r,phi,f,A,h);
% Determines range to ocean surface from radar along the three beams
% taking into account all possible angles.  This program solves the
% nonlinear equation that arises from finding the point of intersection
of
```

```
% the radar beam with the ocean surface.

% Since there may be multiple solutions to the nonlinear equation, it
is
% best to guess the most accurate length of r, while making sure that
the
% guess is shorter than the resulting solution.

%   INPUTS
%       theta   incidence angle (rotation of radar around x axis)
%       delta   rotation angle (rotation of radar around z' axis)
%       r       initial guesses of lengths of beams 1,2,and 3 (meters)
%       phi     angle between RLD and direction from which waves
approach
%               (deg)
%       T       period of ocean waves (sec)
%       A       amplitude of ocean waves (meters)
%       h       height of radar from mean sea surface (meters)
%   OUTPUTS
%       ran
%       x
%       y
%       z

% Chris Evans, RSL, University of Kansas, March 1, 1994
% Adapted from original by Evan Bryson, 12/2003
%   Inserted hard-wired antenna angles from Akins/Legarsky VSG work
%   Inserted h (VSG height) and A (ocean wave amplitude) in input
%   Modified from f to T (using period of ocean wave, rather than
%       frequency)
%   Modified rate of sampling to FBeam rate, instead of 0.1 sec
%   Modified length of time observed to TObs, instead of 1 wave period

beta12 = 3.53;
beta23 = 3.34;
gamma = 90;

dr = pi/180;
%theta = theta*dr;
%delta = delta*dr;
beta12 = beta12*dr;
beta23 = beta23*dr;
gamma = gamma*dr;
%phi = phi*dr;

% h = 20;
% A = 5.0;
f = 1/T;

R11 = cos(delta)*sin(beta12)*cos(gamma-pi/2) +...
    sin(delta)*sin(beta12)*sin(gamma-pi/2);
R21 = sin(delta)*cos(theta)*sin(beta12)*cos(gamma-pi/2) -...
    cos(delta)*cos(theta)*sin(beta12)*sin(gamma-pi/2) +...
    sin(theta)*cos(beta12);
R31 = sin(delta)*sin(theta)*sin(beta12)*cos(gamma-pi/2) -...
    cos(delta)*sin(theta)*sin(beta12)*sin(gamma-pi/2) - ...
    cos(theta)*cos(beta12);
R12 = 0;
R22 = sin(theta);
R32 = -cos(theta);
R13 = -sin(delta)*sin(beta23);
R23 = cos(delta)*cos(theta)*sin(beta23) +...
    sin(theta)*cos(beta23);
R33 = cos(delta)*sin(theta)*sin(beta23) -...
```

```
    cos(theta)*cos(beta23);

k = ((2*pi*f)^2)/9.8;   % spatial frequency of waves (wave number)

% initial guesses
r1(1) = 0;  r1(2) = r(1);
r2(1) = 0;  r2(2) = r(2);
r3(1) = 0;  r3(2) = r(3);

for t = 1:FBeam*TObs       % t is in 1/FBeam second increments
                        % FBeam*T gives 1 period of samples at rate
FBeam
    J = 2;
    %disp(sprintf(' %i4 ',t))        % *****TEMP*******
    while (abs(r1(J)-r1(J-1)) > 1e-13) | (abs(r3(J)-r3(J-1)) > 1e-13),
        % error must be less than this number-----------------^^^^^
        % ranges from equations
        x1(J) = r1(J)*R11;  x2(J) = 0;          x3(J) = r3(J)*R13;
        y1(J) = r1(J)*R21;  y2(J) = r2(J)*R22;  y3(J)=r3(J)*R23;
        z1(J) = r1(J)*R31;  z2(J) = r2(J)*R32;  z3(J)=r3(J)*R33;
        % ocean wave equations
        zeta1(J) = 2*pi*f*t/FBeam + k*sin(phi)*x1(J) +
k*cos(phi)*y1(J);
        zeta2(J) = 2*pi*f*t/FBeam + k*sin(phi)*x2(J) +
k*cos(phi)*y2(J);
        zeta3(J) = 2*pi*f*t/FBeam + k*sin(phi)*x3(J) +
k*cos(phi)*y3(J);
        sea1(J) = A*cos(zeta1(J)) - h;
        sea2(J) = A*cos(zeta2(J)) - h;
        sea3(J) = A*cos(zeta3(J)) - h;
        F1(J) = z1(J) - sea1(J);
        F2(J) = z2(J) - sea2(J);
        F3(J) = z3(J) - sea3(J);
        % derivatives for slope of ocean wave
        Fder1(J) = R31 + A*sin(zeta1(J))*(k*sin(phi)*R11 +
k*cos(phi)*R21);
        Fder2(J) = R32 + A*sin(zeta2(J))*(k*sin(phi)*R12 +
k*cos(phi)*R22);
        Fder3(J) = R33 + A*sin(zeta3(J))*(k*sin(phi)*R13 +
k*cos(phi)*R23);
        % Newton's equation
        r1(J+1) = r1(J) - F1(J)/Fder1(J);
        r2(J+1) = r2(J) - F2(J)/Fder2(J);
        r3(J+1) = r3(J) - F3(J)/Fder3(J);
        J = J + 1;
        %disp(sprintf(' %7i %18.14f %18.14f',J-1,r1(J),r3(J)))    %
*****TEMP*******
    end
    ran(t,:) = [r1(J), r2(J), r3(J)];
    x(t,:) = [x1(J-1), x2(J-1), x3(J-1)];
    y(t,:) = [y1(J-1), y2(J-1), y3(J-1)];
    z(t,:) = [z1(J-1), z2(J-1), z3(J-1)];
    if mod(t,100)==0
        [t J]
    end
end
```

slopepoimod.m

```
function [Sx,Sy] = slopepoimod(ran,theta,delta,A,phi,T,FBeam);
%
% [Sx,Sy] = slopepoi(ran,theta,delta,beta12,beta23,gamma,A,phi,f);
% calculates the slope at the point for which the slope is being
% approximated with slopeab.m
%
%   INPUTS
%       ran     time series of ranges [nx3]
%       theta   angle of incidence (deg)
%       delta   angle of CCW rotation of radar about z' axis
%       beta12  angle between beams 1 and 2
%       beta23  angle between beams 2 and 3
%       gamma   flat angle between beam21 and beam23 lines
%       A       ocean wave amplitude
%       phi     angle of wave approach direction relative to RLD
%       T       period of ocean wave
%       f       frequency of ocean wave = 1/T
%
%   OUTPUTS
%       Sy      slope in y direction (deg)
%       Sx      slope in x direction (deg)
%
% Chris Evans, RSL, The University of Kansas, March 10, 1994
% Adapted from original by Evan Bryson, 12/2003
%   Modified from f to T (using period of ocean wave, rather than
%       frequency)
%   Modified rate of sampling to FBeam rate, instead of 0.1 sec
%   Modified length of time observed to TObs, instead of 1 wave period
%
beta12 = 3.53;
beta23 = 3.34;
gamma = 90;

dr = pi/180;
%theta = theta*dr;
%delta = delta*dr;
beta12 = beta12*dr;
beta23 = beta23*dr;
gamma = gamma*dr;
%phi = phi*dr;

f = 1/T;

[r,c]= size(ran);

% coordinates in the antenna system
x1a = ran(:,1)*sin(beta12)*cos(pi/2-gamma);
y1a = ran(:,1)*sin(beta12)*sin(pi/2-gamma);
z1a = -ran(:,1)*cos(beta12);
x2a = zeros(r,1);
y2a = zeros(r,1);
z2a = -ran(:,2);
x3a = zeros(r,1);
y3a = ran(:,3)*sin(beta23);
z3a = -ran(:,3)*cos(beta23);

% rotation transforms from the antenna to the earth coord system
rot = [cos(delta),         -sin(delta),            0;
    cos(theta)*sin(delta), cos(delta)*cos(theta), -sin(theta);
    sin(theta)*sin(delta), cos(delta)*sin(theta), cos(theta)];

for t=1:r,
```

```
    Ant = [x1a(t), x2a(t), x3a(t);
        y1a(t), y2a(t), y3a(t);
        z1a(t), z2a(t), z3a(t)];

    E = rot * Ant;
    x1e(t) = E(1,1);    x2e(t) = E(1,2);     x3e(t) = E(1,3);
    y1e(t) = E(2,1);    y2e(t) = E(2,2);     y3e(t) = E(2,3);
    z1e(t) = E(3,1);    z2e(t) = E(3,2);     z3e(t) = E(3,3);

    % coordinates are now in the earth system
    A1 = (y1e(t) - y2e(t))*(z3e(t) - z2e(t)) - (y3e(t) -
y2e(t))*(z1e(t) - z2e(t));
    B1 = (x3e(t) - x2e(t))*(z1e(t) - z2e(t)) - (x1e(t) -
x2e(t))*(z3e(t) - z2e(t));
    C1 = (x1e(t) - x2e(t))*(y3e(t) - y2e(t)) - (x3e(t) -
x2e(t))*(y1e(t) - y2e(t));
    A2 = x3e(t) - x2e(t);
    B2 = y3e(t) - y2e(t);
    C2 = z3e(t) - z2e(t);
    A3 = x1e(t) - x2e(t);
    B3 = y1e(t) - y2e(t);
    C3 = z1e(t) - z2e(t);

    R = [A1, B1, C1;
        A2, B2, C2;
        A3, B3, C3];

    D1 = A1*x2e(t) + B1*y2e(t) + C1*z2e(t);
    D2 = 0.5*(x3e(t)^2 + y3e(t)^2 + z3e(t)^2 - x2e(t)^2 - y2e(t)^2 -
z2e(t)^2);
    D3 = 0.5*(x1e(t)^2 + y1e(t)^2 + z1e(t)^2 - x2e(t)^2 - y2e(t)^2 -
z2e(t)^2);

    S = [D1; D2; D3];

    % x and y positions of beam spots
    Q = inv(R)*S;
    Px = Q(1);
    Py = Q(2);

    % ocean wave equations for slopes (-sin is derivative of cos)
    % changed from original program (it had +cos, which is d(sin())/dr
    k = (2*pi*f)^2/9.8;
    Sy(t) = -atan(A*k*cos(phi)*sin(2*pi*f*t/FBeam + k*sin(phi)*Px +...
        k*cos(phi)*Py)) * 180/pi;   % degrees
    Sx(t) = -atan(A*k*sin(phi)*sin(2*pi*f*t/FBeam + k*sin(phi)*Px +...
        k*cos(phi)*Py)) * 180/pi;   % degrees
end
Sy = Sy';
Sx = Sx';
```

slopeab2.m

```
function [Sy,Sx] = slopeab2(x,y,z);
%
%
% Slopeab2 determines the slope time series from the plane given the x,
y,
% and z coordinates of the points of intersection of the radar beams
with
% the ocean surface.
%
%   INPUTS
```

```
%       x   x coordinate of the point of intersection [nx3]
%       y   y coordinate of the point of intersection [nx3]
%       z   z coordinate of the point of intersection [nx3]

%   OUTPUTS
%       Sy  time series of slope in y (RLD) direction (deg)
%       Sx  time series of slope in x (cross) direction (deg)

%   Chris Evans, RSL, The University of Kansas, March 10, 1994

%   B & C are vectors in the plane, and N is the normal to the plane.
B = [(x(:,1) - x(:,2)), (y(:,1) - y(:,2)), (z(:,1) - z(:,2))];
C = [(x(:,3) - x(:,2)), (y(:,3) - y(:,2)), (z(:,3) - z(:,2))];

%   Normal components are found from the cross product of B and C
Nx = B(:,2).*C(:,3) - C(:,2).*B(:,3);
Ny = B(:,3).*C(:,1) - C(:,3).*B(:,1);
Nz = B(:,1).*C(:,2) - C(:,1).*B(:,2);

% The slope of the plane is the negative inverse of the slope of the
line.
Sy = 180/pi*atan(-Ny./Nz);
Sx = 180/pi*atan(-Nx./Nz);
```

evanslope.m

```
function [SlopeXY] = evanslope(R,Theta,HVSG);

% From Chris Evans, June 1994
% compute slope from measured VSG ranges
% INPUTS
%   R       ranges from VSG to ocean surface [beam1 beam2 beam3]
%   Theta   incidence angle (rad)
%   HVSG    vertical height of VSG from mean ocean surface
% OUTPUT
%   SlopeXY slope of ocean surface [Sx Sy] (deg)

% CONSTANTS
dr = pi/180;
b12 = 3.53;     b12 = b12*dr;
b23 = 3.34;     b23 = b23*dr;
g = 90;         g = g*dr;       %
d = 0;          d = d*dr;       % rotation on z' axis
t = Theta;
h = HVSG;

% Coordinates in radar coord system
Ar = [R(1)*sin(b12)*cos(pi/2-g);
    R(1)*sin(b12)*sin(pi/2-g);
    -R(1)*cos(b12)];

Br = [0; 0; -R(2)];

Cr = [0; R(3)*sin(b23); -R(3)*cos(b23)];

% Transformation matrix
Tx = [cos(d),        -sin(d),       0;
      sin(d)*cos(t), cos(d)*cos(t), -sin(t);
      sin(d)*sin(t), cos(d)*sin(t), cos(t)];

% Convert to earth coord system
Ae = Tx*Ar;
Be = Tx*Br;
Ce = Tx*Cr;

% add height of VSG
Ae(3) = Ae(3) + h;
Be(3) = Be(3) + h;
Ce(3) = Ce(3) + h;

% Slope vectors
BA = Ae - Be;
BC = Ce - Be;

% Normal vectors
N = cross(BA,BC);
Nx = N(1);
Ny = N(2);
Nz = N(3);

% x and y Slope in degrees
Sx = atan(-Nx/Nz) / dr;
Sy = atan(-Ny/Nz) / dr;
SlopeXY = [Sx Sy];
```

Appendix C

Summary – Gary Hamilton Paper

DSP Algorithm

A DSP algorithm was developed by [Hamilton, 1998] in five stages. Figure C.1 is a block diagram of this algorithm. A brief summary of the algorithm is in order here. The five stages are:

1) Power spectral density
2) Effective frequency
3) Buffer, sum, and difference averaging
4) Range, Doppler velocity, and 3-D slope calculations
5) Beam power calculation

Stage 1 – Power Spectral Density

Module FindPSD.m

This module incorporates the general function for each point of the PSD:

$$PSD(k) = \left|F\{x(i)H(i)\}\right|^2 \text{ where}$$
$$1 \le i \le L$$
$$1 \le k \le \frac{N}{2} + 1$$

and x = sweep vector [1 x L]

H = Hanning window [1 x L]

F = Fourier transform operator

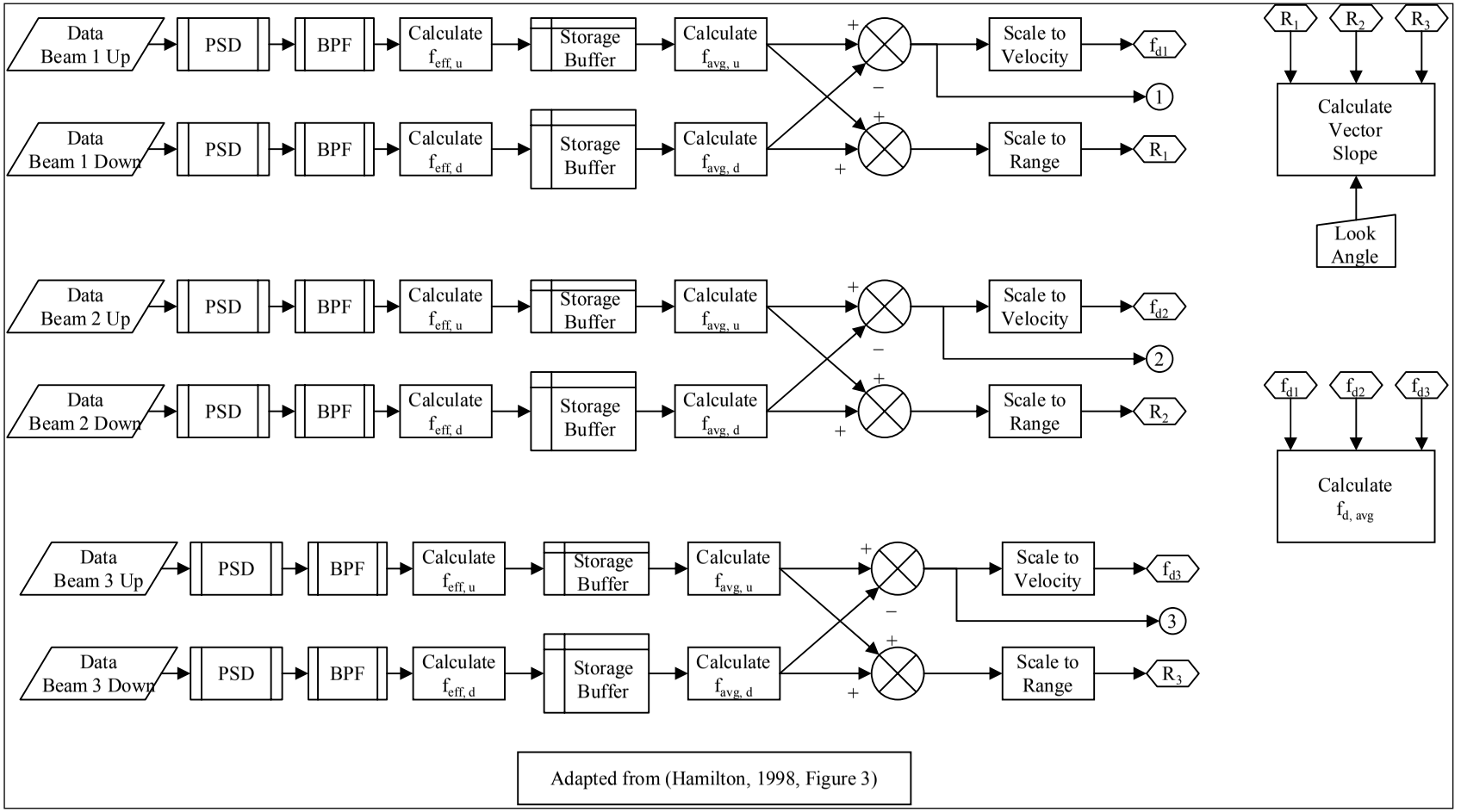

Figure C.1. Block Diagram of VSG Algorithm [Hamilton, 1998]

This module uses *L*, the length of the sweep data vector, as the length of the Hanning window and PSD vectors. Length of the FFT result is *N*, of which only *N*/2 = *L* are used, in order to eliminate redundancy.

Stage 2 – Effective Frequency

This stage calculates and stores the effective frequencies for each of the six sweeps in a cycle.

Module FindEffFreq.m

The effective frequency module is based upon Rice's expected number of zero crossings for normal, stochastic processes [Hamilton, 1998]:

$$f_{\exp} = \sqrt{\frac{\int_{-\infty}^{\infty} f^2 \cdot PSD(f) \cdot df}{\int_{-\infty}^{\infty} PSD(f) \cdot df}}$$

where

$PSD(f)$ = power spectral density

f = frequency(Hz)

For the purposes of VSG analysis, discrete frequencies are summed rather than integrated, limits of the summation are from the lower to the upper cutoff frequencies of the VSG bandpass filter, and discrete frequency spacing replaces *df*:

$$f_{eff} = \sqrt{\frac{\sum_{F_l}^{F_u} k(i)2 \cdot PSD(i) \cdot \Delta k}{\sum_{F_l}^{F_u} PSD(i) \cdot \Delta k}}$$

where

$PSD(i)$ = power spectral density approximation vector [1 x N/2 + 1]

$k(i)$ = Discrete frequency vector [1 x N/2 + 1]

Δk = Discrete frequency vector spacing

F_u = Upper frequency limit

F_l = Lower frequency limit

This module utilizes the "trapz.m" function in MATLAB®, which performs numerical integration using a trapezoidal approximation technique.

Module StoreToBuffer.m

This module appends each effective frequency result to the appropriate one of six data buffers corresponding to the sweep and beam origin of the result.

Stage 3 – Buffer, Sum, and Difference Averaging

Module AverageEffFreq.m

This module computes the average for each frequency buffer according to these equations:

$$\overline{f_{eff-up,j}} = \frac{1}{M}\sum_{i=1}^{M}\{k_{i,j}\} \quad \text{for upsweep}$$

$$\overline{f_{eff-down,j}} = \frac{1}{M}\sum_{i=1}^{M}\{k_{i,j}\} \quad \text{for downsweep}$$

where

M = number of elements in buffer

i = element index {1, 2, …, M}

j = beam j: {1, 2, 3}

In conjunction with calling modules, it averages the contents of each buffer from Stage 2 to obtain six averaged frequency values, one for each sweep of each of the three beams.

Module SumAverage.m

This module produces two results, one an average of all the upsweep frequencies, and the other an average of all the downsweep frequencies. These results are used in DiffAverage.m. They are also used to produce an average overall frequency according to this equation:

$$f_{range,j} = \frac{1}{2}\left(f_{eff-up,j} + f_{eff-down,j}\right)$$

This result is used to estimate the range to the wave.

Module DiffAverage.m

This module computes the difference in the average frequency for each beam according to this equation:

$$f_{Doppler,j} = \frac{1}{2}\left(f_{eff-up,j} - f_{eff-down,j}\right)$$

This is used to estimate Doppler component in the RLD.

Stage 4 – Range, Doppler Velocity, and 3-D Slope Calculation

These three functions were under construction at the time of this writing. Range is determined from estimated frequency using this equation:

$$R_j = \frac{c \cdot f_{range,j}}{4 \cdot BW \cdot f_{sweep}}$$

where

$$j = \text{Beam Identifier} : \{1,2,3\}$$
$$f_{range,j} = \text{Range frequency of beam } j$$
$$f_{sweep} = \text{Sweep frequency of VSG system}$$
$$c = \text{Speed of light}$$
$$BW = \text{Bandwidth of VSG system}$$

Slope is calculated from the three measured range values and VSG orientation using trigonometry. Doppler velocity component toward the radar is measured in stage 3 (See Figure C.2). Orbital velocity is the desired measurement, and it may be extracted by using trigonometry. Group velocity of ocean waves may be obtained through a number of measurements of ocean wave position. (Note that ocean group velocity, vprop, will not appear in actual radar measurements [Moore, 2003].

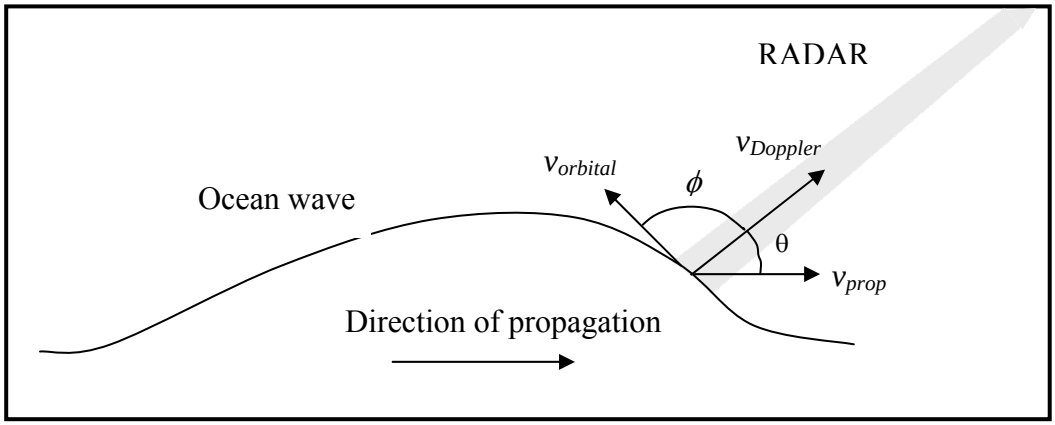

Figure C.2. Doppler Vector Resolution [Hamilton, 1998]

Stage 5- Average Beam Power

Each of the three VSG beams transmits with a slightly different beam power. Beam power is of interest in the measurement of slope modulation. The final stage of the VSG will attempt to calculate the instantaneous average power of each beam and the average of all three beams. Total average power is given by an average of the three independent beam averages:

$$P_{avg,total} = \sum_{j=1}^{3} P_{avg,j}$$

where

$$P_{avg,j} = \text{Average power of beam } j : \{1,2,3\}$$

REFERENCES

Frequency Estimation

Banjanin, Zoran; Zrnic', Dusan S.; and Cruz, J. R. "A Linear Prediction Approach to Doppler Mean Frequency Retrieval in the Presence of Ground Clutter." IEEE Transactions on Aerospace and Electronic Systems 29:3 (July 1993): 1050-58.

Boashash, Boualem. "Estimating and Interpreting The Instantaneous Frequency of a Signal – Part 1: Fundamentals." Proceedings of the IEEE 80:4 (April 1992): 520-38.

Boashash, Boualem. "Estimating and Interpreting The Instantaneous Frequency of a Signal – Part 2: Algorithms and Applications." Proceedings of the IEEE 80:4 (April 1992): 520-38.

Fitz, Michael P. "Further Results in the Fast Estimation of a Single Frequency." IEEE Transactions on Communications 42:2/3/4 (February/March/April 1994): 862-64.

Hocaoglu, A. Koksal and Devaney, Michael J. "Using Bilinear and Quadratic Forms for Frequency Estimation." IEEE Transactions on Instrumentation and Measurement 45:4 (August 1996): 787-92.

Kay, Steven. "A Fast and Accurate Single Frequency Estimator." IEEE Transactions on Acoustics, Speech, and Signal Processing 37:12 (Decemeber 1989): 1987-90.

Lovell, Brian C. and Williamson, Robert C. "The Statistical Performance of Some Instantaneous Frequency Estimators." IEEE Transactions on Signal Processing 40:7 (July 1992): 1708-22.

Lovell, Brian C.; Williamson, Robert C. and Boashash, Boualem. "The Relationship Between Instantaneous Frequency and Time-Frequency Representations." IEEE Transactions on Signal Processing 41:3 (March 1993): 1458-1461.

McIntyre, Mark and Ashley, Anthony. "A Simple Fixed-Lag Algorithm for Tracking Frequency Rate-of-Change." IEEE Transactions on Aerospace and Electronic Systems 29:3 (July 1993): 677-83.

Quinn, Barry G. "Estimation of Frequency, Amplitude, and Phase from the DFT of a Time Series." IEEE Transactions on Signal Processing 45:3 (March 1997): 814-17.

Quinn, B. G. and Hannan, E. J. The Estimation and Tracking of Frequency. Cambridge, U.K.: Cambridge University Press. 2001.

Riezenman, Michael J. "Prolog to 'Estimating and Interpreting The Instantaneous Frequency of a Signal – Part 2: Algorithms and Applications.'" Proceedings of the IEEE 80:4 (April 1992): 539.

Swingler, D. N. "Approximate Bounds on Frequency Estimates for Short Cisoids in Colored Noise." IEEE Transactions on Signal Processing 46:5 (May 1998): 1456-58.

Tretter, Steven A. "Estimating the Frequency of a Noisy Sinusoid by Linear Regression." IEEE Transactions on Information Theory 31:6 (November 1985): 832-35.

Ocean Waves

Bascom, Willard. Waves and Beaches: The Dynamics of the Ocean Surface. Garden City, NY: Anchor Books. 1964.

Kinsman, Blair. Wind Waves: Their Generation and Propagation on the Ocean Surface. Englewood Cliffs, NJ: Prentice-Hall, Inc. 1965.

Kampion, Drew. The Book of Waves: Form and Beauty on the Ocean. Boulder, CO: Roberts Rinehart Publishers. 1997.

Vector Slope Gauge

Evans, Christopher T. "Analysis of a Three-Beam Radar as an Instrument for Determining Ocean Wave Heights and Vector Slopes." RSL Technical Report 8621-5. Lawrence, KS: Radar Systems and Remote Sensing Laboratory, Dept. of Electrical Engineering and Computer Science, University of Kansas. June, 1994.

Haimov, Samuel J. and Moore, Richard K. "Part A: Two-Dimensional Slope Measurement by Vector Slope Gauge, Part B: Coordinate Rotation for the Vector Slope Gauge." RSL Technical Memo 8620-3, Radar Systems and Remote Sensing Laboratory, Dept. of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS. October 1993.

Hamilton, Gary W., II and Moore, Richard K. "DSP Algorithm Design and Development For Ocean Radar VSG System." Unpublished. Report for

Radar Systems and Remote Sensing Laboratory, Lawrence, KS. May 20, 1998.

Hesany, Vahid. "A Radar Vector Slope Gauge for Ocean Measurements." RSL Technical Report 8621-4. Radar Systems and Remote Sensing Laboratory, Dept. of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS. May 1994.

Legarsky, Justin. "An Improved Vector Slope Gauge and C-, Ku-band Radar for Ocean Measurements." Unpublished Master's Thesis. Lawrence, KS: Radar Systems and Remote Sensing Laboratory, Dept. of Electrical Engineering and Computer Science, University of Kansas. December 1995.

Legarsky, J. J. and Moore, R. K. "1995 Test of the Vector Slope Gauge at Duck, North Carolina." RSL Technical Report 8621-8 and 10530-1. Lawrence, KS: Radar Systems and Remote Sensing Laboratory, Dept. of Electrical Engineering and Computer Science, University of Kansas. May 1996.

Moore, R. K. "A Note on Weighted Orthogonal Regression." RSL Technical Memorandum 8620-1. Lawrence, KS: Radar Systems and Remote Sensing Laboratory, University of Kansas Center for Research, Inc. December 1992.

Radar

Farina, A. and Studer, F. A. Radar Data Processing. Vol. 1: Introduction and Tracking. Letchworth, Hertfordshire, Eng.: Research Studies Press, Ltd. 1986.

Farina, A. and Studer, F. A. Radar Data Processing. Vol. 2: Advanced Topics and Applications. Letchworth, Hertfordshire, Eng.: Research Studies Press, Ltd. 1986.

Haykin, Simon and Puthusserypady, Sadasivan. Chaotic Dynamics of Sea Clutter. New York: John Wiley & Sons, Inc. 1999.

Ulaby, Fawwaz T.; Moore, Richard K.; and Fung, Adrian K., Microwave Remote Sensing: Active and Passive, Vol. 2: Radar Remote Sensing and Surface Scattering and Emission Theory. Norwood, Massachusetts: Artech House Inc., 1986, pp. 512-17, 531-38.

Ulaby, Fawwaz T.; Moore, Richard K.; and Fung, Adrian K., Microwave Remote Sensing: Active and Passive, Vol. 3: From Theory to Applications. Norwood, Massachusetts: Artech House Inc., 1986.

DSP

Campa, C.; D'Alessandro, P.; and Rossini, E. "A general purpose processing system based on digital signal processors." Proceedings of the Fourth International Conference on Signal Processing Applications and Technology, p. 1085-91 vol.2. Newton, Massachusetts: DSP Associates, 1993.

Edward, M. N. and Thompson, H.A. "Array processors for real-time radar signal processing." Parallel Computing and Transputer Applications, vol.2, pp. 945-54. Barcelona, Spain: CIMNE, 1992.

Farina, A., ed. Optimised RADAR Processors. London: Peter Peregrinus Ltd., 1987.

Frerking, Marvin E. Digital Signal Processing in Communication Systems. New York: Chapman & Hall, 1994.

Ingle, Vinay K. and Proakis, John G. Digital Signal Processing using MATLAB®. Pacific Grove, CA: Brooks/Cole Publishing Company. 2000.

Kalman, R. E. and Bertram, J. E. "A Unified Approach to the Theory of Sampling Systems." Journal of the Franklin Institute 267 (May 1959): 405-36.

Oppenheim, Alan V. and Schafer, Ronald W. Digital Signal Processing. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.

Rabiner, Lawrence R. and Gold, Bernard. Theory and Application of Digital Signal Processing. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.

Rabiner, Lawrence R. and Helms, Howard D. Literature in Digital Signal Processing: Terminology and Permuted Title Index. New York: The Institute of Electrical and Electronics Engineers, Inc., 1973.

Taylor, Fred and Mellott, Jon. Hands-on Digital Signal Processing. New York: McGraw-Hill. 1998.

Vaidyanathan, P. P. Multirate Systems and Filter Banks. Englewood Cliffs, NJ: P T R Prentice-Hall, Inc. 1993.

Other

Strang, Gilbert. Introduction to Linear Algebra. Wellesley, MA: Wellesley-Cambridge Press. 1993.

The Math Works. MATLAB® Compiler Guide. Natick, MA: The Math Works. 1995.