# AIRBORNE RADAR ALTIMETER RETURN WAVEFORM COMPUTATIONS

by

Michael Hayes Newkirk

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

**MASTER OF SCIENCE**

in

Electrical Engineering

APPROVED:

Gary S Brown
G. S. Brown, Chairman

[illegible]
T. Pratt

W L Stutzman
W. L. Stutzman

August, 1990
Blacksburg, Virginia

LD
5655
V855
1990
N486
c.2

# AIRBORNE RADAR ALTIMETER RETURN WAVEFORM COMPUTATIONS

by

Michael Hayes Newkirk

Committee Chairman: Dr. Gary S. Brown
Electrical Engineering

(ABSTRACT)

Radar altimeter experiments have shown the need for expanding the downward-looking, single-beam system into a multiple beam radar altimeter employing off-nadir altimetry so that information from a wider swath can be obtained from a single overflight. Problems associated with off-nadir altimetry include the effects of pointing angle errors on the return information and difficulty in performing accurate range tracking. In order to understand these problems, investigation of the sensitivity of the average return waveforms to pointing errors is necessary. These waveforms are computed using a convolutional model, including the effects of asymmetric antenna patterns, which is representative of NASA's Multimode Airborne Radar Altimeter. The necessary convolutions are most efficiently performed by a method that uses the fast Fourier transform. The modeled waveforms are then used to devise a method that provides an estimate the pointing angle.

## ACKNOWLEDGMENTS

I would first like to thank my advisor, Dr. Gary Brown for his support and patience while this research was performed. His wisdom is what makes this document understandable. I would also like to thank my committee members, Dr. Pratt and Dr. Stutzman for their helpful suggestions, time and advice.

I also wish to express my gratitude to my parents, Kenneth and Janice, for their never-ending support while I pursued my goals. Most importantly, I thank my wife Deanna, for without her I'd never have made it this far.

# TABLE OF CONTENTS

# 1.0 INTRODUCTION

Radar altimetry has become one of the most important tools available for such tasks as geodetic and topographic mapping, sea state, wind and current measurements and ice sheet monitoring [18]. Radar altimeters were designed primarily to measure the two-way range between the radar and the earth's surface, but the above information can also be extracted from the radar returns. A few of these highly successful instruments include the SKYLAB S-193 [16,19], GEOS-3 [14], SEASAT [2,31] and GEOSAT [13] radar altimeters. Each of these were pulsewidth-limited, downward-looking, single beam radar altimeters mounted aboard low earth orbiting ( $\sim$ 800 km) satellites. Because they employed only one antenna beam, they were limited to observing a very narrow swath directly below the satellite, and mapping ocean currents or land topography required that the satellite make many non-overlapping passes over the area of interest. Normal satellite orbital-precession requires many days to completely cover these areas; therefore, constructing a large scale (several hundred km wide) topographic or geodetic map could take as long as several months to complete [18]. Unfortunately, many of the features that are of interest to oceanographers change too rapidly to wait for the altimeter to make the necessary passes for complete coverage [23,27].

Recent studies [8,15,18,23] have indicated that to further the capabilities of radar altimeters, multiple beam systems must be developed. The use of several narrow antenna beams, one at nadir and the others at various off-nadir pointing angles, would permit a single satellite to observe several narrow swaths on the earth's surface. This would allow for increased coverage area over that of the single beam and single pass system, thereby enabling the more rapid mapping and measurement of large scale oceanic events such as mesoscale eddies [15]. Multiple beamed altimeters would thus greatly improve the capabilities of satellites for numerous remote sensing tasks. However, very little work has been done with multiple beamed radar altimeters.

One of the most difficult problems created by using off-nadir radar altimetry is achieving good range tracking accuracy [23]. All of the previous altimeters operated in the pulsewidth limited mode. This is acceptable for nadir-looking systems, but causes problems for off-nadir geometries. In the pulsewidth limited mode, off-nadir returns are highly smeared representations of the transmitted pulse [23,27], which makes range determination very difficult. To obtain accurate range measurements, a sharp backscattered pulse is necessary [27]. Operating in the beamwidth-limited mode reduces the smearing effect [23] and makes accurate range tracking considerably easier.

A radar is considered to be beam-limited if the transmitted signal duration is long enough that the entire surface enclosed by the beam is illuminated simultaneously [27]. Thus, the return pulse shape is determined primarily by the antenna beamwidth. In this mode of operation, the backscattered pulse is much narrower (in time) than for pulsewidth limited systems, which reduces range ambiguity. It is the aim of this research to model the beamwidth limited backscattered return pulses, analyze these returns, and gain insight into the range tracking possibilities.

Off-nadir altimetry is also more susceptible to pointing angle errors because they produce faulty range measurements and adversely affect geoid and sea state measurements [23]. When the pointing angle is near nadir, a small pointing error has little effect on the return pulse shape, but for larger pointing angles a small pointing error can produce dramatic changes in the return pulse shape and time location. Accurate knowledge of the pointing angle is required in order to have a successful multiple beam radar altimeter, so one of the applications of this study is to determine the sensitivity of the average return waveforms to small pointing errors.

A multiple beam, short-pulse, radar altimeter has been developed at NASA's Goddard Space Flight Center Wallops Flight Facility [27] where it is currently being tested. The Multimode Airborne Radar Altimeter (MARA) employs five narrow antenna beams in its Multibeam Mode of operation; one pointed at nadir, and the other four mounted on two orthogonal arms, pointed forward, aft, and to either side. This beam orientation aids the adaptive tracking studies which are of current interest [27]. The orthogonal feed horn assembly can also be rotated relative to the flight direction such that the five beams will produce equally spaced footprints on the surface. This configuration provides the desired "push-broom" pattern [23] that increases the surface coverage over single beamed systems.

The MARA system also includes the Surface Contour Radar Mode, in which a rotating mirror scans a single beam across the surface in the direction perpendicular to the direction of flight. The third configuration is the Interferometer Mode, which uses two parabolic reflector antennas in place of the lens and horn antennas used in the previous two modes to produce an interferometric antenna pattern. This mode will provide information essential for the development of future interferometric satellite altimeters [27]. All three of these modes use beamwidth limited altimetry.

In order to obtain the very narrow beamwidth required for beamwidth limited altimetry, a 0.86 m diameter dielectric lens, having a focal length of 1.16 m, is illuminated by five horn antennas (in the Multibeam Mode). This results in a symmetrical beamwidth of roughly 0.6° at nadir, but due to an astigmatism introduced by the lens [25], one beamwidth increases to about 0.9° when the pointing angle is 12°, while the other beamwidth remains constant. This effect must be accounted for in the analysis of the average return pulses.

The first step in performing the studies above is to model the average returns for varying pointing angles. The approach used to model these return pulses is the convolutional model first presented by Moore and Williams [20] and more thoroughly examined by Brown [9] and Hayne [12]. This model recognizes that the average return power waveform for near-normal incidence scattering from a rough surface can be obtained from the convolution of three distinct functions. These functions are [9] the radar system point target response, the height probability density function of the specular points on the surface (or the sea state), and a quantity that is a function of the antenna pattern, radar cross section and range to the backscattering point. The latter function is called the average flat surface impulse response (FSIR) [9]. This is a particularly useful model since the point target response and sea state remain constant as the pointing angle varies. The FSIR does not remain constant as the pointing angle changes, so only the FSIR need be recomputed for each pointing angle.

An important difference between previous calculations using this model and the needs dictated by the MARA system is that past work assumed a circularly symmetric antenna pattern [3,9,12]. The unique MARA antenna beam forming system [25,27] requires that the convolutional model be altered to include asymmetry in the antenna beam. However, this will only require a modification to the FSIR portion of the model. The very narrow antenna pattern used by the MARA system is another difference from past work because this system employs beamwidth limited altimetry.

An imposing aspect of this model is that the convolutions can become quite difficult to perform analytically, especially when effects of the asymmetric antenna pattern are incorporated into the FSIR expression. The necessary convolutions can be completed using the fast Fourier transform, which provides substantial time savings over other numerical methods (such as numerical integration of the convolution integral) while preserving accuracy [7]. Fast computing speed, always desirable, is particularly useful in this case so that computations can be made very rapidly on board the MARA experimental aircraft. Thus, if certain parameters (e.g. the aircraft height or radar cross section) change, the models can be recomputed quickly while in flight instead of waiting until the mission is completed.

The goals of this research are then to incorporate the asymmetric antenna pattern into the FSIR portion of the convolutional model and develop separate expressions for nadir, near-nadir and far off-nadir pointing angles. Two numerical convolution methods will be examined; the numerical integration of the convolution integral and FFT convolution will be compared and the resulting algorithms tested for validity. The FFT algorithm will be chosen to compute the necessary convolutions because of its superior speed over the numerical integration algorithm. Some example waveforms that are representative of the expected MARA system average return pulses will then be computed. These will be used to show that an estimate of the pointing angle can be obtained from the width of the return pulses. Although a useful result for studying pointing angle errors and sensitivity, more research into this topic is required in order to obtain the more accurate pointing angle determination methods that are important for such tasks as geoid and ocean currents and eddies measurements.

# 2.0 FLAT SURFACE IMPULSE RESPONSE (FSIR) COMPUTATION

The average return power waveform as a function of time can be defined as the convolution of the radar system point target response with the convolution of the height probability density function of the specular points on the reflecting surface and the average flat surface impulse response (FSIR) [9,20]. The FSIR, denoted $P_{FS}(t)$, is itself a function of the antenna pattern, the radar cross section per unit scattering area, and the range to the scattering point on the surface [9]. This model can be used as long as the pointing angle remains reasonably near normal incidence [9], an assumption that holds for the MARA system [27].

The basic equation that governs the flat surface impulse response is investigated and the effects of the asymmetric antenna pattern are included to provide simplified expressions for $P_{FS}(t)$ valid for the MARA system. As it turns out, there are three specific regions in pointing angle ($\xi$) where different evaluation methods are necessary to compute the FSIR - when $\xi = 0^\circ$, when $\xi$ is small (relative to a beamwidth), and when $\xi$ is large. These expressions are then used in an algorithm (listed in Appendix A) that automatically determines which should be used to compute $P_{FS}(t)$.

## 2.1 UNEQUAL BEAMWIDTHS MODELING

To include the effects of an asymmetric antenna pattern in the evaluation of the FSIR, the basic equation governing its behavior must be examined. Figure 2.1 shows the geometry of the radar system and [9] gives the general form for $P_{FS}(t)$ or the flat surface impulse response

$$P_{FS}(t) = \frac{\lambda^2}{(4\pi)^2 L_p} \int_S \frac{\delta\left(t - \frac{2r}{c}\right) G^2(\theta,\omega)\ \sigma^\circ(\psi,\phi)}{r^4}\, dA \tag{2.1-1}$$

where

| | |
|---|---|
| $\lambda$ | radar wavelength |
| $L_P$ | two-way propagation loss (over spreading losses) |
| $\delta(t - 2h/c)$ | transmitted delta function delayed in time |
| $c$ | speed of light |
| $G(\theta,\omega)$ | power gain pattern of radar antenna |
| $r$ | range from the radar to the elemental scattering area $dA$ on the surface |

and the integration is performed over the illuminated area $S$. The development of $P_{FS}(t)$ in [9] assumes that the antenna pattern described by $G(\theta,\omega)$ is a circularly symmetric Gaussian beam given by

$$G(\theta,\omega) = G_o\ exp(-\tfrac{2}{\gamma}\ sin^2\theta). \tag{2.1-2}$$

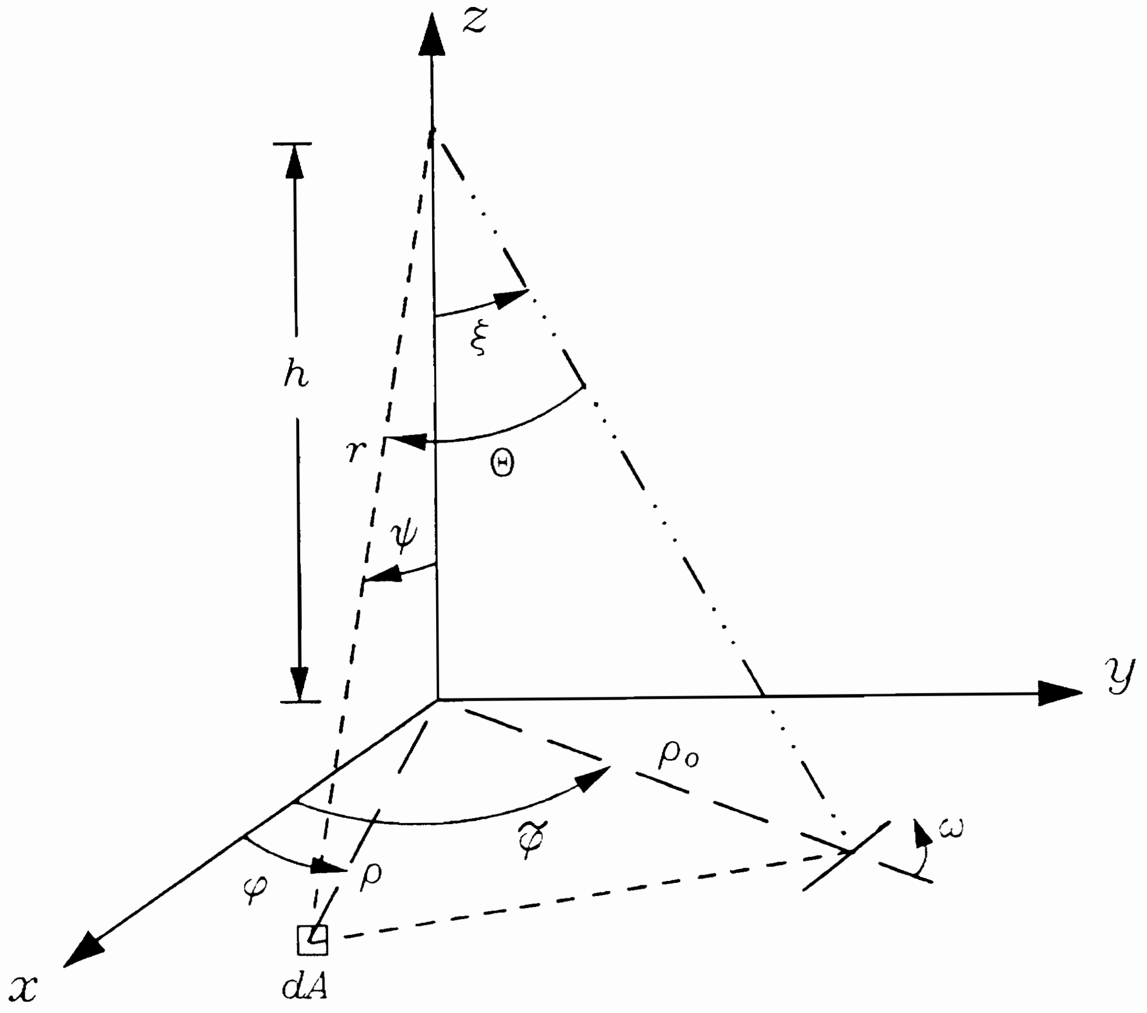


**Figure 2.1.** Radar system geometry, indicating relationship of parameters.

where $\gamma$ is a parameter that is defined by the half-power beamwidth of the antenna power pattern. The MARA system does not, however, have a circularly symmetric antenna pattern.

The MARA system uses a lens and horn configuration to produce a narrow beam antenna pattern [25]. This allows multiple beams to be generated by independent horns, each making use of the same lens. Figure 2.2 shows the geometry of the MARA system where the large arrow indicates the direction of aircraft/spacecraft motion and $\alpha$ determines the orientation of the orthogonal feed horn assembly with respect to the direction of travel. The small numbers on the orthogonal arms indicate the positions of the five horn antennas in the Multibeam Mode. Note that these positions can be changed to any pointing angle from 0 through 12 degrees off-nadir on each arm. The dielectric lens measures 0.86 m in diameter and 15.2 cm (maximum) thick, with a focal length that varies with input angle from about 100 to 115 cm. When $\alpha = 0^\circ$, the antenna footprints are oriented such that the nadir, fore and aft beams are all in the same plane as the direction of travel and the remaining two beams are pointed toward either side, in the cross-track direction. When $\alpha = 26.6^\circ$, the feed assembly becomes oriented such that the antenna footprints are equally spaced on the surface, which creates the previously mentioned 'pushbroom' pattern as shown in Figure 2.2 [23]. Figure 2.3 demonstrates the coordinate system used for the lens, defining the scan and cross-scan directions. Note that scan and cross-scan are referenced to the lens coordinate system and are not dependent on the direction of travel.

For the MARA system, measurements have shown [22,25] that the half-power beamwidth varies in the cross-scan dimension as the pointing angle

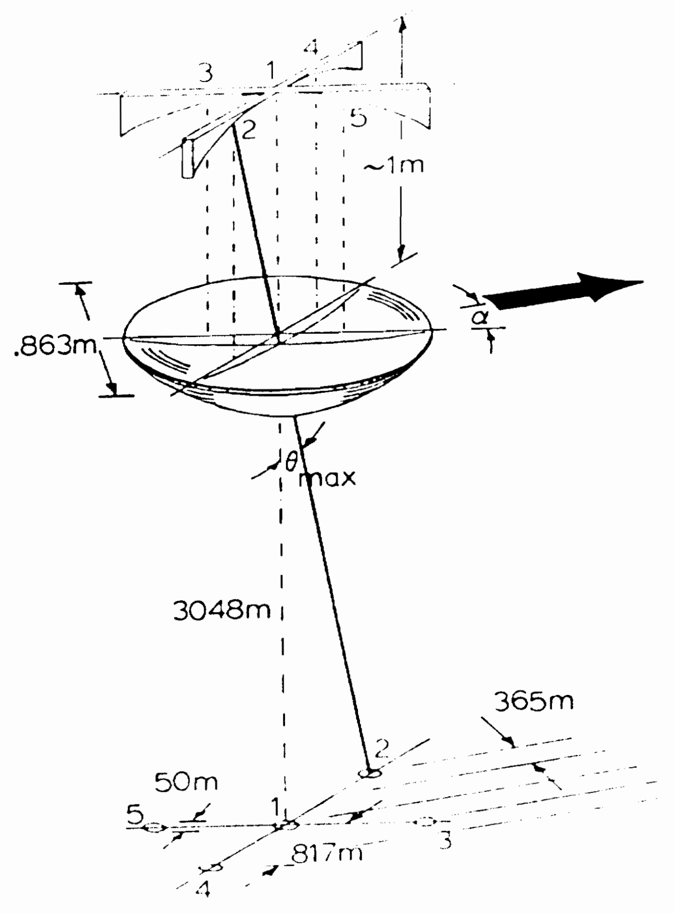


**Figure 2.2.** MARA system geometry [27].

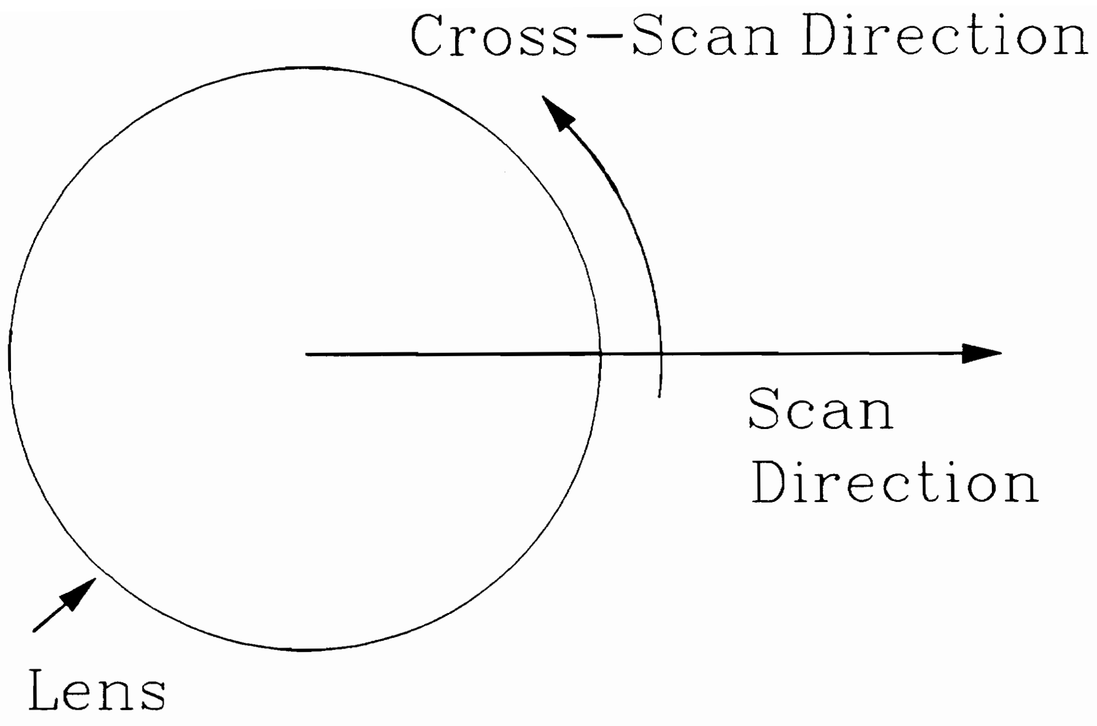


Figure 2.3. Geometry of the MARA lens (top view) [22].

changes, while the half-power beamwidth remains relatively constant in the scan direction. Figure 2.4. shows how the beamwidth changes in the cross-scan direction as the pointing angle increases. In this figure, the crosses represent the measured cross-scan beamwidth values, while the diamond shapes indicate the measured scan beamwidth [25]. A cubic spline-fitting program was used to compute cross-scan half-power beamwidth, or $\theta_{xs}$, values for 0.1° steps in the pointing angle, which are listed in Table 2.1. This table was stored in a data file so that the final waveform generating program could access any of the beamwidth values. The dashed line in Figure 2.4 represents the average scan half-power beamwidth value $\theta_s = 0.6313°$, an average of the measured scan half-power beamwidths. The scan beamwidth is taken to be constant since it changes very little, compared to the cross-scan beamwidth, as the pointing angle changes.

The asymmetrical antenna pattern is thus modeled as an elliptic Gaussian beam using the form similar to (2.1-2), i.e.,

$$G(\theta,\omega) = G_o \, exp\left[-\tfrac{2}{\gamma}\,(1 \, + \, \beta \, sin^2\omega) \, sin^2\theta\right] \tag{2.1-3}$$

where $\omega$ is the angle about the antenna boresight axis, as defined in Figure 2.1, and $\beta$ is a parameter defined by $\theta_{xs}$, the cross-scan (changing) beamwidth. Note that when $\beta = 0$, (2.1-3) reduces to (2.1-2), the circularly symmetric case. It should also be noted here that the value of $G_o$ in (2.1-3) also changes as the pointing angle changes because the beamwidth is not constant. Measurements revealed that at nadir ($\xi = 0°$), $G_o = 46.0$ dB, while at $\xi = 12°$, $G_o = 44.5$ dB [24]. To model the change in gain as the pointing angle changes, an empirical equation was developed from a similar equation found in [30] using the two measured

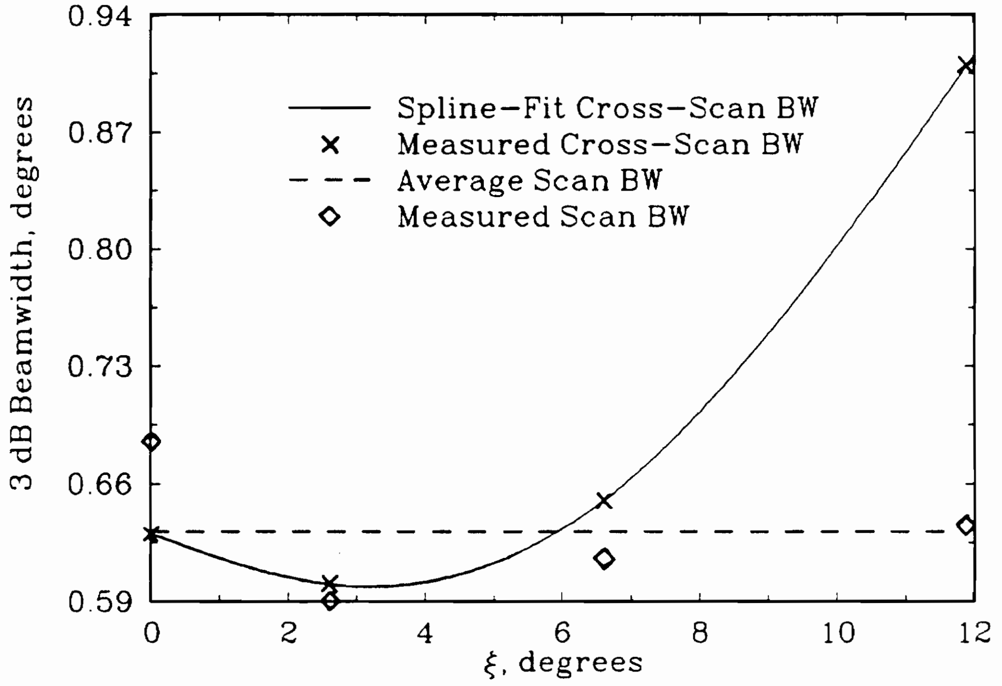


**Figure 2.4.** MARA system half-power beamwidth as the pointing angle changes. Average scan beamwidth (dashed line) is 0.6313°.

Table 2.1. Cross-scan half-power beamwidth values obtained from the cubic spline interpolation routine. All values in degrees.

| $\xi$ | $\theta_{xs}$ | $\xi$ | $\theta_{xs}$ | $\xi$ | $\theta_{xs}$ | $\xi$ | $\theta_{xs}$ |
|---|---|---|---|---|---|---|---|
| 0.00 | 0.6300 | 3.00 | 0.5987 | 6.00 | 0.6331 | 9.00 | 0.7496 |
| 0.10 | 0.6285 | 3.10 | 0.5986 | 6.10 | 0.6356 | 9.10 | 0.7546 |
| 0.20 | 0.6270 | 3.20 | 0.5986 | 6.20 | 0.6383 | 9.20 | 0.7597 |
| 0.30 | 0.6255 | 3.30 | 0.5987 | 6.30 | 0.6411 | 9.30 | 0.7648 |
| 0.40 | 0.6241 | 3.40 | 0.5988 | 6.40 | 0.6440 | 9.40 | 0.7700 |
| 0.50 | 0.6226 | 3.50 | 0.5990 | 6.50 | 0.6469 | 9.50 | 0.7752 |
| 0.60 | 0.6211 | 3.60 | 0.5993 | 6.60 | 0.6500 | 9.60 | 0.7805 |
| 0.70 | 0.6197 | 3.70 | 0.5997 | 6.70 | 0.6532 | 9.70 | 0.7858 |
| 0.80 | 0.6183 | 3.80 | 0.6002 | 6.80 | 0.6564 | 9.80 | 0.7912 |
| 0.90 | 0.6169 | 3.90 | 0.6007 | 6.90 | 0.6598 | 9.90 | 0.7966 |
| 1.00 | 0.6156 | 4.00 | 0.6014 | 7.00 | 0.6633 | 10.00 | 0.8020 |
| 1.10 | 0.6142 | 4.10 | 0.6021 | 7.10 | 0.6668 | 10.10 | 0.8075 |
| 1.20 | 0.6129 | 4.20 | 0.6029 | 7.20 | 0.6705 | 10.20 | 0.8130 |
| 1.30 | 0.6117 | 4.30 | 0.6038 | 7.30 | 0.6742 | 10.30 | 0.8185 |
| 1.40 | 0.6105 | 4.40 | 0.6048 | 7.40 | 0.6780 | 10.40 | 0.8241 |
| 1.50 | 0.6093 | 4.50 | 0.6059 | 7.50 | 0.6819 | 10.50 | 0.8297 |
| 1.60 | 0.6082 | 4.60 | 0.6070 | 7.60 | 0.6859 | 10.60 | 0.8353 |
| 1.70 | 0.6071 | 4.70 | 0.6083 | 7.70 | 0.6900 | 10.70 | 0.8410 |
| 1.80 | 0.6060 | 4.80 | 0.6096 | 7.80 | 0.6942 | 10.80 | 0.8466 |
| 1.90 | 0.6051 | 4.90 | 0.6111 | 7.90 | 0.6984 | 10.90 | 0.8523 |
| 2.00 | 0.6041 | 5.00 | 0.6126 | 8.00 | 0.7027 | 11.00 | 0.8581 |
| 2.10 | 0.6033 | 5.10 | 0.6142 | 8.10 | 0.7071 | 11.10 | 0.8638 |
| 2.20 | 0.6025 | 5.20 | 0.6159 | 8.20 | 0.7116 | 11.20 | 0.8695 |
| 2.30 | 0.6018 | 5.30 | 0.6178 | 8.30 | 0.7161 | 11.30 | 0.8753 |
| 2.40 | 0.6011 | 5.40 | 0.6197 | 8.40 | 0.7207 | 11.40 | 0.8811 |
| 2.50 | 0.6005 | 5.50 | 0.6217 | 8.50 | 0.7254 | 11.50 | 0.8868 |
| 2.60 | 0.6000 | 5.60 | 0.6237 | 8.60 | 0.7301 | 11.60 | 0.8926 |
| 2.70 | 0.5996 | 5.70 | 0.6259 | 8.70 | 0.7349 | 11.70 | 0.8984 |
| 2.80 | 0.5992 | 5.80 | 0.6282 | 8.80 | 0.7397 | 11.80 | 0.9042 |
| 2.90 | 0.5989 | 5.90 | 0.6306 | 8.90 | 0.7446 | 11.90 | 0.9100 |
|  |  |  |  |  |  | 12.00 | 0.9158 |

values and their corresponding orthogonal beamwidth values;

$$G_o(\xi) = \frac{15833.5}{\theta_{xs}(\xi)\,\theta_s}\left[1 \;+\; \frac{\xi}{12}\,(0.03)\right]. \tag{2.1-4}$$

It is important to note that (2.1-4) is only valid for the MARA case at hand; other systems would require a re-computation to fit the desired gain values.

Next the parameters $\gamma$ and $\beta$ in (2.1-3) must be examined. $\gamma$ is determined by the scan (constant) beamwidth, while $\beta$ will describe the way the beamwidth changes in the cross-scan direction; thus $\beta$ is a function of both $\theta_s$ and $\theta_{xs}$. An expression for $\gamma$ can be found by using (2.1-3), setting $\omega = 0$ (scan direction), $G(\theta,\omega)/G_o = 0.5$ (half-power value), and $\theta$ to one half of the unchanging beamwidth $\theta_s$. Solving for $\gamma$ yields

$$\gamma = -\frac{2\,sin(\theta_s/2)}{\ln(1/2)}\,. \tag{2.1-5}$$

To find the value of $\beta$ for a particular $\xi$, we set $\omega = \pi/2$ (cross-scan direction), $G(\theta,\omega)/G_o = 0.5$ (half-power value), and $\theta = \theta_{xs}/2$, found from Table 2.1 as a function of $\xi$. Solving for $\beta$ yields

$$\beta = -\left[1 + \frac{\gamma\,ln(1/2)}{2\,sin(\theta_{xs}/2)}\right]. \tag{2.1-6}$$

The values of $\beta$ defined by (2.1-6) must be recomputed for each $\xi$ and are completely specified by $\theta_s$ and $\theta_{xs}$.

Now (2.1-3) can be used in (2.1-1) to yield;

$$P_{FS}(t) = \frac{G_o^2\,\lambda^2}{(4\pi)^3 L_P} \int\limits_S \frac{\delta\!\left(t - \frac{2r}{c}\right)}{r^4}\,\sigma^o(\psi,\phi)$$

$$\cdot\, exp\!\left\{-\tfrac{4}{\gamma}\,(1 + \beta\, sin^2\omega)\, sin^2\theta\right\} dA. \qquad (2.1\text{-}7)$$

Since the coordinate system of the altimeter is chosen to be cylindrical and $z = 0$ is taken to be the mean surface, the incremental area $dA$ can be written as $\rho\, d\rho\, d\phi$. The range variable, $r$, can be written as $r = \sqrt{h^2 + \rho^2}$ and with $\epsilon = \rho/h$, (2.1-7) can be expressed as;

$$P_{FS}(t) = \frac{G_o^2\lambda^2}{(4\pi)^3 L_P h^4} \int_0^{2\pi}\!\!\int_0^{\infty} \frac{\delta\!\left(t - \frac{2h}{c}\sqrt{1+\epsilon^2}\right)}{(1+\epsilon^2)}\,\sigma^o(\psi,\phi)$$

$$\cdot\, exp\!\left\{-\tfrac{4}{\gamma}\,(1 + \beta\, sin^2\omega)\, sin^2\theta\right\} \rho\, d\rho\, d\phi. \qquad (2.1\text{-}8)$$

Since the pointing angles and beamwidths under consideration are relatively small, it is assumed that $\sigma^o(\psi,\phi)$ is $\phi$-independent. That is, since the small pulse widths and narrow beamwidths encountered in this type of altimeter make the effective illuminated area encompass such a small angular spread, $\sigma^o$ may be assumed constant in $\phi$ [9]. Under a carefully selected sequence of changes of variables, the $\rho$-integration of (2.1-8) can be completed, yielding;

$$P_{FS}(t) = \frac{G_o^2\lambda^2 (c/2)\,\sigma^o(\psi)}{(4\pi)^3 L_P h^3 (ct/2h)^3} \int_0^{2\pi} exp\!\left\{-\tfrac{4}{\gamma}\,(1 + \beta\, sin^2\omega)\, sin^2\theta\right\} d\phi \qquad (2.1\text{-}9)$$

for $t \geq 2h/c$. The $sin^2\omega$ and $sin^2\theta$ terms in the exponential of (2.1-9) can be rewritten in terms of $\xi$, $\phi$ and $\rho$ as follows [5];

$$sin^2\omega = \frac{\rho^2 sin^2\phi}{\rho^2 - 2\rho\rho_o cos\phi + \rho_o^2}$$

$$sin^2\theta = 1 - \frac{(cos\xi + \epsilon\, sin\xi\, cos\phi)^2}{1+\epsilon^2} \qquad (2.1\text{-}10)$$

where $\epsilon = \frac{\rho}{h} = \sqrt{\left(\frac{ct}{2h}\right)^2 - 1}$ and $\rho_o = h\,tan\xi$. When eqs. (2.1-10) are substituted back into (2.1-9), it becomes clear that the $\phi$-integration cannot be accomplished by exact analytical methods; thus, numerical techniques must be applied in order to evaluate (2.1-9).

## 2.2 NADIR ORIENTATION

When the radar antenna is pointed directly at nadir ($\xi = 0^\circ$) the integrand of (2.1-9) simplifies considerably. When $\xi = 0^\circ$, $\rho_o = 0$ and eqs. (2.1-10) become;

$$sin^2\omega = sin^2\phi$$

$$sin^2\theta = 1 - \frac{1}{1+\epsilon^2} = \frac{\epsilon^2}{1+\epsilon^2}\,. \qquad (2.2\text{-}1)$$

Thus, (2.1-9) becomes;

$$P_{FS}(t,\, \xi{=}0^\circ) = \frac{G_o^2\lambda^2(c/2)\sigma^o(\psi)}{(4\pi)^3 L_P h^3 (ct/2h)^3} \int_0^{2\pi} exp\left\{-\frac{4}{\gamma}\,(1+\beta\, sin^2\phi)\frac{\epsilon^2}{1+\epsilon^2}\right\} d\phi \qquad (2.2\text{-}2)$$

for $t \geq 2h/c$. Let $W$ represent the integral in (2.2-2):

$$W = \int_0^{2\pi} exp\left\{-\frac{4}{\gamma}\,(1+\beta\, sin^2\phi)\frac{\epsilon^2}{1+\epsilon^2}\right\} d\phi. \qquad (2.2\text{-}3)$$

After rearranging terms and using the trigonometric identity $sin^2\phi = \frac{1}{2}(1 - cos2\phi)$ [5], (2.2-3) becomes;

$$W = exp\left\{-\left[\frac{4\epsilon^2}{\gamma(1+\epsilon^2)}\right]\left[1+\frac{\beta}{2}\right]\right\} \int_0^{2\pi} exp\left\{\frac{2\beta\epsilon^2}{\gamma(1+\epsilon^2)}\, cos2\phi\right\} d\phi. \qquad (2.2\text{-}4)$$

Now the exponential inside the integral of (2.2-4) can be expanded into a series of modified Bessel functions denoted by $I_n$, using the identity [1]:

$$e^{z cos\theta} = I_0(z) + 2\sum_{k=1}^{\infty} I_k(z) cos\, k\theta. \qquad (2.2\text{-}5)$$

Substituting (2.2-5) into (2.2-4) yields;

$$W = exp\left\{-\left[\frac{4\epsilon^2}{\gamma(1+\epsilon^2)}\right]\left[1+\frac{\beta}{2}\right]\right\}\left\{\int_0^{2\pi} I_0\left[\frac{2\beta\epsilon^2}{\gamma(1+\epsilon^2)}\right] d\phi \right.$$
$$\left. + 2\int_0^{2\pi} \sum_{k=1}^{\infty} I_k\left[\frac{2\beta\epsilon^2}{\gamma(1+\epsilon^2)}\right] cos\,2k\phi\; d\phi\right\}. \qquad (2.2\text{-}6)$$

Note, however, that the second integrand in (2.2-6) is $cos\,2k\phi$ over the interval $\phi \in [0,2\pi]$. Since $k$ is always an integer, the second integral in (2.2-6) reduces to zero; the first integrand is independent of $\phi$, so its evaluation is trivial. Thus, the infinite series summation in unnecessary and only $I_0(\cdot)$ is required to evaluate the equation. There are many polynomial approximations available [1] for evaluating $I_o$ that are quite accurate and suitable for numerical evaluations (see Section

2.2.1).

Now (2.2-6) reduces to;

$$W = 2\pi\, exp\left\{-\left[\frac{4\epsilon^2}{\gamma(1+\epsilon^2)}\right]\left[1+\frac{\beta}{2}\right]\right\} I_0\left[\frac{2\beta\epsilon^2}{\gamma(1+\epsilon^2)}\right] \tag{2.2-7}$$

and when substituted in to (2.2-2), the expression for $P_{FS}(t)$ for nadir pointing ($\xi = 0^\circ$) becomes;

$$P_{FS}(t,\, \xi{=}0^\circ) = \frac{G_o^2\lambda^2 c\sigma^o(\psi)}{4(4\pi)^2 L_p h^3 (ct/2h)^3}\, exp\left\{-\left[\frac{4\epsilon^2}{\gamma(1+\epsilon^2)}\right]\left[1+\frac{\beta}{2}\right]\right\} I_0\left[\frac{2\beta\epsilon^2}{\gamma(1+\epsilon^2)}\right] \tag{2.2-8}$$

for $t \geq 2h/c$. This can be somewhat simplified if the time variable $t$ is converted to the two-way incremental ranging time $\tau = t - 2h/c$. For the MARA height (3048 m) and since the ranging time will not be more than about 500 ns [24], $c\tau/\mathrm{h} \ll 1$ and (2.2-8) can be written as;

$$P_{FS}(\tau,\, \xi{=}0^\circ) = \frac{G_o^2\lambda^2 c\sigma^o(\psi)}{4(4\pi)^2 L_p h^3}\, exp\left\{-\frac{4\epsilon^2}{\gamma}\left[1+\frac{\beta}{2}\right]\right\} I_0\left[\frac{2\beta\epsilon^2}{\gamma}\right], \quad \tau \geq 0 \tag{2.2-9}$$

where $\epsilon = \sqrt{c\tau/h}$. Note that $\epsilon$ is now different from its form in (2.1-10).

### *2.2.1 BESSEL FUNCTION EVALUATION*

When the pointing angle is zero, the expression for $P_{FS}(\tau)$ contains an evaluation of the zeroth order modified Bessel function $I_0(\cdot)$. [1] provides several useful formulas to approximate various Bessel functions, each exact to several decimal places. Those most relevant to this application are reproduced here for convenience;

$$I_0(x) = 1 + a_2t^2 + a_4t^4 + a_6t^6 + a_8t^8 + a_{10}t^{10} + a_{12}t^{12} + \epsilon$$
$$|\epsilon| \leq 1.6 \times 10^{-7}, \quad |x| \leq 3.75 \qquad (2.2\text{-}10)$$

$$\sqrt{x}\, e^{-x} I_0(x) = b_0 + b_1t^{-1} + b_2t^{-2} + b_3t^{-3} + b_4t^{-4}$$
$$+ b_5t^{-5} + b_6t^{-6} + b_7t^{-7} + b_8t^{-8} + \epsilon \qquad (2.2\text{-}11)$$
$$|\epsilon| \leq 1.9 \times 10^{-7}, \quad 3.75 \leq x \leq \infty$$

which are eqs. 9.8.1 and 9.8.2, respectively, of [1]. In (2.2-10,11), $t = x/3.75$ and the $a_n$'s and $b_n$'s are given in [1]. Note that both of these equations are polynomial *approximations*, each with its individual error, $\epsilon$. Multiplying both sides of (2.2-10) by $e^{-x}$ and both sides of (2.2-11) by $x^{-1/2}$ yields two equations for $e^{-x}I_0(x)$ over the entire range $0 \leq x < \infty$. It is important to note that these equations are valid only for positive $x$. The argument of $I_o$ in (2.2-9) is $\frac{2\beta\epsilon^2}{\gamma}$ which can be negative since $\beta$ can be negative. To remedy this problem consider the analytical continuation equation for Bessel functions of the first kind [1]

$$I_\nu(xe^{jm\pi}) = e^{jm\nu\pi} I_\nu(x), \quad (m \text{ an integer}). \qquad (2.2\text{-}12)$$

If $m = 1$ and $\nu = 0$ (the zeroth order Bessel function is the only one of concern), (2.2-12) reduces to:

$$I_0(-x) = I_0(x). \tag{2.2-13}$$

(2.2-13) simply shows that the zeroth order Bessel function is an even function. Incidentally, examining (2.2-12) further reveals that the even-ordered Bessel functions are even functions while the odd-ordered Bessel functions are odd functions.

Because of (2.2-13), any of the zeroth order Bessel function evaluations can be made using the magnitude of the argument without any error. Thus,

$$I_0\left(\frac{2\beta\epsilon^2}{\gamma}\right) = I_0\left(\frac{2\,|\,\beta\,|\,\epsilon^2}{\gamma}\right) \tag{2.2-14}$$

can be used for the required evaluations in (2.2-9). Now (2.2-9) can be manipulated into a form having the factor $e^{-x}I_0(x)$ if the term $\frac{2\,|\,\beta\,|\,\epsilon^2}{\gamma}$ is added and subtracted inside the exponential. This yields;

$$P_{FS}(\tau,\ \xi{=}0^{\circ}) = \frac{G_o^2\lambda^2 c\sigma^o(\psi)}{4(4\pi)^2 L_p h^3}\, exp\left\{-\frac{4\epsilon^2}{\gamma}\left[1+\frac{\beta}{2}-\frac{|\,\beta\,|}{2}\right]\right\} F\left[\frac{2\,|\,\beta\,|\,\epsilon^2}{\gamma}\right] \tag{2.2-15}$$

for $\tau \geq 0$ and $F(x) = e^{-x}I_0(x),\quad x > 0$. Now $F$ can be replaced by the modified versions of either (2.2-10) or (2.2-11), depending on the magnitude of $\frac{2\,|\,\beta\,|\,\epsilon^2}{\gamma}$, allowing reliable evaluation of the FSIR at nadir where the magnitude of the error will be no greater than $1.9 \times 10^{-7}$.

## 2.3 NEAR-NADIR ORIENTATION

For the more general case of when $\xi \neq 0^\circ$, but $\xi$ is still only a few beamwidths, the only option left in evaluating (2.1-9) is to perform a numerical integration of the $\phi$ variable. However, it is still more advantageous to evaluate (2.1-9) in terms of $\tau$ instead of $t$;

$$P_{FS}(\tau) = \frac{G_o^2 \lambda^2 (c/2) \sigma^o(\psi)}{(4\pi)^3 L_P h^3} \int_0^{2\pi} exp\left\{ - \frac{4}{\gamma} \left[ 1 + \beta \frac{\rho^2 sin^2\phi}{\rho^2 - 2\rho\rho_o cos\phi + \rho_o^2} \right] \cdot \left[ 1 - \frac{(cos\xi + \epsilon\, sin\xi\, cos\phi)^2}{1+\epsilon^2} \right] \right\} d\phi \qquad (2.3\text{-}1)$$

for $\tau \geq 0$ and $\epsilon = \sqrt{\frac{c\tau}{h}}$, $\rho_o = h\, tan\xi$, and eqs. (2.1-10) were substituted into (2.1-9).

The integrand in the above expression is a well behaved, smoothly varying function that is peaked at the point where the antenna boresight axis intersects the surface. One of the simplest numerical integration schemes that can be applied to this problem is Simpson's rule. Numerical comparisons were performed and it was found that dividing the $2\pi$ range of $\phi$ into 200 intervals was sufficient to accurately compute the integral. The integrand is evaluated at each increment with the result summed according to the extended Simpson's rule [1]. Even though many integrations must be done over the range of $\tau$, this simple method results in the best accuracy for given computation time and algorithm complexity. As will be seen in Section 2.4, this method of evaluating the FSIR is only necessary out to a few beamwidths, or equivalently, only a few nanoseconds in $\tau$.

## 2.4 FAR-OFF-NADIR ORIENTATION

An important feature to note about the $\phi$-integration in (2.3-1) is that since the antenna beamwidth is very small, there will only be a significant contribution to the return signal when $\phi$ is in the neighborhood of $\phi_o$ (see Figure 2.1) and $\epsilon = \sqrt{\frac{c\tau}{h}}$ is close to $\rho_o$. Due to this highly peaked nature of the integrand in this region, the $\phi$-integral can be evaluated asymptotically using Laplace's method [4].

To apply Laplace's method to this problem, first note that (2.3-1) can be rewritten in the form;

$$P_{FS}(\tau) = \Gamma \int_{-\pi}^{\pi} exp(W)\, d\phi \;, \quad \tau \geq 0 \tag{2.4-1}$$

where

$$\Gamma = \frac{G_o^2 \lambda^2 (c/2) \sigma^o(\psi)}{(4\pi)^3 L_P h^3}$$

$$W = -\tfrac{4}{\gamma}\,(1 + \beta\, sin^2\omega)\, sin^2\theta.$$

Laplace's method states that if $W$ is a peaked function, then (2.4-1) can be approximated by [4];

$$P_{FS}(\tau) \simeq \Gamma\; exp(W(\phi_o)) \sqrt{\frac{-2\pi}{W''(\phi_o)}}\;, \quad \tau \geq 0 \tag{2.4-2}$$

where $\phi_o$ is determined by;

$$W'(\phi_o) = \frac{dW}{d\phi}\bigg|_{\phi=\phi_o} = 0\ ,\quad W''(\phi_o) = \frac{d^2W}{d\phi^2}\bigg|_{\phi=\phi_o}$$

Making use of the chain rule and several trigonometric identities [5], (2.4-2) becomes;

$$P_{FS}(\tau) \simeq \Gamma\ exp\left\{\frac{4}{\gamma}\left[-1 + \frac{(cos\xi + \epsilon\, sin\xi)^2}{1+\epsilon^2}\right]\right\}\sqrt{\frac{\pi}{\frac{4\epsilon}{\gamma(1+\epsilon^2)}\cdot T}}$$

where (2.4-3)

$$T = cos\xi\, sin\xi + \epsilon\,(sin^2\xi + \beta\, cos^2\xi)$$

for $\tau \geq 0$ and $\Gamma$ was defined in (2.4-1). The integral in (2.3-1) has thus been replaced by the analytical expression in (2.4-3) which is obviously much faster to compute. However, this expression is an *approximation* to the FSIR, so an investigation into the error introduced by using this method is necessary.

## 2.4.1 ERROR IN THE ASYMPTOTIC FSIR

Unfortunately, the asymptotic approximation to $P_{FS}(\tau)$ given by (2.4-3) cannot be used over the entire desired range of pointing angles. Specifically, the approximation breaks down as the pointing angle approaches nadir while the best agreement is seen when the pointing angle is on the order of many beamwidths from nadir. This occurs because as $\xi \to 0^\circ$, the argument of the exponential in (2.4-3) approaches $\frac{4\epsilon^2}{\gamma}$, and $\epsilon = \sqrt{\frac{c\tau}{h}}$ becomes small since $\tau$ is small for $\xi$ near zero. Thus the exponential in (2.4-3) becomes less peaked as $\xi$ approaches zero, which violates the assumption necessary to use Laplace's method [4]. When $\xi$ is large (at least several beamwidths) the exponential is very peaked and Laplace's method can still be used. What must be resolved is when (in $\tau$) and where (in $\xi$) the asymptotic form may be used to obtain $P_{FS}(\tau)$, and still guarantee that the error introduced by this method is below some acceptable level.

This problem was addressed in [10] for the case of circularly symmetric antenna beamwidths, but modifications to that analysis must be made in order to include the case of a changing, asymmetric beam. In [10], extensive numerical studies were used to derive a condition that, if met, would guarantee that the error in the asymptotic form would be less than 2% when compared to exact calculations. This condition on $\tau$ and $\xi$, reproduced here, is;

$$\frac{8\,tan\xi}{\gamma(1 + tan^2\xi)}\sqrt{\frac{c\tau}{h}} \geq 6.79 \tag{2.4-4}$$

where $\gamma$ is determined by the antenna beamwidth by (2.1-5). Thus, for a given look angle $\xi$ and the value of $\gamma$ found from (2.1-5), a condition on $\tau$ can be found from (2.4-4);

$$\tau \geq \frac{h}{c}\left[\frac{0.849\,\gamma(1 + tan^2\xi)}{tan\xi}\right]^2 . \tag{2.4-5}$$

This states that whenever (2.4-5) is met, the asymptotic form of the FSIR can be used with the error guaranteed to be less than 2% of the exact value. Figure 2.5

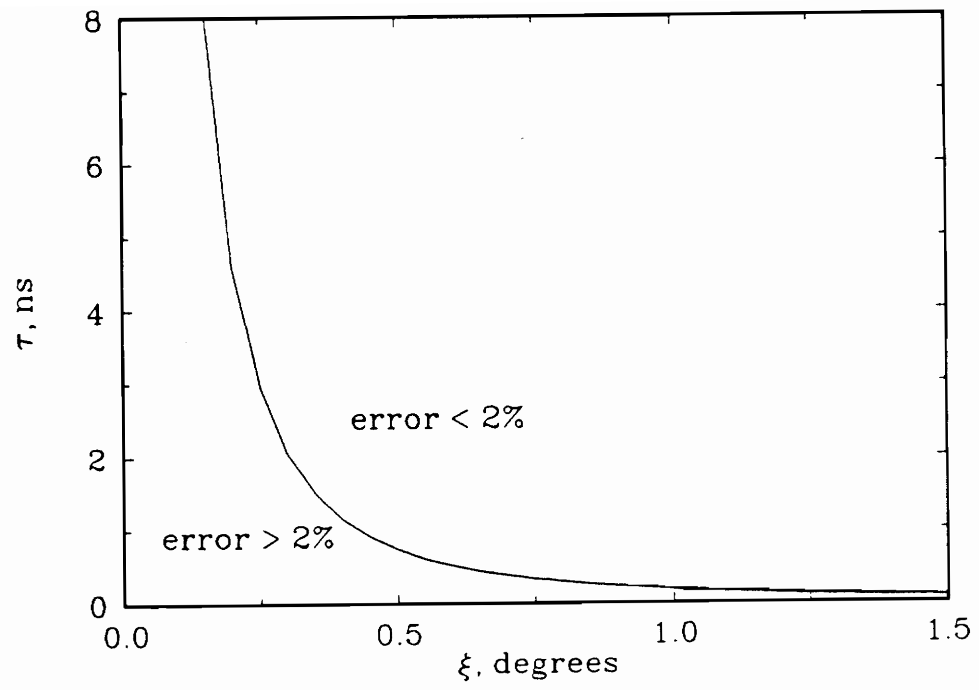


**Figure 2.5.** 2% error boundary, equation (2.4-5), for $h = 3048$ m (10,000 ft) and circularly symmetric beamwidth of 0.6313°.

shows a plot of $\tau$ versus $\xi$ for a beamwidth of 0.6313° and a radar altitude of 3048 meters (10,000 feet). It should be emphasized once again that (2.4-5) is valid only for the case of circularly symmetric antenna beams. This expression must be modified to provide an equally useful condition for unequal beamwidths.

Examining (2.4-5) shows that if the parameter $\gamma$ (or equivalently, the beamwidth) varies, then the minimum allowable time for the use of the asymptotic evaluation will also change. Instead of computing $\gamma$ as was done for the constant beamwidth case in (2.1-5), consider using the the changing cross-scan beamwidth, $\theta_{xs}$, to compute an alternate $\gamma$, called $\gamma_a$, the same way as was done for $\gamma$ in (2.1-5), i.e.;

$$\gamma_a = -\frac{2\,sin^2(\theta_{xs}/2)}{ln(1/2)} \tag{2.4-6}$$

where $\theta_{xs}$, depends on $\xi$ as shown in Figure 2.4 or Table 2.1. Using $\gamma_a$ in place of $\gamma$ in (2.4-5) yields an alternate condition on $\tau$ and $\xi$;

$$\tau_a \geq \frac{h}{c}\left[\frac{0.849\,\gamma_a(1+tan^2\xi)}{tan\xi}\right]^2. \tag{2.4-7}$$

Now (2.4-7) and (2.4-5) provide two different expressions which, if satisfied, means that the asymptotic evaluation of $P_{FS}(\tau)$ can be used in place of the more time consuming numerical integration; but which condition is correct? Intuitively, if one condition yields a greater delay time than the other for a given look angle, then that condition should be used, which would guarantee that the asymptotic form is not used too soon. While this is the more conservative approach, the difference in the two values is rarely large, so forcing the more time-consuming numerical integration method to be used longer increases the overall computing time only slightly.

One last point that should be mentioned is that if an evaluation of $P_{FS}(\tau)$ at $\tau = 0$ is desired for any $\xi \neq 0^\circ$, then the numerical integration is always required. This can be seen in Figure 2.5, by noting that the error curve tends toward infinity as $\tau$ goes to zero. As will be seen in Section 5.3, when the pointing angle is large, $P_{FS}(\tau)$ is essentially zero for $\tau$ small, so it is advantageous to begin evaluating $P_{FS}(\tau)$ at some value of $\tau > 0$ to avoid unnecessary computing time spent in the numerical integration.

There are now three different expressions, each derived from the same basic equation, which can be used to compute the flat surface impulse response for a given range of pointing angles. Equation (2.2-15) can be used for the case when the pointing angle is zero ($\xi = 0^\circ$). Equation (2.3-1) must be used when the pointing angle is not zero but is still the order of a beamwidth, and equation (2.4-3) can be used when the pointing angle and time value satisfy either (2.4-5) or (2.4-7), whichever is more conservative. In any case, these three expressions can be placed in an algorithm that automatically determines which method should be used, given the pointing angle and beamwidth information, as is done in the subroutine PFSIR, listed in Appendix A, Section A.2.2.

# 3.0 COMBINED SEA STATE AND POINT TARGET RESPONSE

As stated earlier, the average return power waveform can be expressed as the convolution of three individual functions: the flat surface impulse response, the radar system point target response, and the sea state function [9]. Chapter 2 dealt with the development of methods to compute the flat surface impulse response; this chapter will detail the development of the two remaining functions, the sea state and radar system point target responses.

The sea state function represents the statistical distribution of the wave heights on the ocean. The radar system point target response is a measure of how the system responds to a point or very small target. Neither of these functions are dependent on the pointing angle, as is the case for the FSIR. For this reason, they can be computed, convolved and stored just once and then convolved with the FSIR for a given pointing angle. Not having to recompute this combined distribution for each pointing angle results in considerable time savings.

## 3.1 FORMING THE COMBINED RESPONSE

From Fourier transform theory [6] it is known that convolution is associative, i.e.

$$\left[P_{FS}(\tau) * P_{PT}(\tau)\right] * q(\tau) = P_{FS}(\tau) * \left[P_{PT}(\tau) * q(\tau)\right]. \tag{3.1-1}$$

where $*$ is the convolution operator and $q(\tau)$ and $P_{PT}(\tau)$ represent the surface height probability density function, or sea state response, and the radar system point target response, respectively. This means that the three functions can be convolved in any order without affecting the outcome. Since the radar system point target response and the sea state response remain unchanged as the pointing angle changes, these two functions will be convolved together first, yielding;

$$r(\tau) = P_{PT}(\tau) * q(\tau) \tag{3.1-2}$$

Once this result is known, it can be stored and convolved with the FSIR for each $\xi$ to obtain;

$$P_R(\tau) = P_{FS}(\tau) * r(\tau) \tag{3.1-3}$$

What must first be done is to derive the equations that govern the behavior of $P_{PT}(\tau)$ and $q(\tau)$ and then discuss their convolution.

## 3.2 RADAR SYSTEM POINT TARGET RESPONSE

The radar system point target response (PTR), represented by $P_{PT}(\tau)$, can be thought of as the radar system's response to the illumination of a target of infinitesimal extent (compared to the transmitted pulsewidth and antenna beamwidth) by an energy envelope of specific width and amplitude. Essentially, the PTR is represented by the transmitted pulse shape (pulse width and peak amplitude). The MARA system [26] transmits a 1 kW, 6.55 ns pulse at a variable rate. Figure 3.1 shows the Extended Interaction Klystron (EIK) output pulse as reported in [26], which is essentially $P_{PT}(\tau)$. These analog values were sampled at 0.4 ns spacing and stored in a data file. The same cubic spline algorithm that was used to obtain the beamwidth data in Section 2.1 can be used to obtain the point target response data for any sample spacing desired. The plot in Figure 3.1 used a 0.1 ns sampling interval.

### *3.2.1 GAUSSIAN APPROXIMATION*

As an alternative, $P_{PT}(\tau)$ can also be approximated by a Gaussian function [9] whose half-power width can be specified to match the measured data;

$$P_{PT}(\tau) = \widehat{P}_T \, exp\left\{-\frac{1}{2}\left(\frac{\tau - \mu_p}{\sigma_p}\right)^2\right\} \tag{3.2-1}$$

where $\mu_p$ and $\sigma_p$ are the mean and standard deviation, and $\widehat{P}_T$ is the peak

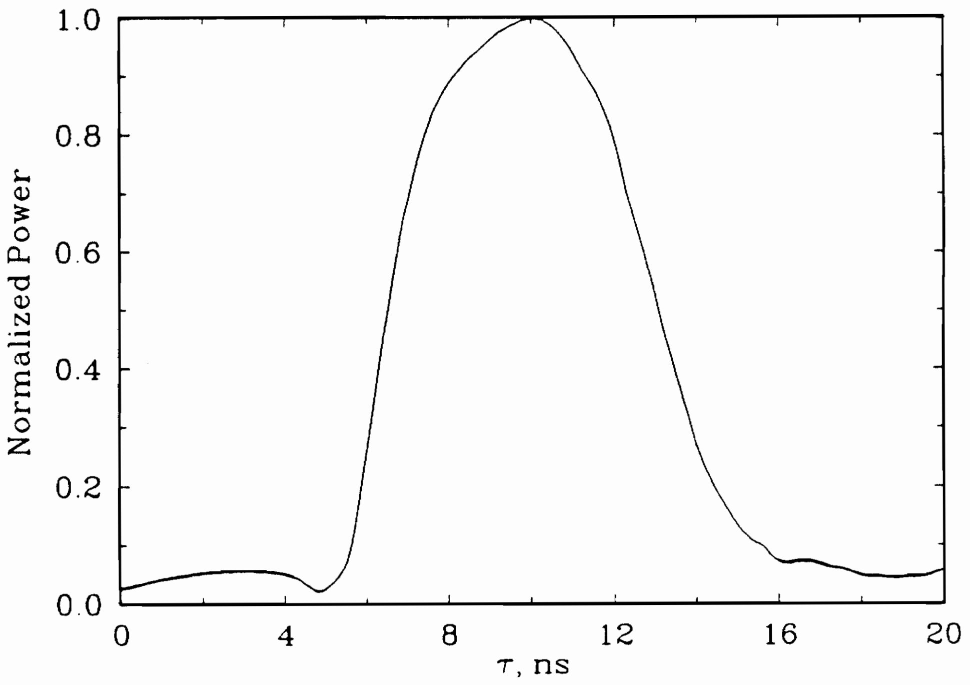


**Figure 3.1.** Extended Interaction Klystron output pulse [26] at 0.1 ns sampling, which can be taken as the radar system point target response.

amplitude of $P_{PT}(\tau)$, which is the peak transmitted power. To relate the half-power width of $P_{PT}(\tau)$ to $\sigma_p$, we set $P_{PT}(\tau)/\widehat{P}_t = 0.5$, $\mu_t = 0$ (center the function) and solve for $\sigma_p$;

$$\sigma_p = \sqrt{\frac{\tau_{hp}^2}{-2\ ln(1/2)}} \tag{3.2-2}$$

where $\tau_{hp}$ is equal to one-half of the 3 dB width of $P_{PT}(\tau)$. Figure 3.2 shows the EIK point target response representation seen in Figure 3.1 overlaid with a Gaussian function having a half-power width of 6.55 ns, or $\sigma_p = 2.78$ ns. The Gaussian is indeed a reasonable approximation to the measured $P_{PT}(\tau)$.

The Gaussian representation will be useful in testing the convolution algorithms that will be discussed in Section 3.4, while the measured data that represents $P_{PT}(\tau)$ will be used for the actual computations of $P_R(\tau)$.

## 3.3 THE SEA STATE DISTRIBUTION

The sea state function, which is more accurately called the probability density function of the height of the specular points on the ocean surface, will be represented by the Gaussian distribution [3,9];

$$q(\tau) = \frac{1}{\sqrt{2\pi}\sigma_q}\ exp\left\{-\frac{1}{2}\left(\frac{\tau - \mu_q}{\sigma_q}\right)^2\right\} \tag{3.3-1}$$

where $\mu_q$ and $\sigma_q$ are the mean and standard deviation of $q(\tau)$. Note that (3.3-1) is

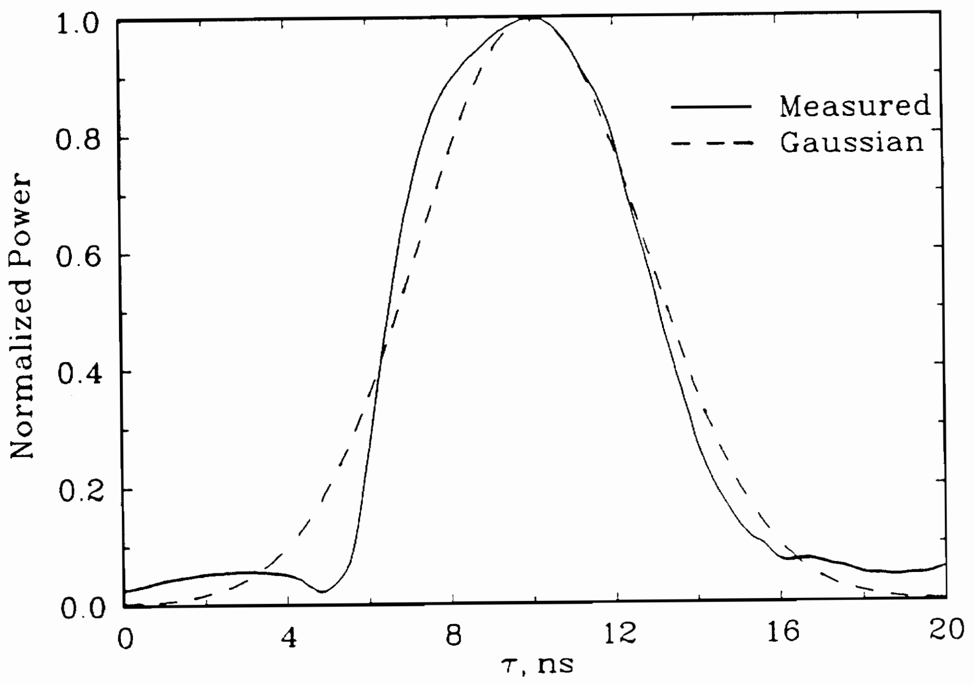


**Figure 3.2.** EIK output pulse [26] with overlaid Gaussian approximation.

a true unit area Gaussian, where (3.2-1) is Gaussian shaped, having peak amplitude $\hat{P}_t$. To find $\sigma_q$, the rms height of the specular points relative to the mean sea level, or $\sigma_s$, must be converted from a spatial quantity to an 'equivalent' temporal quantity using the relation [3,9];

$$\sigma_q = 2\,\frac{\sigma_s}{c} \tag{3.3-2}$$

where $c$ is the speed of light in free space, 0.3 m/ns. The use of a Gaussian to approximate the sea state is a reasonable choice and does not incur any appreciable error [3].

## 3.4 CONVOLUTION OF THE SEA STATE AND POINT TARGET RESPONSES

With both $P_{PT}(\tau)$ and $q(\tau)$ defined, these now can be convolved and stored for subsequent convolution with $P_{FS}(\tau)$. Of particular interest is the case where $P_{PT}(\tau)$ and $q(\tau)$ are both represented by Gaussian-type functions. This will provide a test case for the numerical convolution algorithms that will be developed in Chapter 4.

For the case where $P_{PT}(\tau)$ is non-Gaussian, the numerical convolution algorithms must be used to obtain the combined response. As will be seen in the following chapters, the most efficient way to convolve these two functions is by using the FFT convolution method described in Section 4.2. This is the most accurate and rapid method, which will become important as the time increments become very small to achieve reasonable results.

## 3.4.1 GAUSSIAN CONVOLUTION

From probability theory [21], the convolution of two Gaussian functions having individual variances $\sigma_p^2$ and $\sigma_q^2$ is a Gaussian function whose variance is the sum of the individual variances, i.e.;

$$\sigma_t^2 = \sigma_p^2 + \sigma_q^2 \,. \tag{3.4-1}$$

It must again be noted that (3.2-1) is not a true Gaussian distribution since its amplitude is specified to be $\widehat{P}_T$. However, the convolution of (3.2-1) and (3.3-1) can be performed analytically to obtain;

$$r(\tau) = \frac{\widehat{P}_T \sigma_p}{\sigma_t}\, exp\left\{ -\frac{1}{2}\left(\frac{\tau - (\mu_q + \mu_p)}{\sigma_t}\right)^2 \right\} \tag{3.4-2}$$

where $\sigma_t$ is defined by (3.4-1). Thus, $r(\tau)$ is also Gaussian-shaped, but its amplitude is modified such that it is not a true unit area Gaussian distribution.

As stated previously, this result provides a convenient test case for the numerical convolution algorithms that will be developed in Chapter 4. The result of the numerical convolution of (3.2-1) and (3.3-1) can be compared to (3.4-2) to validate the algorithms. Once this is checked and the numerical convolution algorithms are shown to be correct, the convolution of the actual $P_{PT}(\tau)$ with $q(\tau)$

can be performed with confidence in the results.

With the combined sea state and radar system point target response derived for both the test case (of both being Gaussian) and the more realistic case where the point target response is represented by measured data, what remains to be done is to convolve one or the other with the flat surface impulse response computed by one of the methods described in Chapter 2. It is this convolution that must be investigated next.

# 4.0 CONVOLUTION ALGORITHMS

An important part of this research effort lies in the convolution methods used to complete the computation of $P_R(\tau)$ from the individual functions discussed in Chapters 2 and 3. The simplest of these methods has already been developed in Section 3.4, where Gaussian convolution [6] was used to obtain the combined sea state/point target response $r(\tau)$, when each was a Gaussian function. However, not all of the functions that are encountered are so trivial; in general, each of the functions $P_{FS}(\tau)$, $P_{PT}(\tau)$ and $q(\tau)$ will be smoothly varying, but not ideal Gaussian distributions. Each will be evaluated at discrete points in time $\tau$, which must then be convolved numerically to obtain the average return power signal, $P_R(\tau)$.

Two methods - numerical integration of the convolution integral and Fourier transform convolution - are discussed in this chapter. It will be seen that the numerical integration, while straightforward, is quite time consuming. The algorithm that makes use of Fourier transform techniques turns out to be highly accurate as well as very rapid, making this method the obvious choice for full-time use. It is essential, however, that the two methods provide a mutual validation, assuring accurate results.

## 4.1 NUMERICAL INTEGRATION OF THE CONVOLUTION INTEGRAL

The most straightforward approach to evaluating the convolution integral is by numerical integration. Consider the definition of the convolution integral [6] applied to this problem;

$$P_R(\tau) = \int_{-\infty}^{\infty} P_{FS}(t)\, r(\tau - t)\, dt = P_{FS}(\tau) * r(\tau) \tag{4.1-1}$$

where $r(\tau)$ is itself the convolution of $P_{PT}(\tau)$ and $q(\tau)$, as discussed in Chapter 3. Since the functions describing $P_{FS}(\tau)$, $P_{PT}(\tau)$ and $q(\tau)$ are not easily convolved by analytical methods, (4.1-1) must be evaluated numerically. Performing this operation amounts to a shift, sum, shift, sum, etc. process. To evaluate (4.1-1), both of the functions $P_{FS}(\tau)$ and $r(\tau)$ can be evaluated at discrete, evenly spaced values of time and stored in an array for the impending convolution;

$$\begin{aligned} \bar{P}_{FS}(i) &= P_{FS}(\tau_{o_1} + iT), \quad i = 1, \ldots, k \\ \bar{r}(j) &= r(\tau_{o_2} + jT), \quad j = 1, \ldots, l \end{aligned} \tag{4.1-2}$$

where $T$ is the time interval, or the sampling interval, between discrete values of both $\bar{P}_{FS}$ and $\bar{r}$, and $\tau_{o_1}$ and $\tau_{o_2}$ are the time values where the sampling of the functions are to be started. The convolution of these two arrays can be performed by evaluating the expression;

$$s_j(i) = \overline{P}_{FS}(j-i)\,\overline{r}(i), \quad i = 1,\ldots,j, \quad j = 1,\ldots,k+l \tag{4.1-3}$$

and then integrating for each $j$, the shift variable, thus accomplishing the shift, integrate, shift, integrate, etc. operation that is inherent to numerical convolutions. The resulting series of integrations gives;

$$\overline{P}_R(j) = \int_i s_j(i)\,di, \quad j = 1,\ldots,k+l. \tag{4.1-4}$$

$\overline{P}_R(j)$ is the discrete representation of $P_R(\tau)$, whose accuracy depends on the accuracy of the integration method and sampling interval $T$. The integration can be performed by using the extended trapezoidal rule [1]. Although this is a very simplistic integration method, it is just that which makes the trapezoidal rule particularly suited to this application. Note that as $j$ is incremented, the number of samples in $s$ that must be integrated alternates between odd and even as that number increases. For many advanced numerical integration techniques, such as Gaussian integration [1], the sampled points are not, in general, evenly spaced. Compounding the problem, most integration formulas do not allow for a changing number of integration points. Even the extended Simpson's rule [1] always requires an odd number of samples. Thus, the trapezoidal rule is most useful since it doesn't restrict the number of samples, and the samples are evenly spaced.

The disadvantage to numerically integrating the convolution integral is that if $k$ and $l$ become large, this process becomes very time consuming. This will indeed happen if the sampling interval is desired to be small, increasing the number of samples, $m$ and $n$, in each function $\overline{P}_{FS}$ and $\overline{r}$. Unfortunately, a small sampling increment is necessary in order to obtain an accurate representation for $P_R(\tau)$. Thi is especially important at small pointing angles, since the FSIR decays so rapidly (see Section 5.1.1). Thus a trade off arises: high accuracy with time consuming operations or poor accuracy with rapid computation. Fortunately, other methods exist to evaluate the convolution of two functions.

## 4.2 COMPUTING THE CONVOLUTION INTEGRAL USING FOURIER TRANSFORMS

Another widely used and more readily applicable approach to computing the convolution of two functions is through the use of Fourier transform techniques. Consider the definition of the Fourier transform [6] applied to $P_{FS}(\tau)$;

$$\tilde{P}_{FS}(f) = \int_{-\infty}^{\infty} P_{FS}(\tau)\, e^{-j2\pi f\tau}\, d\tau \tag{4.2-1}$$

$$P_{FS}(\tau) = \int_{-\infty}^{\infty} \tilde{P}_{FS}(f)\, e^{j2\pi f\tau}\, df. \tag{4.2-2}$$

(4.2-1) is called the *forward* Fourier transform, while (4.2-2) is called the *inverse* Fourier transform and $\tilde{P}_{FS}(f)$ is said to be the frequency spectrum of $P_{FS}(\tau)$. The convolution theorem states [6] that if two functions, for convenience say $P_{FS}(\tau)$ and $r(\tau)$, have Fourier transforms given by $\tilde{P}_{FS}(f)$ and $R(f)$, respectively, then the convolution of the two can be found from;

$$P_R(\tau) = \mathfrak{F}^{-1}\Big\{ \tilde{P}_{FS}(f)\, R(f)\Big\} \qquad (4.2\text{-}3)$$

where $\mathfrak{F}^{-1}\{\cdot\}$ is the inverse Fourier transform operator. Thus, if the Fourier transforms of $P_{FS}(\tau)$ and $r(\tau)$ can be found, their convolution reduces to an inverse Fourier transform of a simple point-by-point multiplication in the frequency domain given by (4.2-3), provided the inverse Fourier transform of the product can also be found [6].

Fourier transforms of simple functions are quite trivial and are listed in numerous sources [5,6,7], but for more complicated functions, such as those used in this research, their Fourier transforms are much more difficult to obtain. Consider using (4.2-1) to obtain the transform of $P_{FS}(\tau)$ given by (2.3-3). It is easy to see that carrying out this integration analytically would be cumbersome and time consuming, assuming a closed form exists. For this reason, the discrete Fourier transform (DFT), the discrete equivalent of the Fourier transform, was developed [7] to compute the Fourier transform of discrete data. Although directly applicable to this problem, the DFT is still quite a time consuming operation when the number of sample points become large, as will be encountered in this scenario.

In the early 1960's, Cooley and Tukey [7] developed the first and now-famous fast Fourier transform (FFT) algorithm for digital computers. The FFT takes advantage of certain symmetry conditions to compute DFT's very rapidly. The computational speed of these FFT algorithms are best appreciated when the number of samples becomes large. On digital computers, a good estimate of the amount of time an algorithm will take to perform a task is given by the number of multiplications that the algorithm will require. As a rule of thumb [7], if $N$ is assumed to be the number of data points that represent a given function, then the DFT algorithm requires $N^2$ multiplications to compute the Fourier transform of that data, where the FFT algorithm requires only $N log_2 N$ multiplications. Figure 4.1 illustrates the difference in the number of multiplications required to compute DFT's and FFT's for a given $N$. For the application at hand, the base-2 Cooley-Tukey algorithm will be used for the large $N$ that will be encountered, where $N$ is required to be an integer power of 2, i.e. $N = 2^m$ , $m$ an integer. This requirement allows for the development of the most efficient FFT algorithm. The large values of $N$ that will be used (e.g. 2048, 4096, 8192) will be necessary for good accuracy in the results, especially for small pointing angles.

One of the disadvantages of the FFT convolution is that the algorithm assumes that the data to be transformed are periodic [7]. For this reason, the data must be *zero-padded*, or augmented with zeros, to eliminate *end*, or *overlap effects* [7]. This effect creates additional energy in the convolved result because the FFT algorithm assumes that the data in the array can be continuously strung together to reconstruct a periodic signal. In order to prevent this phenomenon, the following constraints on $N$ must be strictly enforced:

$$\begin{aligned} &N \geq P + Q - 1 \\ &N = 2^m, \quad m \text{ an integer.} \end{aligned} \tag{4.2-4}$$

$P$ and $Q$ are the number of data points that represent the two functions that are to be transformed, analogous to use of $k$ and $l$ in Section 4.1. For example, if the

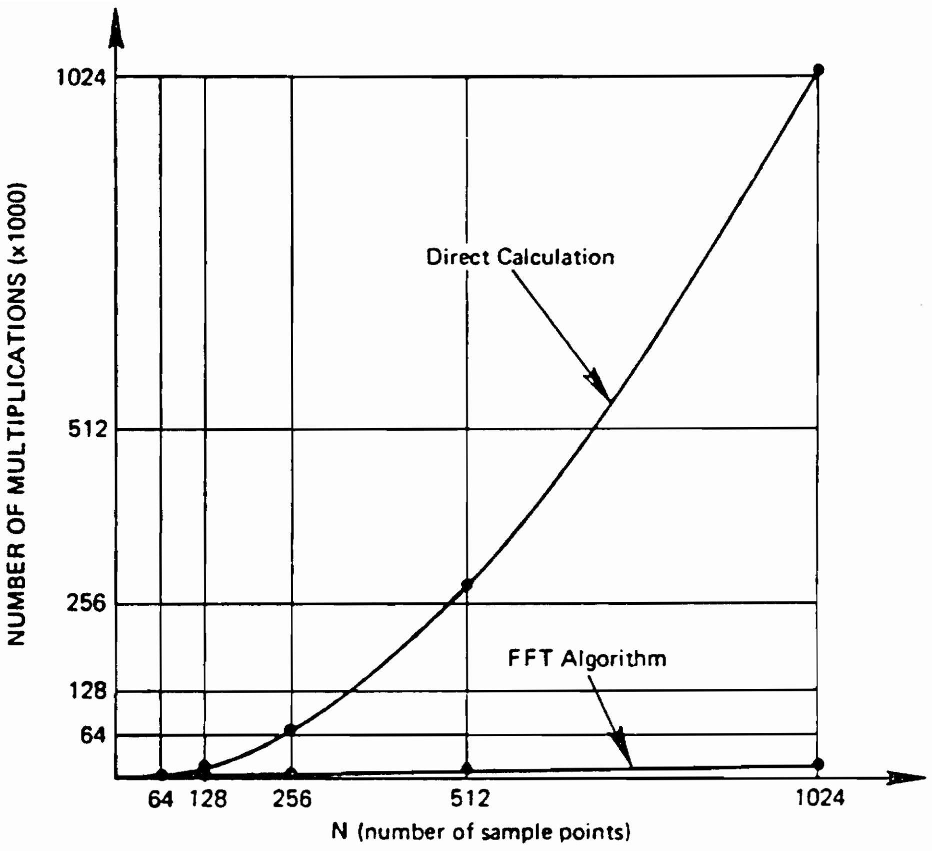


Figure 4.1. Comparison of the number of multiplications required by the DFT and FFT methods [7].

functions $P_{FS}$ and $r$ were represented by 15 and 25 samples, respectively, then according to the first of eqs. (4.2-4), $N$ would have to be greater than $15+25-1=39$. The next highest power of two is 64, so both data sets would have to have enough zeros added to make the total number of samples for each abide by (4.2-4). If this condition is observed, the resulting convolution will not be tainted by end effects.

Another important feature of the FFT is that it can compute both the forward and inverse Fourier transforms without any modification to the algorithm. All that is required is a slight modification to the input data in order for the inverse Fourier transform to be performed. Specifically, the input data must first be conjugated before the FFT algorithm can perform the inverse transform. For example, if a function $x(t)$ were FFTed to obtain $X(f)$, the array need only be conjugated and FFTed again to retrieve $x(t)$. This time- and code size-saving capability serves to make the FFT an even more valuable tool for this research.

## *4.2.1 FFT IMPLEMENTATION*

In the previous chapters, expressions were developed for $P_{FS}(\tau)$, $P_{PT}(\tau)$ and $q(\tau)$. These functions are then evaluated at discrete values of $\tau$, which are equally spaced by some time increment $T$. These data sets are then placed in complex arrays (a requirement for the FFT algorithm [7]) with the data occupying the real portion of the array and the imaginary portion set to zero, since the functions of interest are real valued. Then the two arrays which are to be convolved, say $P_{PT}(\tau)$ and $q(\tau)$, are each FFTed to yield $\tilde{P}_{PT}(f)$ and $Q(f)$, respectively;

$$\begin{aligned} \tilde{P}_{PT}(f) &= FFT\{P_{PT}(\tau)\} \\ Q(f) &= FFT\{q(\tau)\}. \end{aligned} \tag{4.2-5}$$

Then the two complex arrays $\tilde{P}_{PT}(f)$ and $Q(f)$ are multiplied together to yield $R(f)$. Next $R(f)$ is conjugated to allow the FFT algorithm to perform an inverse transform, then FFTed, resulting in a scaled representation for $r(\tau)$. To obtain the true $r(\tau)$, the FFT output must be divided by $N$, the number of data points in the FFT, then multiplied by $T$, the sampling width [7]. In terms of (4.2-5):

$$r(\tau) = \frac{T}{N}\, FFT\{\, (\tilde{P}_{PT}(f) \cdot Q(f))^* \,\}. \tag{4.2-6}$$

### 4.2.2 DISCONTINUITIES IN TIME RESPONSES

The FFT convolution method works quite well for all but a handful of cases. When either of the two signals that are to be convolved has a time discontinuity, the FFT can give erroneous results. However, the same FFT process can be applied to discontinuous functions if certain precautions are taken. An example of a discontinuous function that is of particular interest can be found in (2.2-9). This form of $P_{FS}(\tau)$ is zero for $\tau < 0$, and behaves like a decaying exponential for $\tau \geq 0$. It is this discontinuity a $\tau = 0$ that presents a problem.

Fourier analysis states [6] that a function with a time discontinuity, such as (2.2-9), will have an infinite energy spectrum. This means that a large portion of the energy of that function is contained in very high frequencies. Since the FFT has limited bandwidth for a finite $N$ (it cannot represent infinite frequency) [7], the discontinuity must be accounted for before the FFT is used. The remedy to this problem is found in [7]. For any discontinuity, that point should be represented by the average value of the discontinuity, i.e.

$$P_{FS}(0)\Big|_{\xi = 0^\circ} = \tfrac{1}{2}\left[P_{FS}(0^-) + P_{FS}(0^+)\right]. \tag{4.2-7}$$

Although (4.2-7) was applied at $\tau = 0$ for (2.2-9), this rule applies to any discontinuity at any value of $\tau$. When this condition is enforced, the infinite frequency spectrum is eliminated and the FFT yields the correct transform.

## 4.3 COMPARISON OF COMPUTATION TIMES FOR THE TWO CONVOLUTION METHODS

To illustrate the difference in time required by the numerical integration and FFT convolution algorithms, the computation of the average return power waveform for pointing angles of both $\xi = 0^\circ$ and $\xi = 12^\circ$ were performed using each convolution method. For the pointing angle $\xi = 0^\circ$, the sample spacing of $T = 0.01$ ns is required to accurately represent the FSIR. This sampling interval is the smallest that will be necessary since the FSIR for other pointing angles will have a wider time width (see Sections 5.1.2 and 5.1.3) and be more smoothly varying. This interval requires $N = 4096$ for the FFT algorithm (see Section 4.2) and $k + l = 4201$ for the numerical integration method (see Section 4.1). The resulting computing times were about 1.6 minutes for the FFT algorithm compared to about 13.4 minutes for the numerical integration computation, a difference by a factor of more than 8 times.

This case of $\xi = 0^\circ$ turns out to be a worst-case scenario; larger pointing angles will not require the fine sampling interval (or equivalently, large arrays) to obtain accurate results. As an example, the computation of the average return power for a pointing angle of $\xi = 12^\circ$ only requires a sampling interval of 0.1 ns, which requires $N = 2048$ for the FFT method and $k + l = 1501$ for the numerical integration method. The resulting computation times are 0.75 minutes for the FFT method and 1.8 minutes for the numerical integration, a little more than 3 times larger. While not as great as for $\xi = 0^\circ$, the time difference between the two methods is considerable.

Even though the two methods provide the same results for all pointing angles, it is easy to see that as the returns are computed using the numerical integration convolution algorithm at several pointing angles, the total amount of time required for these computations skyrockets when compared to total time for the FFT routine. For this reason it is obvious that the FFT method is more desirable for use in evaluating the required convolutions.

A reliable method is thus available to rapidly compute the convolution of two arbitrary functions represented by discrete data points. The FFT method is both highly accurate and very quick in computing the convolution, so much more than the numerical integration method that it is the logical choice for this application. It is important, however, to apply this method with caution to functions that show a discontinuity, such as the decaying exponential that approximated the flat surface impulse response for a pointing angle of zero.

# 5.0 ALGORITHM TESTING AND NUMERICAL RESULTS

An important part of any research effort that involves writing computer codes is the validation of the resulting algorithms. In this case, two basic questions need to be addressed. First, does the flat surface impulse response algorithm, with its three different evaluation methods, agree with expected results? Secondly, do the numerical convolution codes give accurate results? Both of these issues are addressed by developing test cases that utilize portions of the developed code that can be compared with analytical methods to check their validity.

The computer codes that were written during this research effort are listed in Appendix A, along with some of the more important aspects of their logic. The main program that manages all of the individual subroutine calls and input/output (I/O) is called the Return Power Master (RPM) program. It calls several subroutines including: PFSIR, which decides which of the three methods derived in Chapter 2 to use in computing the FSIR; DCOMP, which computes the Gaussian function that represents the sea state and convolves it with either the data file or Gaussian representation of the point target response; CONVI /CONVF, two subroutines that perform the numerical integration or FFT convolutions; FFT, the FFT algorithm; and INOT, a Bessel function evaluation algorithm.

Some example computations performed by RPM are given in Section 5.3 which are typical of the situations encountered with the MARA system. Comparison is also made with some early signal-to-noise calculations made for the MARA system [27], and a preliminary investigation of pointing angle determination from the modeled returns is made.

## 5.1 FSIR TESTS

The first algorithm that will be examined is that which performs the computation of the flat surface impulse response. As mentioned above and in Chapter 2, there are three equations that govern the behavior of the FSIR: (2.2-15) for $\xi = 0^\circ$, (2.3-1) for $\xi$ near nadir and (2.4-3) for $\xi$ large. The most obvious test is to see if the shape and amplitude of $P_{FS}(\tau)$ behave as expected as the pointing angle $\xi$ increases from $0^\circ$ to $12^\circ$.

### *5.1.1 FSIR AT NADIR*

Examining (2.2-15) reveals that when the pointing angle $\xi$ is zero, this function is zero for $\tau < 0$ and a decaying exponential behavior modified by a Bessel function for $\tau \geq 0$. Thus, a plot of this function should have some peak value at $\tau = 0$ and decay to zero essentially as an exponential as $\tau$ increases. Figure 5.1 shows the FSIR for the MARA system at nadir, with $\theta_s = 0.6313^\circ$ and $\theta_{xs} = 0.6300^\circ$. The radar altitude is set to 3048 meters (10,000 feet), $\sigma^o = 2$ dB (a reasonable value for $\xi = 0^\circ$ [27]), sampling interval $T = 0.02$ ns and $L_P = 0$ dB. The figure shows that the function decays to zero very rapidly as expected since the beamwidth is so narrow.

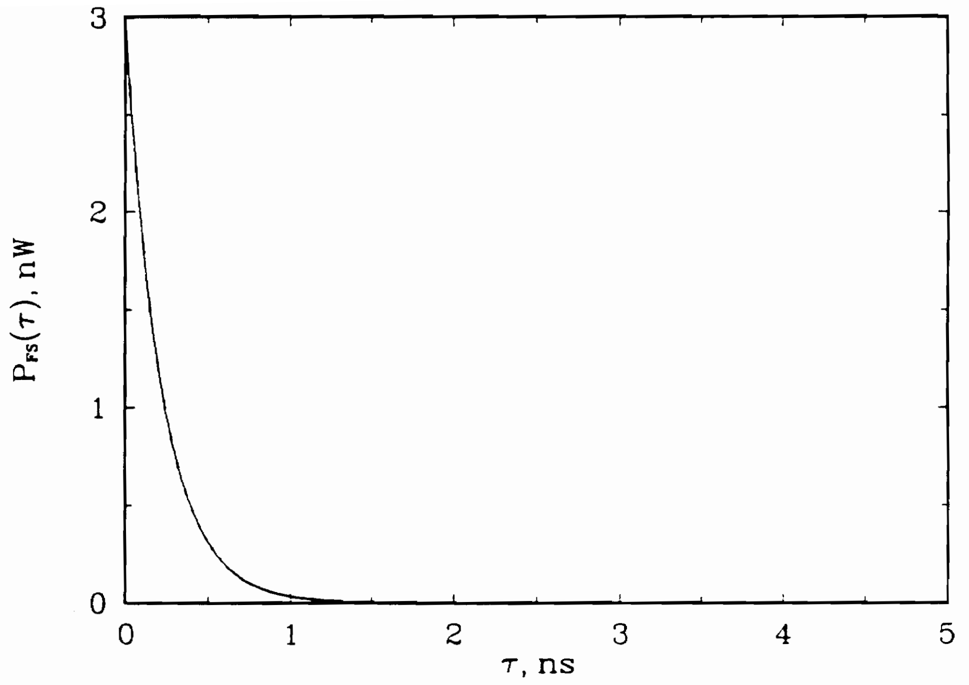


**Figure 5.1.** Flat surface impulse response for $\xi = 0^\circ$, with $h = 3048$ m, $T = 0.02$ ns and $\sigma^o = 2$ dB.

## 5.1.2 FSIR AT SMALL POINTING ANGLES

At small pointing angles, were $\xi$ is on the order of a beamwidth, the exact numerical integration form of $P_{FS}(\tau)$ in (2.3-1) is used out to some value of $\tau$ given by either (2.4-5) or (2.4-7) and then the asymptotic form in (2.4-3) takes over. An important test lies in verifying that as the evaluation switches from one method to the other, the result is continuous. Consider Figure 5.2, where the pointing angle was chosen to be $\xi = 0.30^\circ$. At this angle, $\theta_s = 0.6313^\circ$ and $\theta_{xs} = 0.6255^\circ$. According to (2.4-5), when $\tau \geq 2.047$ ns (the more conservative condition, in this case) (2.4-3) can be used. In Figure 5.2 a sample spacing of 0.05 ns was used, so the transition takes place at $\tau = 2.05$ ns. The transition is indeed smooth, showing that the algorithm performs well in this aspect.

Another related test is to see if the asymptotic form for $P_{FS}(\tau)$ given by (2.4-3) gives the same results as the exact numerical integration in (2.3-1) after the value of $\tau$ given by the appropriate condition, (2.4-5) or (2.4-7). Using the same parameters as were used to obtain Figure 5.2, the resulting waveforms were virtually the same after $\tau = 2.05$ ns. This demonstrates that the asymptotic evaluation is being used at the appropriate time, resulting in less than 2% error.

## 5.1.3 FSIR AS THE POINTING ANGLE INCREASES

As was seen in Section 5.1.1, the shape of $P_{FS}(\tau)$, when $\xi = 0^\circ$, is a

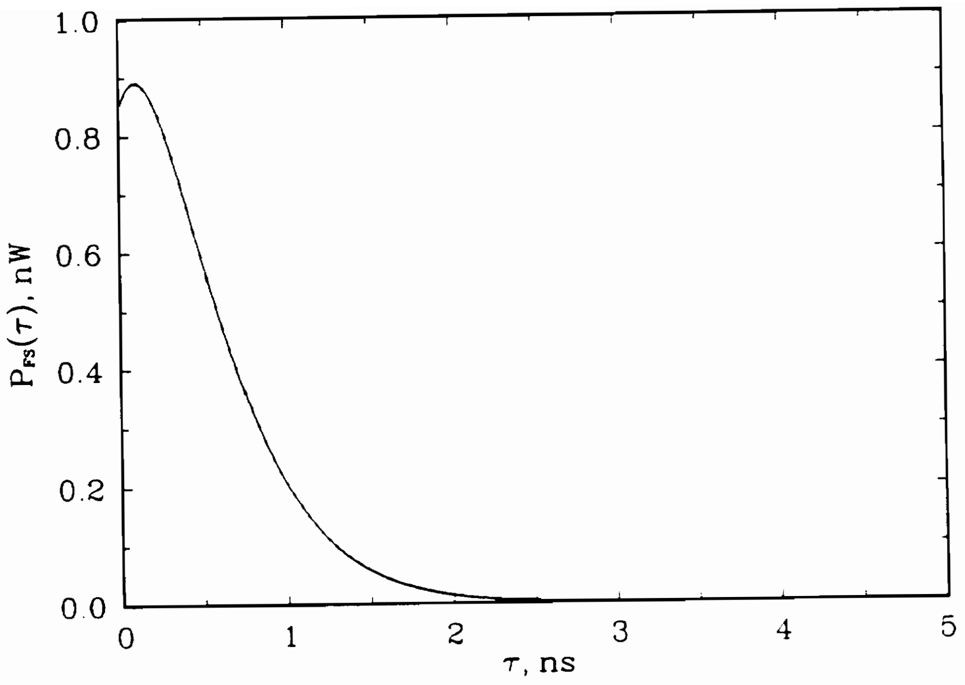


**Figure 5.2.** Flat surface impulse response for $\xi = 0.3^{\circ}$, with $h = 3048$ m, $T = 0.05$ ns and $\sigma^o = 2$ dB.

decaying exponential that is slightly modified by a Bessel function, as shown in Figure 5.1. Figure 5.2 contained $P_{FS}(\tau)$ for $\xi = 0.3^\circ$, with the other parameters the same, except for the cross-scan beamwidth, $\theta_{xs}$. This figure seems to resemble the decaying exponential in Figure 5.1, but delayed and modified somewhat. In fact, as the pointing angle continues to increase, the FSIR tends to be 'smeared' in $\tau$.

Figure 5.3 shows the $P_{FS}(\tau)$ functions for pointing angles of 2, 4, 6, 8, 10 and 12 degrees. For this figure, the beamwidths are asymmetric, behaving as in Figure 2.4, $\sigma^o$ is taken to be constant over the range of $\xi$ and the signal levels are each normalized to unity so that the delay and smearing effects can be seen. The peak FSIR levels actually behave much like the peak FSIR, $\widehat{P}_{FS}$, in Figure 5.6(a) in the next section. Thus, as the pointing angle increases, the FSIR does indeed widen, or smear, and becomes more delayed in time.

This phenomenon can be explained by considering the geometry of Figure 5.4. When an impulse is transmitted toward nadir, as the FSIR implies, the energy traveling in a spherical shell reaches the mean flat surface and first illuminates a point; the spherical shell then intersects the surface creating an annular ring that increases in radius as the impulse travels away from the radar. Thus the first energy returned to the receiver is an impulse, and the remaining energy returns slightly delayed and tailing off toward zero due to the antenna pattern. When the pointing angle increases, as seen in the figure, the first energy strikes the surface, not at boresight, but at the nadir point where the antenna pattern will reduce the signal level. The peak return comes when the impulse strikes at the boresight intersection with the surface, and then the return trails off once again due to the antenna pattern. Thus, the smearing increases as the

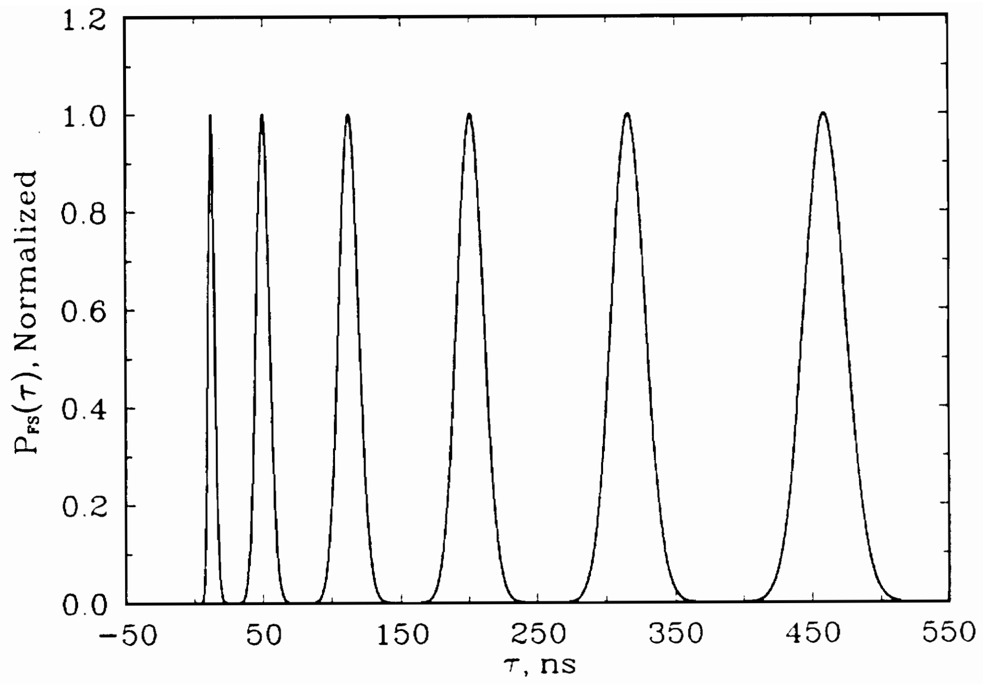


**Figure 5.3.** Flat surface impulse response for pointing angles of 2, 4, 6, 8, 10, and 12 degrees, illustrating the smearing and delay effects. $T = 0.1$ ns.

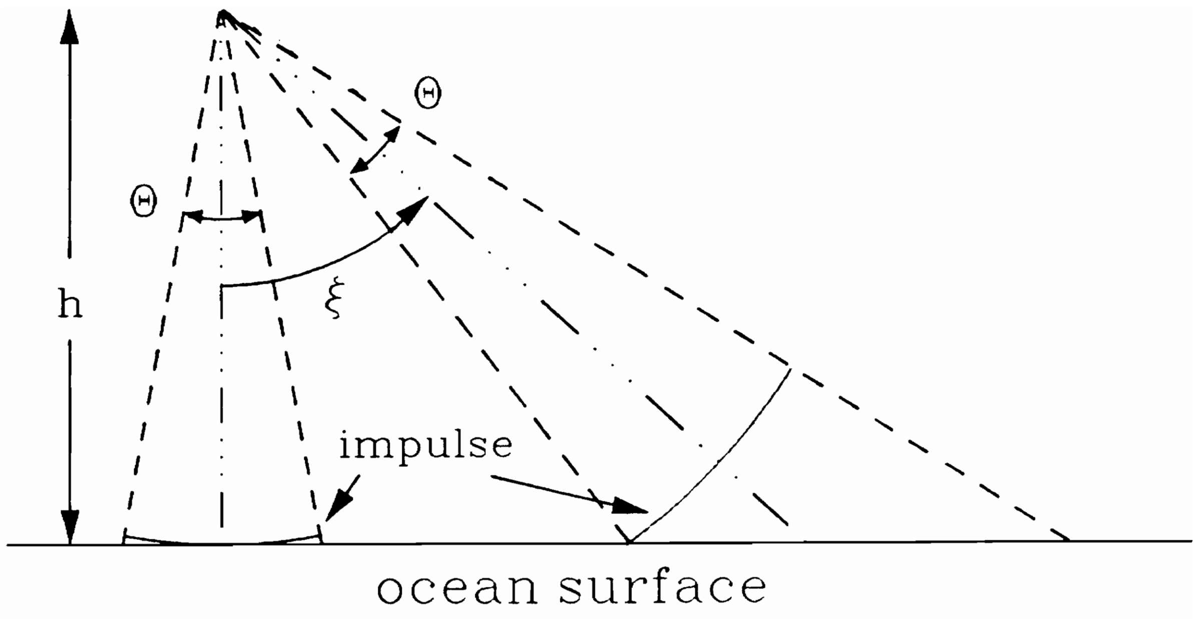


**Figure 5.4.** Illustration of the smearing effect of large pointing angles on the flat surface impulse response.

pointing angle becomes larger, creating a widened and delayed return signal. This then directly affects the resulting $P_R(\tau)$ waveforms through the convolution.

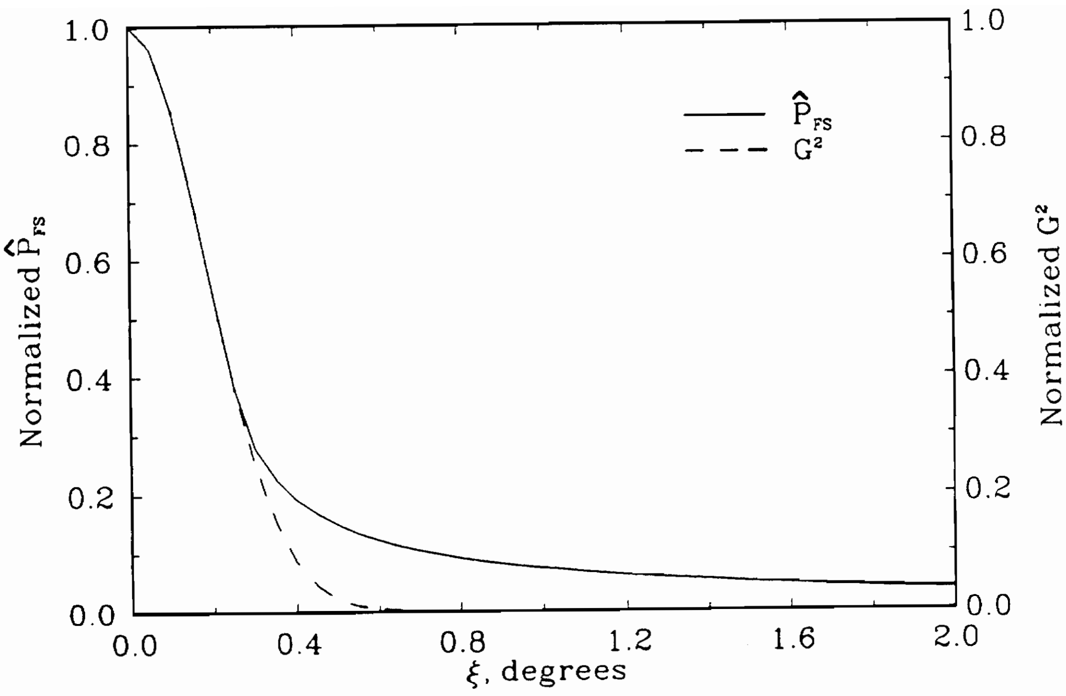
## 5.1.4 FSIR PEAK LEVEL

The FSIR peak level occurs when the transmitted spherical wavefront intersects the surface at the point where the antenna boresight axis intersects the mean flat surface [9]. Brown [9] showed that for small pointing angles (relative to the beamwidth), the peak FSIR, or $\widehat{P}_{FS}$, behaves like the two-way antenna pattern, or;

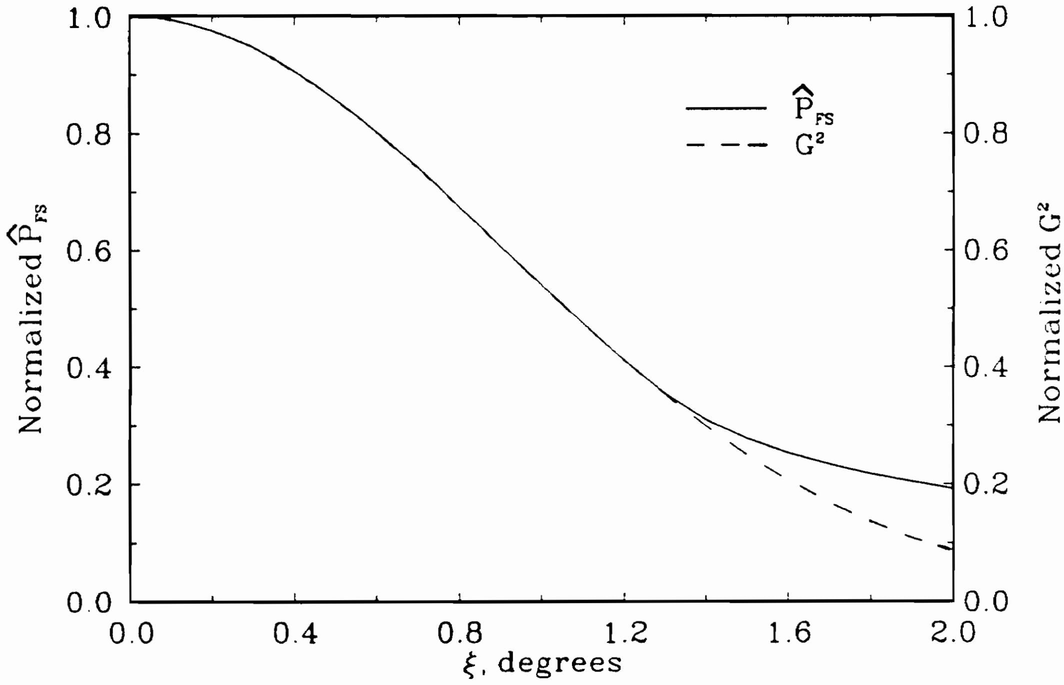
$$\widehat{P}_{FS} \sim \sigma_o(\xi)\, e^{-\frac{4}{\gamma}\sin^2\xi}. \tag{5.1-1}$$

This was for the circularly symmetric antenna pattern case. The algorithms developed here are capable of computing the FSIR for the equal beamwidth case when $\beta = 0$ in any of the equations. Figure 5.5(a) shows the comparison of $\widehat{P}_{FS}$ to $G^2(\xi)$ as a function of $\xi$ with the beamwidth set to 0.60°, while Figure 5.5(b) shows the same comparison for a beamwidth of 3.0°. In both figures, $\sigma^o(\xi) = 0$ dB (constant in $\xi$), and $\widehat{P}_{FS}$ tracks the $G^2$ curve exactly until $\xi$ is about half the beamwidth.

It was also shown in [9] that when the pointing angle is large relative to the beamwidth, the $\widehat{P}_{FS}$ behaves like;



(a)



(b)

**Figure 5.5.** Peak flat surface impulse response and two-way gain vs. pointing angle for (a) 0.6° beamwidth and (b) 3.0° beamwidth.

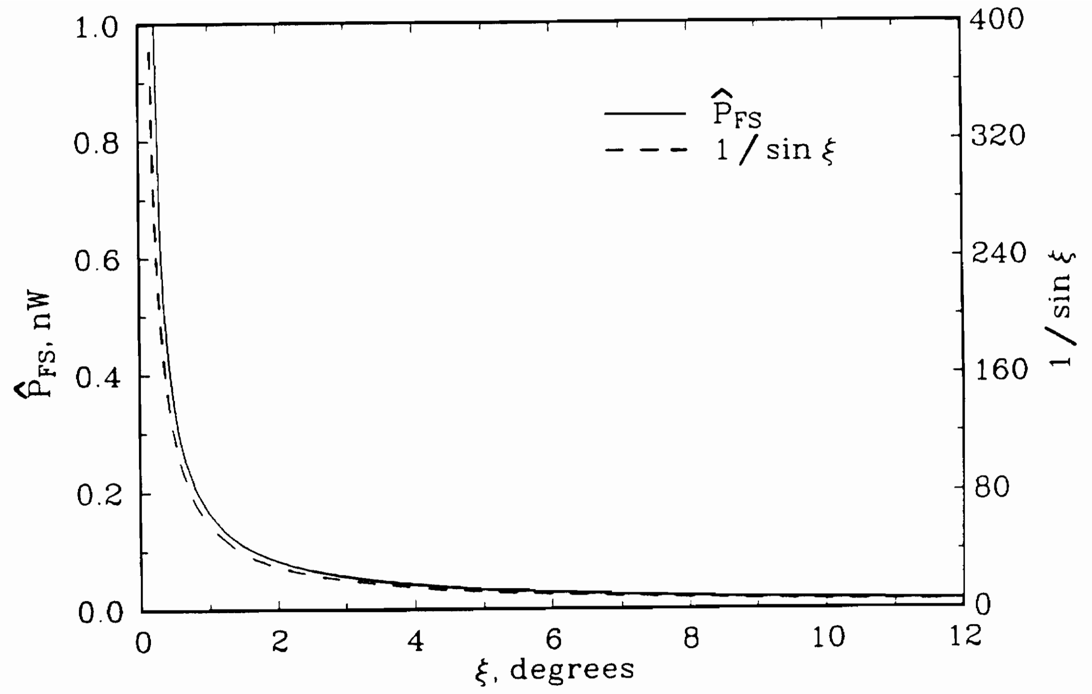
$$\widehat{P}_{FS} \sim \frac{\sigma^o(\xi)}{sin\ \xi}. \tag{5.1-2}$$

As an example, Figure 5.6(a) compares $\widehat{P}_{FS}$ to the ratio $1/sin\ \xi$ for a symmetric beamwidth of 0.6° and Figure 5.6(b) compares the same with the beamwidth set to 3.0°, with $\sigma^o(\xi) = 0$ dB in both. These figures show that as the pointing angle becomes large, the peak FSIR does indeed tend to follow $1/sin\ \xi$.

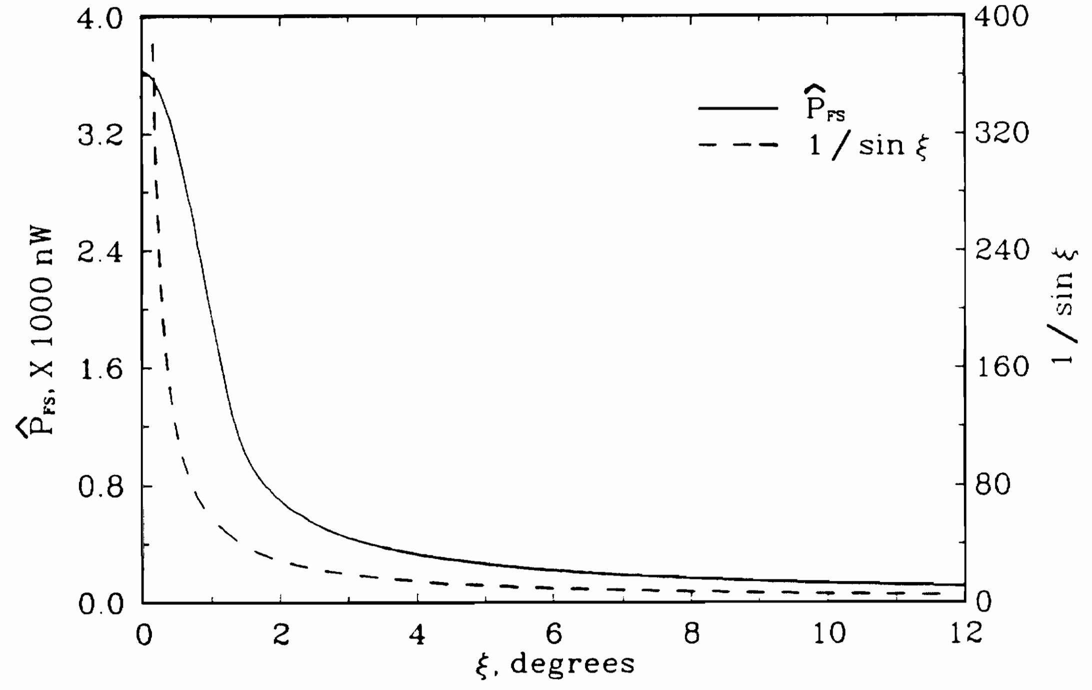
These tests show that the algorithm that generates $P_{FS}(\tau)$ behaves as expected and also shows that the peak return power, $\widehat{P}_R$, will behave significantly differently over the range of $\xi$.

## 5.2 CONVOLUTION TESTS

With the flat surface impulse response algorithm validated, the next logical step in the testing procedure is to test the convolution algorithms. Two convolution subroutines were developed, one to perform the operation using the numerical integration technique, and the other using the Fourier transform method. The two methods allowed for independent checks against one another. A few test cases can be used to see if these methods give correct results.



(a)



(b)

**Figure 5.6.** Peak flat surface impulse response and $1/sin\xi$ vs pointing angle for (a) 0.6° beamwidth and (b) 3.0° beamwidth.

### *5.2.1 GAUSSIAN CONVOLUTION*

One of the standard test cases for these programs lies in the convolution of two Gaussian distributions. This process is detailed in Section 3.4.1 for the case when the sea state function was a true unit Gaussian and the point target response was Gaussian shaped. When both functions are unit Gaussians, i.e., the integration of the function over the limits $(-\infty,\infty)$ is unity, their convolution reduces to the summation of the individual variances as given by (3.4-1). In Section 3.4.1, the amplitude term was given as $\widehat{P}_T\sigma_p/\sigma_t$. However, when both function are unit Gaussians, the resulting convolved function is also a unit Gaussian with the familiar amplitude term $1/\sqrt{2\pi}\sigma_t$, where $\sigma_t$ is given by (3.4-1).

The convolution subroutines that were developed can be used in a test program that convolves two Gaussians of known variance, and the result can be compared to a Gaussian of summed individual variances. When this task was performed, the resulting curves were identical for both methods, indicating that this first test was successful.

### *5.2.2 CONVOLVING AN EXPONENTIAL AND UNIT STEP*

There are some cases where the convolution integral can be evaluated analytically, but only for very simplistic functions, It was seen earlier that the flat surface impulse response very closely resembled an exponential function when the pointing angle was zero. This was for the asymmetric beamwidth case, but when the antenna pattern is circularly symmetric, $\beta = 0$ in (2.2-9) or (2.2-15) and the argument of the Bessel function goes to zero. The evaluation of $I_0(0)$ is unity, so the function $P_{FS}(\tau)$ reduces to a pure exponential with an amplitude term;

$$P_{FS}(\tau) = \frac{G_o^2 \lambda^2 c \sigma^o}{4(4\pi)^2 L_p h^3} \, exp\left\{- \frac{4}{\gamma}\frac{c\tau}{h}\right\}, \quad \tau \geq 0. \tag{5.2-1}$$

In this example it is assumed that $\sigma^o(\psi) = \sigma^o$ (constant).

Although the unit step function does not represent anything physical, it is convenient to use it to represent the combined sea state and point target response for a test convolution because the convolution of an exponential with the unit step can be performed analytically. The unit step is defined as;

$$u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0. \end{cases} \tag{5.2-2}$$

When placed in the convolution integral with $r(\tau) = u(\tau)$;

$$P_R(\tau) = \int_{-\infty}^{\infty} P_{FS}(t) \; u(\tau - t) \; dt. \tag{5.2-3}$$

When the functions are substituted in (5.2-3), $P_R$ becomes;

$$P_R(\tau) = \frac{G_o^2 \lambda^2 c \sigma^o}{4(4\pi)^2 L_p h^3} \int_0^{\tau} exp\left\{- \frac{4}{\gamma}\frac{ct}{h}\right\} dt$$

$$= \frac{G_o^2 \lambda^2 c \sigma^o}{4(4\pi)^2 L_p h^3} \left(- \frac{\gamma h}{4c}\right) \left[ exp\left\{- \frac{4}{\gamma}\frac{ct}{h}\right\} \right]_0^{\tau}$$

$$= \frac{G_o^2 \lambda^2 \sigma^o \gamma}{(16\pi)^2 L_P h^2} \left[ 1 - exp\left\{ - \frac{4}{\gamma} \frac{c\tau}{h} \right\} \right]. \tag{5.2-4}$$

This function can then be evaluated and compared to the two convolution routines for accuracy. Figure 5.7 compares the three methods, where the sampling interval is 0.01 ns, $h = 3048$ m, $G_o = 46$ dB and the symmetric beamwidth is 0.6313°. It should be noted here that when the FFT convolution method is used, the discontinuities found in $P_{FS}(\tau)$ and $u(\tau)$ must be handled as in (4.2-7); i.e. for $\tau = 0$ in both functions, $P_{FS}(0)$ and $u(0)$ must be set to one-half of their true values (see Section 4.2). Note that each curve is indistinguishable from the others, indicating that the algorithms work properly for this case.

## *5.2.3 CONVOLVING AN EXPONENTIAL WITH A GAUSSIAN*

A more realistic test case is when the exponential that was used in the previous section is convolved with a Gaussian function. The Gaussian, as was noted in Sections 3.2 and 3.4, is a very reasonable approximation to the combined sea state and point target response, $r(\tau)$. This test will then yield a reasonable approximation to $P_R(\tau)$ when the pointing angle is zero.

Again consider using (5.2-1) to represent $P_{FS}(\tau)$, the FSIR for a circularly symmetric antenna pattern. For the function $r(\tau)$, (3.4-2) will be used as the Gaussian shaped distribution, with $\mu_q + \mu_p = 4\sigma_t$. This is an artificial mean in that the Gaussian representation begins at the point where the amplitude is $4\sigma_t$

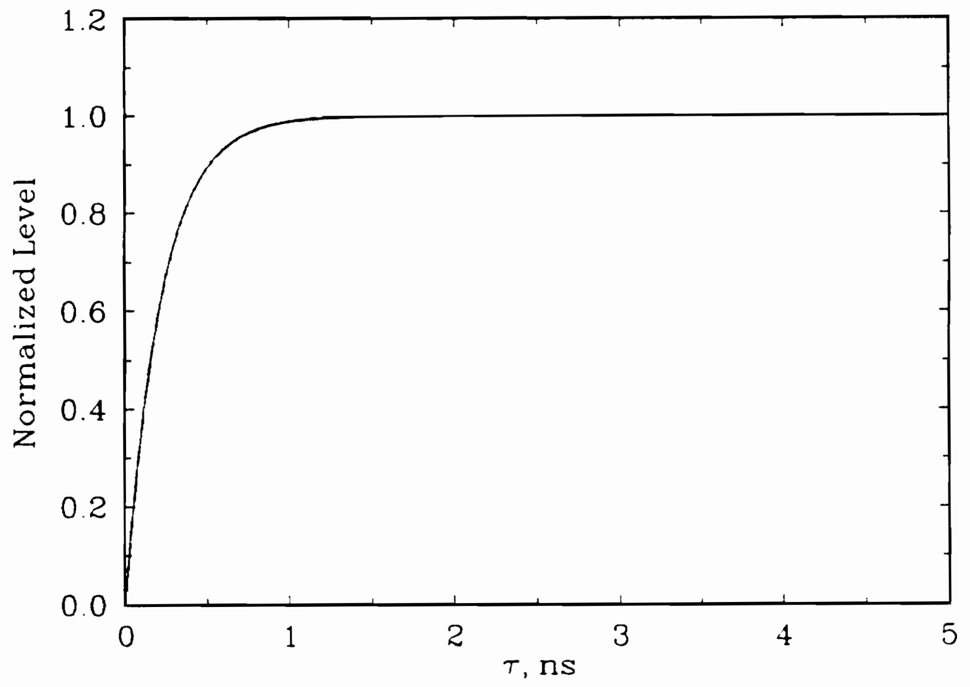


Figure 5.7. Convolution of the flat surface impulse response, represented by an exponential, with the unit step by analytical, numerical integration and FFT methods. The three curves are identical.

down from its maximum, providing a time reference for comparison of all return signals. [1] provides a useful indefinite integration formula to aid in this evaluation;

$$\int exp\left\{-(ax^2+2bx+c)\right\}dx = \tfrac{1}{2}\sqrt{\tfrac{\pi}{a}}\,exp\left\{\frac{b^2-ac}{a}\right\}erf\left\{\sqrt{a}x+\frac{b}{\sqrt{a}}\right\}+C, \qquad (5.2\text{-}5)$$

where $erf(x)$ is the error function. Examination of the integrand reveals how this formula can be used. The $x^2$ term represents the Gaussian behavior of $r(\tau)$, the $x$ term represents the exponential behavior of $P_{FS}(\tau)$, and the constant term represents any left-over cross products that are independent of $x$ or $x^2$, created in formation of the convolution integral. (5.2-1) and (3.4-2) can be written as

$$P_{FS}(\tau) = B\,exp\{2b_1\tau\}, \quad \tau \geq 0 \qquad (5.2\text{-}6)$$

and

$$r(x) = A\,exp\{-ax^2+2b_2x\} \qquad (5.2\text{-}7)$$

where

$$a = \frac{1}{2\sigma_t^2}, \qquad b_1 = -\frac{2c}{\gamma h}, \qquad b_2 = \frac{2}{\sigma_t},$$

$$A = \frac{\widehat{P}_T\sigma_p}{\sigma_t}\,e^{-8}, \qquad B = \frac{G^2\lambda^2c\sigma^o}{4(4\pi)^2L_Ph^3}\,.$$

If (5.2-6) is shifted by $-x$, the result:

$$P_{FS}(\tau - x) = B\,exp\{2b_1\tau - 2b_1 x\}, \;\; \tau - x \geq 0 \tag{5.2-8}$$

can be placed in the convolution integral with (5.2-7):

$$\begin{aligned}
P_R(\tau) &= \int_{-\infty}^{\infty} r(x)\; P_{FS}(\tau - x)\; dx \\
&= AB\int_{-\infty}^{\tau} exp\{-ax^2 + 2b_2 x - 2b_1 x + 2b_1\tau\}\; dx \\
&= AB\int_{-\infty}^{\tau} exp\Big\{-\Big(ax^2 + 2(b_1 - b_2)x + (-2b_1\tau)\Big)\Big\}\; dx \\
&= AB\int_{-\infty}^{\tau} exp\Big\{-\;(ax^2 + 2bx + c)\Big\}\; dx
\end{aligned} \tag{5.2-9}$$

where

$$b = b_1 - b_2 \qquad\qquad c = -2b_1\tau.$$

Now the integral in (5.2-9) can be evaluated using (5.2-5), which yields

$$\begin{aligned}
P_R(\tau) &= \frac{AB}{2}\sqrt{\frac{\pi}{a}}\,exp\Big\{\frac{b^2 - ac}{a}\Big\}\Big[erf\Big\{\sqrt{a}x + \frac{b}{\sqrt{a}}\Big\}\Big]_{-\infty}^{\tau} \\
&= \frac{AB}{2}\sqrt{\frac{\pi}{a}}\,exp\Big\{\frac{b^2 - ac}{a}\Big\}\Big[erf\Big\{\sqrt{a}x + \frac{b}{\sqrt{a}}\Big\} + 1\Big].
\end{aligned} \tag{5.2-10}$$

Great care must be taken when evaluating (5.2-10) due to the multiplication of the rapidly varying exponential and error functions. After manipulating the equation to alleviate this problem (see Appendix B for detailed derivation) and substituting back in for the parameters $a$, $b$, c, $A$, and $B$, (5.2-10) becomes

$$P_R(\tau) = \frac{G_o^2 \lambda^2 c \sigma^o P_T \sigma_2 e^{-8}\sqrt{2}}{8(4\pi)^2 L_p h^3} \left(\frac{1}{x}\right) exp\left\{\frac{\tau}{2\sigma_t^2}(8\sigma_t - \tau)\right\}$$

$$\cdot \left\{1 + \sum_{m=1}^{\infty} \frac{(-1)^m [1 \cdot 3 \cdot \cdot \cdot (2m-1)]}{(2x^2)^m}\right\} \qquad (5.2\text{-}11)$$

where $\tau < 2\sigma_t^2\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right)$ and $x = \sqrt{2}\sigma_t\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right) - \frac{\tau}{\sqrt{2}\sigma_t}$. This result came about by making use of an asymptotic approximation found in [1] for the error function (see Appendix B).

Now (5.2-11) can be compared with the results of the FFT convolution method. Figure 5.8 shows this comparison for the case of $h = 3048$ m, $\sigma^o = 0$ dB, a sampling interval of 0.01 ns and a symmetric beamwidth of 0.6313°. Examining this figure shows that these algorithms do indeed perform as expected.

An important result of all of the above tests is that there is no essential difference between the results of the two numerical convolution algorithms. Since the output is identical for the two methods, the logical conclusion is that the faster of the two routines, the FFT algorithm, should be used exclusively to perform any of the necessary convolutions. Section 5.3 makes use of this result, which speeds up the overall computation time markedly.

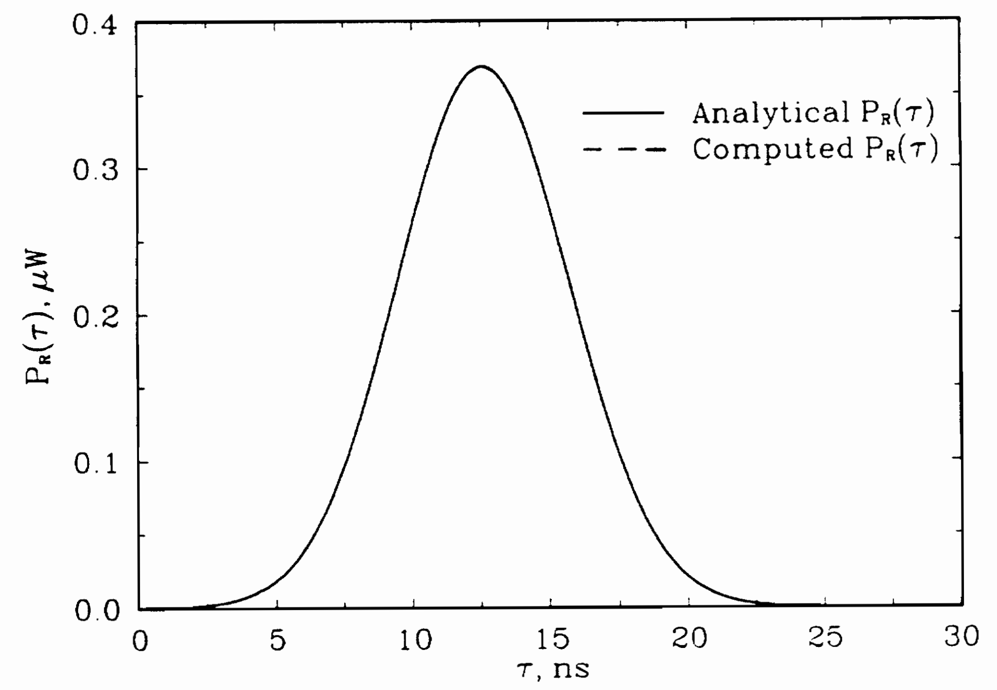


**Figure 5.8.** Average return power for nadir, showing that the analytical and FFT convolution methods are in excellent agreement.

## 5.3 NUMERICAL RESULTS

The algorithm testing performed in the previous sections gave a few examples of the RPM program, so in this section, more examples will be discussed that will complete the set that reflect on the MARA system capabilities.

First, the convolved sea state and radar system point target response, $r(\tau)$, will be investigated, showing its modeled or expected representation. Then the flat surface impulse response, $P_{FS}(\tau)$, waveforms that were modeled in Section 5.1.3 will be convolved with $r(\tau)$ to obtain models of the average return power waveforms, $P_R(\tau)$, for various pointing angles. At this point some signal-to-noise calculations will be performed and compared to results obtained before the MARA system was constructed. Finally, the issue of pointing angle determination from these modeled returns will be investigated, revealing some preliminary findings that are of practical use.

### *5.3.1 MODELED SEA STATE AND POINT TARGET RESPONSE*

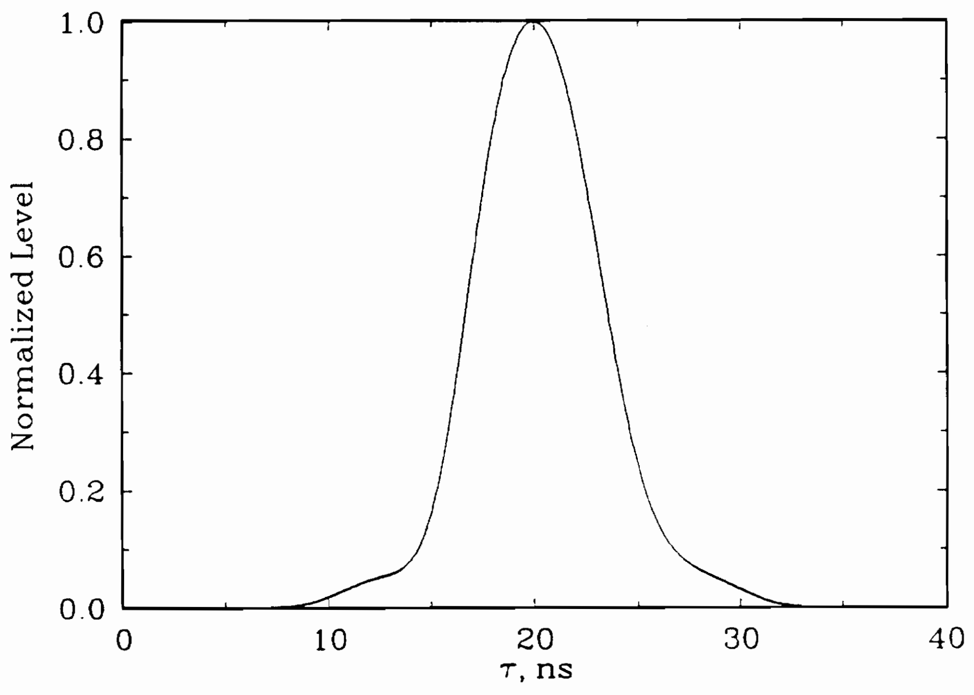
In Chapter 3, the sea state function, more accurately known as the height probability density function of the specular points, was shown to be represented by a Gaussian distribution. The radar system point target response could be represented either as another Gaussian-type function or by the measured Extended Interaction Klystron output pulse given in [26] for the MARA system.

When both functions were Gaussian, their convolution was shown to be rather trivial, independent of the convolution methods discussed in Chapter 4. However, even though this was a reasonable estimate of the combined function, $r(\tau)$, it was still not as exact as could be achieved.

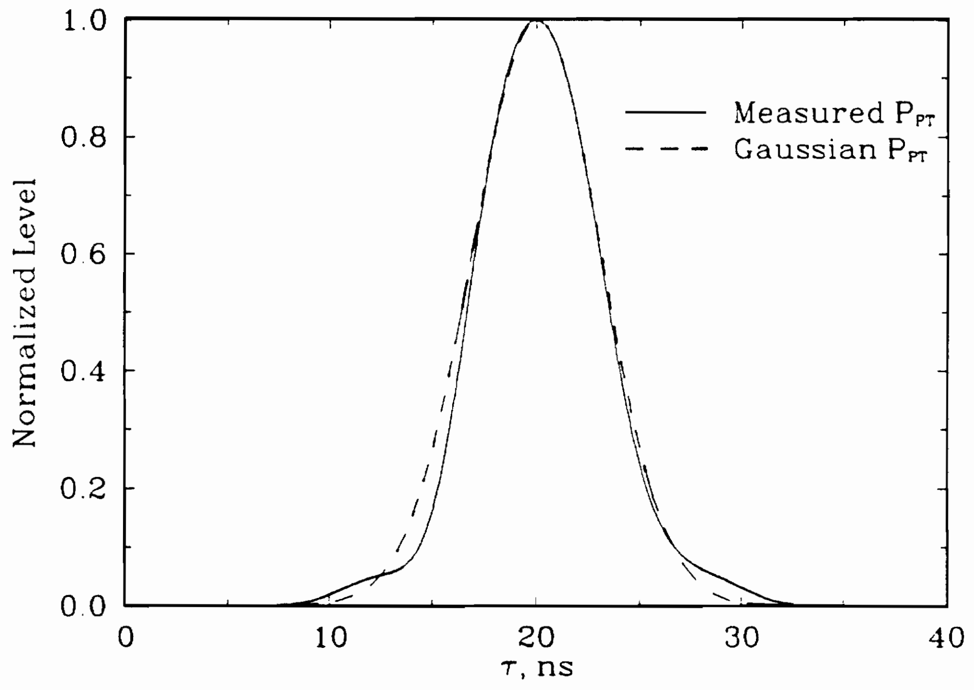
By using the actual measured data for the point target response, the resulting representation for $r(\tau)$ is as close to the true function as can be obtained. When the true point target response function, seen in Figure 3.1, is convolved with the Gaussian sea state function whose rms height, $\sigma_s$, is 0.2 m (see Section 3.3), the resulting $r(\tau)$ is seen in Figure 5.9a. Figure 5.9b shows the corresponding Gaussian representation compared to the true $r(\tau)$, indicating that the Gaussian representation is a reasonable approximation. For the remainder of this chapter, the $r(\tau)$ representation found using the measured point target response will be used to compute $P_R(\tau)$ for the MARA system.

### *5.3.2 MODELED AVERAGE RETURN POWER WAVEFORMS*

The flat surface impulse response, $P_{FS}(\tau)$, for a pointing angle of $\xi = 0^\circ$ has already been computed in Section 5.1.1 for the MARA system and is shown in Figure 5.1. When this is convolved with the $r(\tau)$ representation in Figure 5.9a, the result is the average return power waveform, $P_R(\tau)$, shown in Figure 5.10. Convolving the discontinuous $P_{FS}(\tau)$ with the smoothly varying $r(\tau)$ has the effect of smearing out $P_{FS}(\tau)$, resulting in the delayed and smoothed $P_R(\tau)$ signal. Figure 5.10 represents the average of many returns seen by the MARA system



(a)



(b)

**Figure 5.9.** Convolution of the sea state and the MARA point target responses (a), and Gaussian approximation (b). $\sigma_s = 0.2$ m for both.

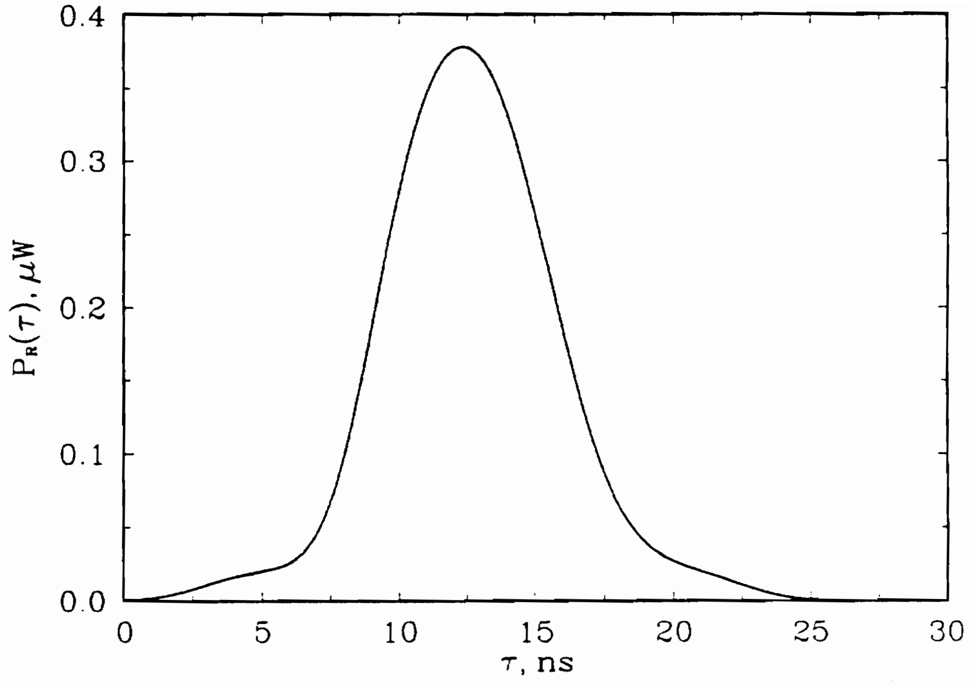


Figure 5.10. Average return power as a function of $\tau$ for the pointing angle of $\xi = 0^{\circ}$, asymmetric beamwidth and $\sigma^{o} = 0$ dB.

when pointed at nadir.

Section 5.1.3 discussed how the flat surface impulse response behaves as the pointing angle $\xi$ increases. These waveforms, a few of which are shown in Figure 5.3, were computed using the changing beamwidth model, discussed in Section 2.1, that is necessary to account for the behavior of the MARA system. These waveforms are now convolved with the $r(\tau)$ discussed in the previous section to obtain the signals shown in Figure 5.11, the $P_R(\tau)$ waveforms for pointing angles of 2, 4, 6, 8, 10 and 12°. This figure shows how the returns broaden as the pointing angle increases, as did the $P_{FS}(\tau)$ signals in Figure 5.3.

Another interesting aspect of the $P_R(\tau)$ computations to examine is the behavior of the peak amplitude of these returns, $\widehat{P}_R$, as the pointing angle varies. Figure 5.12 shows this plot, which decays as expected, since the peak FSIR was also shown to decay as $\xi$ increases in Section 5.1.4. Intuitively, as a radar is pointed further from nadir, the peak return is expected to decrease at a rate dependent on the surface conditions and carrier frequency. Figure 5.12 reflects this roll-off for the MARA system parameters [27].

### *5.3.3 SIGNAL-TO-NOISE CALCULATIONS*

Before the MARA system was built, considerable planning went into how the radar should perform under worst-case conditions [27]. Part of the process involved calculating the signal-to-noise ratio, $S/N$, at nadir for the worst case of

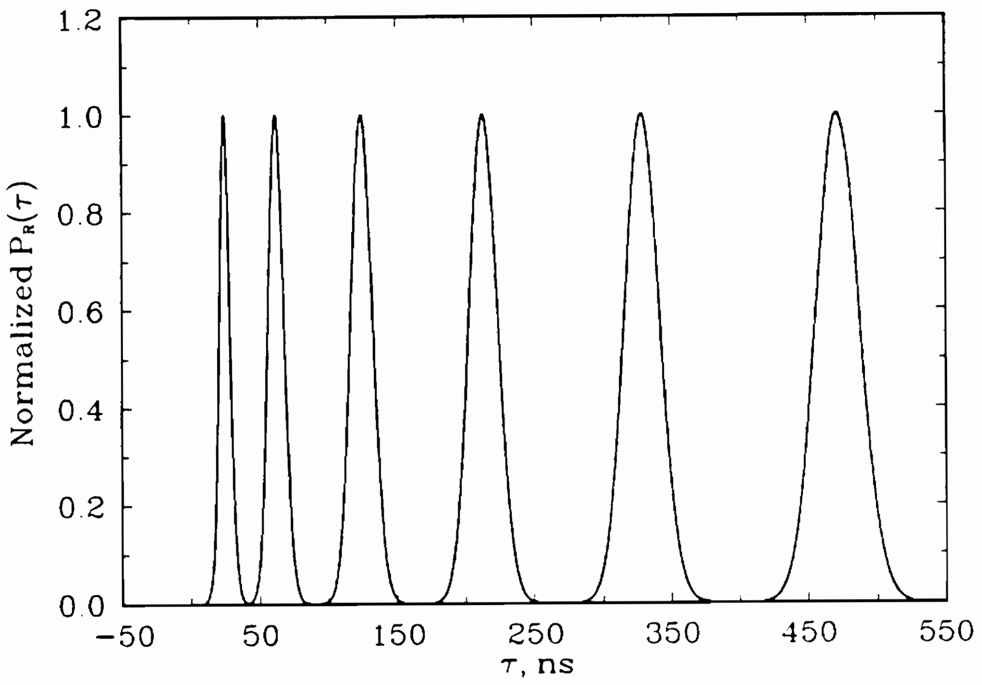


**Figure 5.11.** Average return power as a function of $\tau$ for pointing angles of 2, 4, 6, 8, 10 and 12°, using asymmetric beamwidth modeling and $\sigma^o = 0$ dB.

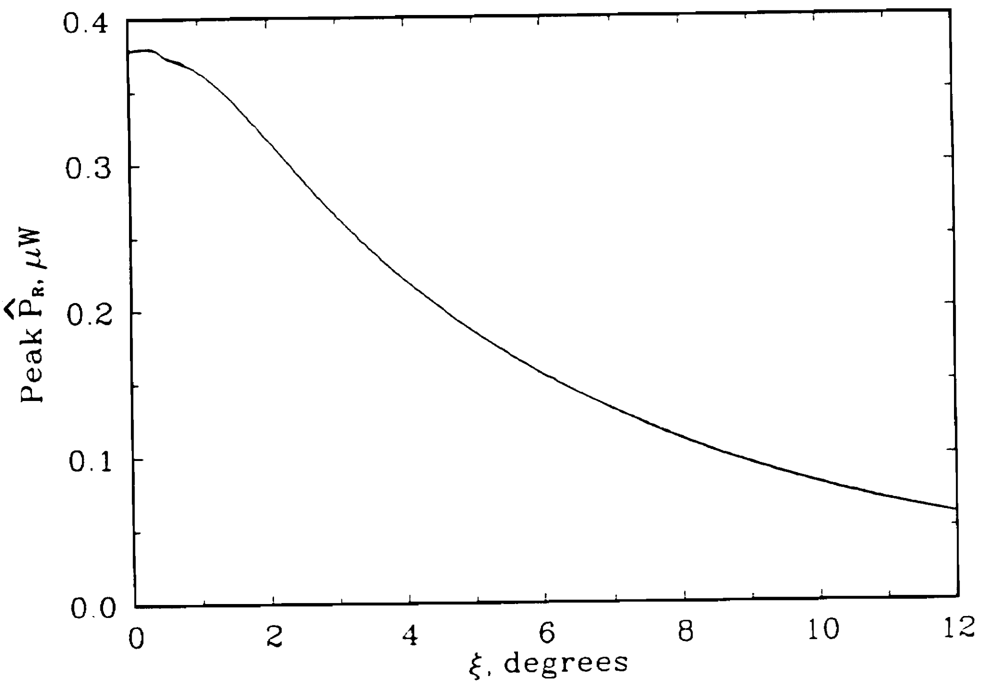


**Figure 5.12.** Peak average return power as a function of pointing angle, using asymmetric beamwidth modeling and $\sigma^o = 0$ dB and $h = 3048$ m.

**Table 5.1.** Preliminary MARA system $S/N$ results as reported in [27]. All values in dB unless otherwise noted.

| | |
|---|---|
| Pointing angle, $\xi$ | 0° |
| Peak power, $\widehat{P}_T$ | 30.0 dB |
| Antenna gain, $G^2$ | 95.0 dB |
| Wavelength, $\lambda^2$ | -41.6 dB |
| Backscatter, $\sigma^o$ | -23.0 |
| Beamwidth, $\theta_s^2$ | -42.7 |
| Constant, $1/256\pi^2$ | -34.0 |
| Range, $1/h^2$ | -69.7 |
| Noise, $1/kT_oB$ | 121.0 |
| Receiver noise figure, $1/F$ | -10.0 |
| Instrument losses, $L$ | -10.0 |
| Atmospheric losses, $L_p$ | 0.0 |
| $S/N$ | 15.0 dB |

$\sigma^o$ = -23 dB, the minimum expected radar cross section to be encountered in MARA studies. Table 5.1 shows the parameters assumed for this computation and the resulting $S/N$ of 15.0 dB. The analysis in [27] used the noise power approximation given by [29]:

$$N = kT_oFB \tag{5.3-1}$$

where $N$ is the noise power at the receiver output in Watts, $k = 1.38 \times 10^{-23}$ J/K, $T_o = 290$K, $F$ is the receiver noise figure and $B$ is the receiver bandwidth in Hz. A more accurate representation of the noise power is given by [28]:

$$N = kT_o(F-1)B. \tag{5.3-2}$$

When the noise power is computed using (5.3-1) in the RPM algorithm, the result is $S/N = 13.8$ dB, which compares favorably with the result in Table 5.1 that was also computed using (5.3-1). Using the more accurate (5.3-2) yields $S/N = 14.3$ dB, only slightly different than the approximate method. Either result, however, indicates that the program is operating as expected.

### *5.3.4 POINTING ANGLE DETERMINATION*

As stated in the introduction, one of the desired goals of this research is to determine a method for using the average return power waveforms to find the pointing angle that the radar used to obtain the signal. Figure 5.11 shows the $P_R(\tau)$ waveforms for various pointing angles, which reveals the dependence of the time-width of the signals on the pointing angle. This feature can be exploited to determine the pointing angle from an arbitrary return signal. Figure 5.13 shows the half-power width of the $P_R(\tau)$ signals as a function of the pointing angle. If the half-power width of the test return can be determined with reasonable accuracy, then it may be possible to infer its pointing angle from this width, based on the curve of Figure 5.13. The problem lies in the accuracy of the half-power width determination. If the samples that represent the test signal are spaced too far apart, the true half-power width will be very difficult to determine. The method used by the RPM program finds the maximum of the signal, then goes through the data points to determine where each half-power point falls between two points. A linear interpolation is then used to estimate the actual location of the half-power point for each side of the return, providing an estimate of the true half-power width.

Although a crude method, this works fairly well for large pointing angles. Note that in Figure 5.13, when the pointing angle is large, the half-power width curve has a large enough slope that there is little problem determining one pointing angle's width from an adjacent width. However for small pointing angles, on the order of a beamwidth or less in the MARA case, the slope of the curve is quite small, making the distinction between adjacent widths difficult for a given pointing angle. For this reason, the half-power width determination can be used as an estimate for the pointing angle, but when the pointing angle is suspected to be small, another more definite method must also be used to find the pointing angle to a higher degree of certainty. A possible method that may

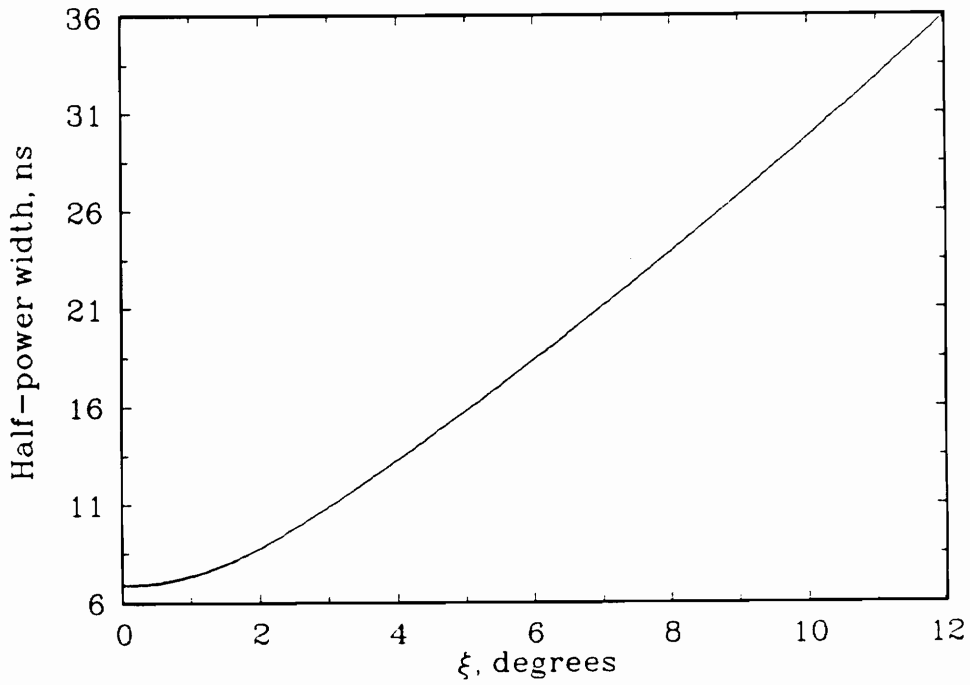


**Figure 5.13.** Half-power width of the average return power waveforms as a function of pointing angle, using asymmetric beamwidth modeling.

perform this task is the use of integration gates [9] applied to the test signal, in a manner that generates one or more parameters that are unique to one particular pointing angle. This concept remains to be investigated further, but these preliminary findings indicate that using the half-power width as a pointing angle estimation technique is a good first guess method.

# 6.0 CONCLUSIONS

An accurate algorithm that makes use of the flat surface impulse response has been developed for the computation of the average return power waveforms seen by the MARA system. The effects of the MARA system lens on the antenna pattern was accounted for by modifying the portion of the flat surface impulse response that describes the antenna pattern and gain. Simplified expressions for the flat surface impulse response were then developed, each for a specific range of pointing angles.

The radar system point target response can be represented by the measured output of the pulse generating klystron on the MARA system, and the height probability density function for the surface specular points (the sea state function) by a Gaussian distribution. Each of these functions were then found to be most accurately and quickly convolved by Fourier transform methods, in particular, by use of the fast Fourier transform (FFT).

Extensive testing was performed to ensure that the algorithms were performing properly. Through these tests and a few more examples, a sampling of the capabilities of these algorithms were seen. As a result of these examples, investigation of the behavior of the average return power waveforms showed that it is possible to estimate the pointing angle from the return based on its half-power width. This, however is only a preliminary finding; it turns out to be more difficult to distinguish between smaller pointing angles since these returns have similar half-power widths. Further investigation is necessary to obtain a more reliable method for pointing angle determination.

# 7.0 REFERENCES

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions*, Dover Publications, Inc., New York, 1972.

[2] D. E. Barrick and C. T. Swift, "The SEASAT microwave instruments in historical perspective," *IEEE J. Oceanic Eng.*, vol. OE-5, pp. 74-79, 1980.

[3] T. Berger, "Satellite altimetry using ocean backscatter," *IEEE Trans. Antennas and Propagation*, vol. AP-20, pp. 295-309, 1972.

[4] C. M. Bender and S. A. Orszag, *Advanced Mathematical Methods for Scientists and Engineers*, McGraw-Hill, Inc., New York, 1978.

[5] W. H. Beyer, ed., *CRC Standard Mathematical Tables, 27th Edition*, CRC Press, Inc., Boca Raton, FL, 1984.

[6] R. N. Bracewell, *The Fourier Transform and Its Applications*, McGraw-Hill, Inc., New York, 1986.

[7] E. O. Brigham, *The Fast Fourier Transform*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[8] G. S. Brown, "Concept for research memorandum: multiple beam radar altimetry for oceanographic and terrain remote sensing," Applied Science

Associates, Apex, NC, pp., 1976.

[9] G. S. Brown, "The average impulse response of a rough surface and its applications," *IEEE Trans. Antennas and Propagation*, vol. AP-25, pp. 67-74, Jan. 1977.

[10] G. S. Brown, "A useful approximation for the flat surface impulse response," *IEEE Trans. Antennas and Propagation*, vol. AP-37, pp. 764-767, June 1989.

[11] G. B. Bush, E. B. Dobson, R. Matyskiela, E. Walsh, and C. C. Kilgus, "An analysis and simulation of the multibeam altimeter," JHU/Applied Physics Laboratory Report SDO 5559, 1980.

[12] G. S. Hayne, "Radar altimeter mean return waveforms from near-normal-incidence ocean surface scattering," *IEEE Trans. Antennas and Propagation*, vol. AP-28, pp.687-692, 1980.

[13] Johns Hopkins University Applied Physics Laboratory, *Johns Hopkins APL Technical Digest*, vol. 8, no. 2, 1987.

[14] *Journal of Geophysical Research*, vol. 84, no. B8, 1979.

[15] H. S. Lee and C. L. Parsons, "Mesoscale ocean eddy measurements by multibeam altimetry," *J. Geophys. Res.*, vol. 91, pp. 9693-9399, 1986.

[16] J. T. McGoogan, L. S. Miller, G. S. Brown, and G. S. Hayne, "The S-193 radar altimeter experiment," *Proc. IEEE*, vol. 62, pp. 793-803, 1974.

[17] J. T. McGoogan, C. D. Leitao, W. T. Wells, L. S. Miller, and G. S. Brown, "SKYLAB altimeter applications and scientific results," *Progress in Astronautics and Aeronautics*, vol. 48, pp. 299-326, 1976.

[18] J. T. McGoogan and E. J. Walsh, "Real-time determination of geophysical parameters from a multibeam radar altimeter," *Progress in Astronautics and*

*Aeronautics*, vol. 67, pp. 110-133, 1978.

[19] L. S. Miller and D. L. Hammond, "Objectives and capabilities of the SKYLAB S-193 altimeter experiment," *IEEE Trans. Geoscience Electronics*, vol. GE-10, pp. 73-79, 1972.

[20] R. K. Moore and C. S. Williams, Jr., "Radar terrain at near vertical incidence," *Proc. IRE*, vol. 45, pp. 228-238, 1957.

[21] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, Inc., New York, 1965.

[22] C. L. Parsons and L. S. Miller, "Design of a large-aperture lens antenna usable over a $\pm 15^\circ$ scanning sector," *IEEE Trans. Antennas and Propagation*, vol. 36, pp. 1162-1165, 1988.

[23] C. L. Parsons and E. J. Walsh, "Off-nadir radar altimetry," *IEEE Trans. Geoscience and Remote Sensing*, vol. 27, pp. 215-224, 1989.

[24] C. L. Parsons, "Airborne multibeam radar altimetry," Private Communication, Sept. 1988.

[25] C. L. Parsons, "MARA antenna system document," Private Communication, Sept. 1989.

[26] C. L. Parsons, "Update on MARA Multibeam Mode status," Private Communication, Sept. 1989.

[27] C. L. Parsons, ed., *MARA System Documentation, Volume I - MARA System Requirements Document*, NASA Reference Publication 1226, July 1989.

[28] T. Pratt and C. W. Bostian, *Satellite Communications*, John Wiley and Sons, New York, 1986.

[29] M. I. Skolnik, *Introduction to Radar Systems, Second Edition*, McGraw-Hill, Inc., New York, 1980.

[30] W. L. Stutzman and G. A. Thiele, *Antenna Theory and Design*, John Wiley and Sons, New York, 1981.

[31] W. F. Townsend, "An initial assessment of the performance achieved by the SEASAT-1 radar altimeter," *IEEE J. Oceanic Eng.*, vol. OE-5, p. 82, 1980.

# A. PROGRAM LISTINGS

The Return Power waveform Master (RPM) program, version 2.0, is a general algorithm capable of performing several tasks. Its primary purpose is to compute the average return power waveform as a function of delay time for a given pointing angle, radar height and sea state. This function is accomplished in the Single Angle Mode, which can also compute these returns for several pointing angles is one program run. Other possible outputs of this mode are the flat surface impulse response and signal-to-noise return, both as a function of delay time, and the half-power width of the average return power waveforms. The other basic option is to operate in Continuous Angle Mode, which can compute such quantities as the peak flat surface impulse response, the peak average return power, peak signal-to-noise ratio, and half-power width of the average return waveforms, all as a function of pointing angle.

Control over the program's input parameters and output information is done by modifying the input file RPMIN.FIL, an example of which is shown for a pointing angle of zero in Section A.1. The user can specify whether to compute the desired output for circularly symmetric or elliptic beamwidth modeling (see Section 2.1), varying radar cross-section per unit scattering area, radar height, frequency, rms waveheight and peak transmit power. The point target response can be modeled as a Gaussian function or specified by a data file (see Section 3.2). Time and pointing angle increments, receiver noise figure and bandwidth, FFT lengths (see Section 4.2) and alternate parameters can each be individually specified. The alternate parameters are specified for the Continuous Mode, so that at a particular pointing angle parameters such as time increment, pointing angle increment and FFT length can be changed to reflect the requirements of the particular pointing angle. In particular, at nadir and small pointing angles, smaller time increments and larger FFT lengths are necessary to obtain accurate results (see Chapter 5).

Each of the individual program listings are given in Section A.2, starting with the main program (RPM) (A.2.1), which controls the input and output of the program. Separate subroutines follow, each responsible for a particular portion of the return power computation. PFSIR (A.2.2) computes the flat surface impulse response using one of the three expressions derived in Chapter 2. DCOMP (A.2.3) computes the function that represents the convolved sea state and radar system point target response. CONVF (A.2.4) performs the convolution of the FSIR with the sea state/point target response using the FFT while CONVI (A.2.5) performs this convolution by numerical integration. FFT (A.2.6) lists the FFT algorithm used to perform the convolutions and INOT (A.2.7) is used to compute the value of the zeroth order Bessel function.

The program automatically determines which output files are necessary and assigns filenames that are descriptive of their contents. In the Single Angle Mode, possible filenames are *PFSxx.###*, *RPWxx.###*, *INFxx.###* and *STNxx.###* for the FSIR, average return power waveform, S/N, and general information files, respectively. The *xx* portion indicates whether equal or unequal beamwidths were used, and if the convolution was performed using the FFT or numerical integration methods. The three digits in the extension .###, represent the pointing angle used in tenths of a degree. As an example, the FSIR file that uses equal beamwidths modeling and numerical integration convolutions for a pointing angle of 4.6° would be contained in *PFSIE*.46.

In the Continuous Angle Mode, the output files can be named *PMxx.DAT*, *RMxx.DAT*, and *HPxx.DAT*, representing the peak FSIR and average return power and half-power width of the average return power, respectively, each as a function of pointing angle. The *xx* portion again indicates the beamwidth modeling and convolution type as for the Single Angle Mode.

Section A.3 lists the ouput file that was obtained from running the RPM program with the input file shown in Section A.1. The output file lists pertinent information for the output waveforms in Single Angle Mode, including maximum amplitudes and time locations fro the FSIR and average return power signals, and the half-power width of the return signal. This file is not generated in the Continuous Angle Mode.

These routines were written in Microsoft® FORTRAN 5.0 on an IBM PS/2 model 50 computer. The main goal was to retain portability and compact, efficient algorithms while maximizing the usefulness. Future work will include reducing code size and removing the convolution by numerical integration subroutine CONVI, in order to further streamline the program.

## A.1 RPM INPUT FILE

```
Input file for RPM.FOR, Version 2.0. Change only information on left.
****  FSIR parameters  ****
U              : QB, Beamwidth type, (E)qual or (U)nequal
BEAM05.FIL     : BWFILE, cross-scan BW data file          (check XINC)
0.6313         : HPSC, scan beamwidth, degrees, case (U) only
0.6313         : HPB, symmetric beamwidth, degrees, case (E) only
-5.0           : SIGODB, backscattering cross section, dB
10.0           : LOSSDB, system losses, including atmospheric, dB
3048.0         : H, radar height above mean sea level, m
36.0           : FRQ, radar frequency, GHz
****  Sea state / point target response parameters  ****
F              : QR, point target response type, (F)ile or (G)aussian
PT002.FIL      : PTFILE, point target response data fn    (check TINC)
0.20           : SIGS, rms sea height, m
6.55           : WDTH, point target width, ns
1000.0         : PTX, transmit Power, watts
****  Increments and algorithm type  ****
0.02           : TINC, time increment, ns                (check PTFILE)
0.05           : XINC, pointing angle increment, degrees (case C)
S              : QT, algorithm type (C)ontinuous or (S)ingle angle(s)
****  Convolution paramters  ****
F              : QC, convolution type (F)FT or (I)ntegration
12             : MR, FFT length parameter (LFFT=2**MR)    (check TINC)
****  Receiver Specifications  ****
5.5            : NF, noise Figure, dB
Y              : QM, match bandwidth to PTR width? (Y or N)
152.67         : BAND, bandwidth, MHz, if QM = N
****  Alternate parameters (case C)  ****
Y              : QS, switch to alternate parameters? (Y)es or (N)o
1.0            : CX, angle where QS switch will occur
0.1            : TINC2, time increment, ns              (check PTFILE2)
0.1            : XINC2, pointing angle increment, degrees
11             : MR2, FFT length parameter               (check TINC2)
PT01.FIL       : PTFILE2, point target response data fn  (check TINC2)
****  Output parameters  ****
Y              : QW, output PFS to file, (Y)es or (N)o
N              : QP, PFS (N)ormalized or (U)nnormalized
A              : QF, output files in (V)idgraf or (A)xum format
N              : QD, dBm output, (Y)es or (N)o
Y              : QN, output S/N plots? (Y)es or (N)o  (in dB)
****  Pointing angle list  ****
0.0            : XSTRT & XEND, range (C), or X, list of angle (S)
```

## A.2 RPM 2.0 PROGRAM AND RELATED SUBROUTINES

### *A.2.1 RPM 2.0 MAIN PROGRAM*

```
      program rpm
c  Return Power waveform Master program.  Version 2.0.
c  Revision 6/13/90.  M. H. Newkirk.
c  Uses the convolutional model to compute either the return power
c  waveform for a given look angle or set of look angles, or the
c  maximum return power vs. look angle. Convolutions can be performed
c  via FFT's or by numerical integration. The file RPMIN.DAT must
c  be modified to obtain the desired results.
c
c  Subroutines called:
c
c    PFSIR : Computes the Flat Surface Impule Response function.
c    DCOMP : Computes the distribution that approximates the
c          : sea state and point target responses.
c    FFT   : Computes the Fast Fourier Transform of a data set.
c    CONVF : Computes the FFT convolution.
c    CONVI : Computes the integral convolution.
c
      real*8 cl,h,pi,dtr,xi,hp,gam,pmax,bw,beta,c,s,tmin,tinc
      real*8 amax,pfs(5500),dist(5001),bmw(250),hpb,gam2,tinc2
      real*8 dmax,loss,gain,lamb,pfc,sigs,wdth,ptx,hpsc,nf,band
      real*8 sigodb,sigo,frq,lossdb
      complex*16 resp[huge](4096)
      dimension conv(9000)
      common /dcom/ dist
      common /gcom/ tmin,beta,gam,gam2,pfc,h
      common /pcom/ pfs
      common /rcom/ resp
      common /fcom/ istrt,idif,conv
      character*12 bwfile,rpwfil,pfsfil,infofl,ptfil,ptfil2,stnfil
      character qb,qc,qt,qp,qr,qw,qf,qs,qd,qn,qm
      character*3 nmc
c
c  Input section
c
      write(*,29)
   29 format(/' RPM: Return Power waveform Master program,'
     1' Version 2.0, by M Newkirk'/)
      open(1,status='scratch')          ! Open scratch file
c
      open(2,file = 'rpmin.fil')        ! Open input data file
      read(2,*)
      read(2,*)
      read(2,20) qb                       ! Beamwidth model type:
      if(qb.eq.'e') qb = 'E'              ! Equal BW
      if(qb.eq.'u') qb = 'U'              ! Unequal BW
   20 format(a1)
      if(qb.eq.'U') then                  ! If unequal model
         read(2,21) bwfile                ! Filename of unequal BW data
   21    format(a12)
         open(3,file=bwfile,status='old')
         read(3,30)
   30    format(//)
         do i = 1,500
            read(3,*,end=7) dummy,bmw(i)
         end do
    7    close(3)
         read(2,*) hpsc                   ! Scan BW value
         read(2,*)
      else                                ! Otherwise equal BW model
         read(2,*)
         read(2,*)
         read(2,*) hpb                    ! Equal BW value
      endif
      read(2,*) sigodb                    ! σ° in dB
      read(2,*) lossdb                    ! All losses
      read(2,*) h                         ! Radar height
      read(2,*) frq                       ! Frequency of radar
      read(2,*)
c
      read(2,20) qr                       ! Point target response type:
      if(qr.eq.'f') qr = 'F'              ! From data file, or
      if(qr.eq.'g') qr = 'G'              ! Use gaussian approximation.
      read(2,21) ptfil                    ! Filename for data file
      read(2,*) sigs                      ! Sea height parameter, cm
      read(2,*) wdth                      ! PTR gaussian 3 dB width, ns
      read(2,*) ptx                       ! Peak transmit power, W
      read(2,*)
c
      read(2,*) tinc                      ! Desired time increment
      read(2,*) xinc                      ! Angle increment for qt = C
      read(2,20) qt                       ! RPM algorithm type:
      if(qt.eq.'c') qt = 'C'              ! Continuous (max vs. angle)
      if(qt.eq.'s') qt = 'S'              ! Single angle(s) (waveforms)
      read(2,*)
c
      read(2,20) qc                       ! Convolution type:
      if(qc.eq.'f') qc = 'F'              ! FFT method, or
      if(qc.eq.'i') qc = 'I'              ! Integration method.
      read(2,*) mr                        ! FFT length parameter
      lfft = 2 ** mr
      read(2,*)
c
      lcon = idnint(30.d0 / tinc)         ! Set integration length
      if(qc.eq.'I') write(*,31) lcon
   31 format(' Integration length is:',i5,' to accomondate'
     1' the given time increment'/)
c
      read(2,*) nf                         ! Receiver noise figure
      read(2,20) qm                        ! Match BAND to PTR width?
      if(qm.eq.'y') qm = 'Y'               ! Yes, or
      if(qm.eq.'n') qm = 'N'               ! No
      if(qm.eq.'N') then
         read(2,*) band                    ! Receiver bandwidth, if QM = N
         band = band * 1.d-3               ! Convert to GHz
      else
         read(2,*)
      endif
      read(2,*)
c
      read(2,20) qs                        ! Change to alternate params?
      if(qs.eq.'y') qs = 'Y'               ! Yes, or
      if(qs.eq.'n') qs = 'N'               ! No.
      read(2,*) cx                         ! Angle where change is made
      read(2,*) tinc2                      ! Alternate time increment
      read(2,*) xinc2                      ! Alternate angle increment
      read(2,*) mr2                        ! Alternate FFT parameter
      read(2,21) ptfil2                    ! Alternate pt targ resp fn
      read(2,*)
c
      read(2,20) qw                        ! Output FSIR? (qt = S only):
      if(qw.eq.'y') qw = 'Y'               ! Yes, or
      if(qw.eq.'n') qw = 'N'               ! No
      read(2,20) qp                        ! FSIR format:
      if(qp.eq.'n') qp = 'N'               ! Normalized to unity, or
      if(qp.eq.'u') qp = 'U'               ! Unnormalized
      read(2,20) qf                        ! Output files format:
      if(qf.eq.'v') qf = 'V'               ! Vidgraf, or
      if(qf.eq.'a') qf = 'A'               ! AXUM
      read(2,20) qd                        ! Output power files in:
      if(qd.eq.'y') qd = 'Y'               ! dBm, or
      if(qd.eq.'n') qd = 'N'               ! Normal
      read(2,20) qn                        ! Output S/N plots?
      if(qn.eq.'y') qn = 'Y'               ! Yes, or
      if(qn.eq.'n') qn = 'N'               ! No
      read(2,*)
c
c  Set constants.
c
      cl = 0.3d0                           ! Speed of light
      lamb = cl / frq                      ! Wavelength
      sigo = 10.d0 ** (sigodb / 10.d0)     ! Convert σ° from dB to ratio
      loss = 10.d0 ** (lossdb / 10.d0)     ! Convert loss from dB to ratio
      pi = 2.d0 * dacos(0.d0)              ! π
      dtr = pi / 180.d0                    ! Degrees-to-radians conversion
```

```
c
c  Set HP based on beamwidth type and calculate gamma.
c
      if(qb.eq.'U') then
         hp = hpsc/2 * dtr
      else
         hp = (hpb / 2.d0) * dtr
      endif
      gam = -2.0d0 * (dsin(hp)**2) / dlog(0.5d0)
c
c  Compute the starting time for evaluation ( t > 2h/c ), based on
c  radar antenna height.
c
      tmin = 2.d0 * (h / cl)
c
      if(qt.eq.'C') then
         read(2,*) x,xend               ! Range of angles to evaluate
         if(qb.eq.'U'.and.qc.eq.'F') then
            open(4,file = 'pm.dat')
            open(5,file = 'rm.dat')
            open(6,file = 'hp.dat')
            if(qn.eq.'Y') open(7,file = 'sn.dat')
         elseif(qb.eq.'E'.and.qc.eq.'F') then
            open(4,file = 'pme.dat')
            open(5,file = 'rme.dat')
            open(6,file = 'hpe.dat')
            if(qn.eq.'Y') open(7,file = 'sne.dat')
         elseif(qb.eq.'E'.and.qc.eq.'I') then
            open(4,file = 'pmie.dat')
            open(5,file = 'rmie.dat')
            open(6,file = 'hpie.dat')
            if(qn.eq.'Y') open(7,file = 'snie.dat')
         else
            open(4,file = 'pmi.dat')
            open(5,file = 'rmi.dat')
            open(6,file = 'hpi.dat')
            if(qn.eq.'Y') open(7,file = 'sni.dat')
         endif
         if(qf.eq.'V') then
            if(qs.eq.'Y') then
               nl = nint( (cx - x) / xinc + (xend - cx) / xinc2 ) + 1
            else
               nl = nint( (xend - x) / xinc ) + 1
            endif
            write(4,*) nl
            write(5,*) nl
            write(6,*) nl
            if(qn.eq.'Y') write(7,*) nl
         endif
      else                              ! Single angle case:
         read(2,*) x                    ! first angle to evaluate
      endif
c
    6 if(qc.eq.'F') then      ! Set up DIST for FFT convolution
         call dcomp(sigs,qr,ptfil,wdth,ptx,tinc,dmax,lfft/2,mr,ishft)
         do i = 1,lfft        ! Put DIST in wrap around array RESP
            if(i.lt.lfft/4) then
               resp(i) = dcmplx(dist(lfft/4+i),0.d0)
            elseif(i.gt.3*lfft/4) then
               resp(i) = dcmplx(dist(i-3*lfft/4),0.d0)
            else
               resp(i) = dcmplx(0.d0,0.d0)
            endif
         end do
         call fft(resp(1),mr)      ! Transform RESP
      else                    ! Set up DIST for integral convolution
         call dcomp(sigs,qr,ptfil,wdth,ptx,tinc,dmax,lcon,mr,ishft)
      endif
c
c  Compute Noise Power
c
      tn = 290.d0 * (10.d0**(nf / 10.d0) - 1.d0)   ! Noise temp, K
      if(qm.eq.'Y') band = 1.d0 / wdth             ! In GHz
      pn = 10.d0 * dlog10(1.38d-14 * tn * band)    ! Noise Power, dBW
      if(qn.eq.'Y') write(*,40) band*1.d3, pn
   40 format(' Bandwidth: ',f7.2,' MHz; Noise power: ',f7.2,' dBW'/)
c
c  Open files depending on algorithm type.
c
    1 if(qt.eq.'S') then
         nm = nint(x*10.0)
         rewind 1
         if(nm.lt.10) then
            write(1,41) nm
            rewind 1
            read(1,44) nmc
         elseif(nm.lt.100.and.nm.ge.10) then
            write(1,42) nm
            rewind 1
            read(1,44) nmc
         else
            write(1,43) nm
            rewind 1
            read(1,44) nmc
         endif
   41 format(i1)
   42 format(i2)
   43 format(i3)
   44 format(a3)
         if(qb.eq.'U'.and.qc.eq.'F') then
            rpwfil = 'rpwf.'//nmc
            pfsfil = 'pfsf.'//nmc
            infofl = 'inff.'//nmc
            stnfil = 'stnf.'//nmc
         elseif(qb.eq.'U'.and.qc.eq.'I') then
            rpwfil = 'rpwi.'//nmc
            pfsfil = 'pfsi.'//nmc
            infofl = 'infi.'//nmc
            stnfil = 'stni.'//nmc
         elseif(qb.eq.'E'.and.qc.eq.'F') then
            rpwfil = 'rpwfe.'//nmc
            pfsfil = 'pfsfe.'//nmc
            infofl = 'inffe.'//nmc
            stnfil = 'stnfe.'//nmc
         else
            rpwfil = 'rpwie.'//nmc
            pfsfil = 'pfsie.'//nmc
            infofl = 'infie.'//nmc
            stnfil = 'stnie.'//nmc
         endif
         open(4,file=infofl)
         open(5,file=rpwfil)
         if(qw.eq.'Y') open(6,file=pfsfil)
         if(qn.eq.'Y') open(7,file=stnfil)
         write(*,*)
         if(qw.eq.'Y') write(*,55) pfsfil
         if(qn.eq.'Y') write(*,45) stnfil
         write(*,56) rpwfil
         write(*,57) infofl
   55    format(' FSIR will be in a file named:    ',a9)
   56    format(' RPW will be in a file named:     ',a9)
   45    format(' S/N ratio in a file named:       ',a9)
   57    format(' Information about these is in:   ',a9/)
      endif
c
c  Using the look angle, compute a rough estimate of the time location
c  of the maximum return.  Based on this, determine the evaluation
c  interval.
c
    3 write(*,58) x,tinc
   58 format(/' Look Angle is: ',f5.2,' degrees;  Time increment: '
     1,f7.5,' ns'/)
      xi = dble(x) * dtr
      c = dcos(xi)
      s = dsin(xi)
      tloc = tmin * (1.d0 / c - 1.d0)
      tl = aint(tloc + 0.5)
      if(x.ge.10.0) then
         tdif = 60.0
         tend = tl+tdif
      elseif(x.lt.10.0.and.x.gt.2.0) then
         tdif = 50.0
         tend = tl+tdif
      elseif(x.le.2.0.and.x.gt.1.0) then
         tdif = 40.0
         tend = tl+tdif
      else
         tdif = 12.0
         tend = tdif
      endif
      if((tl-tdif).lt.0.0.or.x.le.1.0) then
         tstrt = 0.0
      else
         tstrt = tl-tdif
      endif
c
c  Retrieve the beamwidth value that corresponds to the chosen look
c  angle.  Then compute values for GAIN and BETA.
c
      if(qb.eq.'E') then
         beta = 0.d0
         gain = 15866.1508d0 / hpb**2
         gam2 = 0.d0
      else
         ind = nint(20.0 * x) + 1
         bw = bmw(ind) * dtr
         gam2 = -2.d0 * dsin(bw/2.d0)**2 / dlog(0.5d0)
         beta = dlog(0.5d0) * gam / (-2.d0 * dsin(bw/2.d0)**2) - 1.d0
         gain = 15833.4786d0 * (1.d0 + dble(x/12.0) * 0.02910595d0)
     1         / (hpsc * dble(bmw(ind)))
      endif
      gdb = 10.0 * dlog10(gain)
      write(*,67) beta
      if(qb.eq.'U') write(*,70) gdb
c
c  Compute the leading coefficient for the FSIR.
c
      pfc = (gain * lamb)**2 * cl * sigo
     1    / ((2.d0 * pi) * 4.d0 * (4.d0 * pi)**2 * loss * h**3)
      if(qt.eq.'S') then
c
c  Write pertinent information to INFO file (if QT = S).
c
         write(4,63) pfsfil,rpwfil
         write(4,64) x,h,tinc
         if(qn.eq.'Y') write(4,40) band*1.d3, pn
         if(qb.eq.'U') then
            write(4,65)
         else
            write(4,66) hpb
         endif
         write(4,67) beta
         if(qc.eq.'F') then
            write(4,68) lfft
         else
            write(4,69) lcon
         endif
         write(4,70) gdb
         write(4,75) sigodb,lossdb
         write(4,71) tstrt,tend
         write(4,62) pfc
   62 format(' The leading coefficient for the FSIR = ',d15.8/)
   63 format(' Information for files: ',a8,' and ',a8//)
   64 format(' Look angle: ',f5.2,' degrees; Height: ',f8.2,' m;'
     1' Time incr: ',f6.4,' ns'/)
   65 format(' Beamwidth modelled as unequal'/)
   66 format(' Beamwidth modelled as equal, BW = ',f6.4,' degrees'/)
   67 format(' Beta value for this look angle:',d22.15/)
   68 format(' Convolution is by FFT, FFT length: ',i4/)
   69 format(' Convolution is by Integration, length: ',i4/)
   70 format(' Gain at this look angle is: ',f5.2,' dB'/)
   75 format(' Sigma0: ',f6.2,' dB; Losses: ',f6.2,' dB'/)
   71 format(' Range of FSIR evaluation time: [',f8.4,',',f8.4,
     1'] ns'/)
      endif
c
c  Compute integer beginning and end for evaluations
c
      istrt = idnint(tstrt / tinc) + 1
      iend = idnint(tend / tinc) + 1
      idif = iend - istrt + 1
c
c  Call routine that calculates the Flat Surface Impulse Response.
c
      call pfsir(xi,tinc,istrt,iend,pmax,rmark,mark)
      write(*,72) pmax * 1.d9, rmark
      if(qt.eq.'S') then
         write(4,72) pmax * 1.d9, rmark
      else
         if(qd.eq.'Y') then
            write(4,73) x, 10.0 * dlog10(pmax)
         else
            write(4,73) x, pmax * 1.d9
         endif
      endif
   72 format(' Maximum FSIR value is: ',d13.8,' nW at tau = ',f10.6,
     1' ns'/)
   73 format(1x,f5.2,2x,f15.8)
c
c  Normalize (if desired) PFS and output to data file (if QT = S).
c
      if(qt.eq.'S'.and.qw.eq.'Y'.and.qf.eq.'V') write(6,*) idif
      do i = istrt,iend
         iloc = i - istrt + 1
         outi = float(i-1) * sngl(tinc)
         if(qt.eq.'S'.and.qw.eq.'Y'.and.qd.eq.'Y') then
            if(pfs(iloc).lt.1.d-11) then
               write(6,74) outi, -80.0
            else
               write(6,74) outi, 10.0 * dlog10(pfs(iloc))
            endif
         elseif(qt.eq.'S'.and.qp.eq.'U'.and.qw.eq.'Y') then
            write(6,74) outi, pfs(iloc) * 1.d9
         endif
         pfs(iloc) = pfs(iloc)/pmax
         if(qt.eq.'S'.and.qp.eq.'N'.and.qw.eq.'Y'.and.qd.eq.'N')
     1      write(6,74) outi, pfs(iloc)
      end do
   74 format(1x,f10.6,2x,f15.8)
      if(qt.eq.'S'.and.qw.eq.'Y') close (6)
c
c  Convolution of FSIR and DIST
c
      if(qc.eq.'F') then                  ! Do FFT convolution
         pfs(1) = pfs(1) / 2.d0
         call convf(mr,mark,tinc,ishft,pmax,dmax,amax,amark,k)
      else                                ! Do integration convolution
         call convi(lcon,tinc,ishft,pmax,dmax,amax,amark,k)
      endif
      write(*,75) amax * 1.d6,amark
   75 format(' Maximum RPW value is:  ',d13.8,' μW at  tau = ',10.6,
     1' ns'/)
      write(*,78) 10.0 * dlog10(amax) - pn
   78 format(' Maximum S/N is: ',f7.2,' dB'/)
c
c  Compute the 3 dB width by interpolation
c
      km = 0
      do i = istrt,iend+k
         iloc = i - istrt + 1
         if(km.eq.0.and.conv(iloc).gt.0.5d0) then
            z1 = conv(iloc-1)
            dz = conv(iloc) - z1
            tm1 = float(iloc-1) * tinc + ((0.5 - z1) * tinc) / dz
            km = 1
         elseif(km.eq.1.and.conv(iloc).lt.0.5d0) then
            z1 = conv(iloc-1)
            dz = conv(iloc) - z1
            tm2 = float(iloc-1) * tinc + ((0.5 - z1) * tinc) / dz
            exit
         endif
      end do
      thp = tm2 - tm1
      write(*,46) thp
   46 format(' 3 dB width of RPW is: ',f10.6,' ns'/)
c
```

```
c  Output section
c
      if(qt.eq.'C') then
         if(qd.eq.'N') then
            write(5,73) x,amax * 1.d6
         else
            write(5,73) x, 10.0 * dlog10( amax )
         endif
         if(qn.eq.'Y') write(7,73) x, 10.0 * dlog10(amax) - pn
         write(6,76) x,thp
   76    format(1x,f5.2,2x,f10.6)
c
         x = x + xinc                    ! Increment pointing angle
         if(qs.eq.'Y'.and.x.ge.cx-0.01.and.x.le.cx+0.01) then
            tinc = tinc2                 ! time increment,
            xinc = xinc2                 ! angle increment,
            ptfil = ptfil2               ! pt. targ. file, and
            mr = mr2                     ! FFT length parameter
            lfft = 2 ** mr
            do i = 1,idif                ! Reset array
               pfs(i) = 0.d0
            end do
            write(*,77) x
            goto 6
         endif
   77    format(' *** Switching to alternate parameters at',f6.2,
     1          ' degrees ***'/)
         if(nint(10.0*x).le.nint(10.0*xend)) then
            do i = 1,idif                ! Reset array
               pfs(i) = 0.d0
            end do
            goto 3
         endif
         close (4)
         close (5)
         close (6)
         close (7)
      else                               ! Case qt = S
         write(4,75) amax * 1.d6, amark
         write(4,78) 10.0 * dlog10(amax) - pn
         write(4,46) thp
         if(qf.eq.'V') write(5,*) idif+k
         if(qf.eq.'V'.and.qn.eq.'Y') write(7,*) idif+k
         do i = istrt,iend+k           ! Output RPW to file
            iloc = i - istrt + 1
            outi = float(i-1) * sngl(tinc)
            if(qd.eq.'N') then
               write(5,74) outi,conv(iloc)
            else
               if(amax*conv(iloc).lt.1.e-11) then
                  write(5,74) outi, -80.0
               else
                  write(5,74) outi, 10.0*log10(amax*conv(iloc)*1.e3)
               endif
            endif
            if(qn.eq.'Y')
     1      write(7,74) outi, 10.0*log10(amax*conv(iloc)*1.e3) - pn
         end do
         close (4)
         close (5)
         close (7)
c
         read(2,*,end=5) x                  ! Get next angle
         do i = 1,idif                      ! Reset arrays
            conv(i) = 0.0
            pfs(i) = 0.d0
         end do
         goto 1
    5    close(1,status='delete')
      endif
c
      stop
      end
```

### A.2.2 FSIR SUBROUTINE

```
      subroutine pfsir(xi,tinc,istrt,iend,pmax,rmark,mark)
c  3/05/90  M. Newkirk
c  Subroutine to compute the Flat Surface Impulse Response. Called
c  from RPM.
c  Subroutine called:
c    RINOT : Computes the value of Io(x) when look angle is zero.
c
      double precision pfs(1),xi,tinc,tmin,beta,gam,pfc,h,epr,val
      double precision pmax,arg,pi,c,s,t,taumin,d,eps,r,sum,inot
      double precision p,q1,q2,q3,f,t1,t2,t3,cl,rnot,gam2,taum1,taum2
      common /gcom/ tmin,beta,gam,gam2,pfc,h
      common /pcom/ pfs
c
c  Compute and set contants
c
      c = dcos(xi)
      s = dsin(xi)
      rnot = h * dtan(xi)
      cl = 0.3d0
      pi = 2.d0 * dacos(0.d0)
      pmax = 0.d0
c
c  If look angle is nadir, use closed form expression to evaluate
c  FSIR, then go to the FSIR output section.
c
      if(xi.eq.0.d0) then
         do i = 1,iend
            iloc = i - istrt + 1
            t = dble(i-1) * tinc
            eps = dsqrt(cl * t / h)
            epr = eps**2
            arg = 2.d0 * dabs(beta) * epr / gam
            val = inot(arg)
            if(beta.eq.0.d0) then
               pfs(iloc) = pfc * (2.d0 * pi) * dexp(-4.d0 * epr / gam)
            else
               pfs(iloc) = pfc * (2.d0 * pi) * dexp((-4.d0 * epr /gam)
     1                  * (1.d0 + beta/2.d0 - dabs(beta)/2.d0)) * val
            endif
            if(pfs(iloc).gt.pmax) then
               pmax = pfs(iloc)
               mark = iloc
            endif
         end do
         goto 1
      endif
c
c  When the look angle is not nadir, compute the min tau that can be
```

```
c  evaluated using the asymptotic method, then determine where to go
c  to begin evaluation.
c
      tauml = (6.79d0 * gam / (8.d0 * s * c))**2 * h / cl
      taum2 = (6.79d0 * gam2 / (8.d0 * s * c))**2 * h / cl
      taumin = dmax1(tauml,taum2)
      it = idnint(taumin / tinc)
      if(istrt-1.ge.it) then
         write(*,21)
   21    format(' TSTRT > TAUMIN - asymptotic method used'/)
         iflag = istrt
         goto 2
      endif
      if(it.ge.iend) it = iend-1
c
c       *****   Numerical integration (Simpson's rule) section    *****
c
      ndiv = 200
      d = pi/dble(ndiv/2)
      do i = istrt,it+1
         iloc = i - istrt + 1
         t = dble(i-1) * tinc
         eps = dsqrt(cl * t / h)
         r = eps * h
         sum = 0.d0
c
c  Simpson's rule
c
         do j = 0,ndiv                     ! Function evaluation
            p = dble(j) * d - pi
            q1 = rnot**2 + r**2 - 2.d0 * rnot * r * dcos(p)
            if(q1.lt.1.d-7) q1 = 1.d-7
            q2 = 1.d0 + beta * (r * dsin(p))**2 / q1
            q3 = 1.d0 - (c + eps * s * dcos(p))**2 / (1.d0 + eps**2)
            f = dexp((-4.d0 / gam) * q2 * q3)
            if(j.eq.0.or.j.eq.ndiv) then    ! Summation section
               sum = sum + f
               ll = 1
               cycle
            endif
            if(ll.eq.1) then
               sum = sum + 4.d0 * f
               ll = 0
            else
               sum = sum + 2.d0 * f
               ll = 1
            endif
         end do
         pfs(iloc) = pfc * sum * (d/3.d0)
         if(pfs(iloc).gt.pmax) then
            pmax = pfs(iloc)
            mark = iloc
         endif
      end do
      iflag = it+2
      if(iflag.gt.iend) goto 1
      write(*,22) float(iflag-1) * sngl(tinc)
   22 format(' *** Switching to asymptotic evaluation at t = ',f8.3,
     1' ns ***'/)
c
c             *****   Asymptotic evaluation section   *****
c
    2 do i = iflag,iend
         iloc = i - istrt + 1
         t = dble(i-1) * tinc
         if(i.eq.1) then
            eps = 1.d-8
         else
            eps = dsqrt(cl * t / h)
         endif
         t1 = dexp((-4.d0 / gam) * (s - eps * c)**2 / (1.d0 + eps**2))
         t2 = c * s + eps * (s**2 + beta * c**2)
         if(t2.lt.0.d0) t2 = 1.d-8
         t3 = dsqrt(pi * gam * (1.d0 + eps**2) / (4.d0 * eps * t2))
         pfs(iloc) = pfc * t1 * t3
         if(pfs(iloc).gt.pmax) then
            pmax = pfs(iloc)
            mark = iloc
         endif
      end do
    1 rmark = float(istrt + mark - 2) * sngl(tinc)
      return
      end
```

## A.2.3 SEA STATE/POINT TARGET RESPONSE SUBROUTINE

```
      subroutine dcomp(sigs,qr,ptfile,wdth,ptx,tinc,dmax,ns,m,k)
c  Revised: 4/04/90. M H Newkirk.
c  This subroutine performs the convolution of the sea state and radar
c  system point target reponses, with the result returned in the array
c  Z.  The point target response can be either read from a file or
c  modeled by a gaussian, while the sea state is always modeled by a
c  gaussian. SIGS is the rms height of the sea waves, PTX is the peak
c  transmitted power, and WDTH is the 3 dB width of the (gaussian)
c  point target response, all of which are specified in the calling
c  program. QR defines which point target response representation to
c  use and PTFILE is the filename that contains the point target
c  response data. DMAX returns the maximum amplitude of the combined
c  response, since the data in Z is normalized to unity.  K returns
c  the shift value, an integer, which is equivalent to a 4σ offset
c  in time.
c
      real*8 offset,stdev,t,sig1,z(1),sigs,wdth,ptx,zmax
      real*8 c,tinc,var1,var2,dmax,shift,ptmax,pt[allocatable](:)
      complex*16 x[huge,allocatable](:), y[huge,allocatable](:)
      character qr,ptfile*12
      common /dcom/ z
      open(9,file='dc.dat')
      write(9,*) ns
      allocate( pt(ns), stat = ier )
c
c  Convert rms height of sea waves into an "equivalent" width.
c
      c = 0.3d0
      sig1 = 2.d0 * sigs / c
      var1 = sig1 * sig1
      twopi = 4.d0 * dacos(0.d0)
      n = 2**(m-1)
      allocate( x(n), y(n), stat = ier )
c
c  Method using data file for point target response.
c
      if(qr.eq.'F') then
         open(3,file=ptfile)          ! read data from file
         ptmax = 0.d0
         do i = 1,2100
            read(3,*,end=5) dummy,pt(i)
            if(pt(i).gt.ptmax) ptmax = pt(i)     ! find maximum
         end do
    5    close(3)
         lptr = i - 1                 ! length of data in file
         lp2 = nint(float(i)/2.0)
         lq = 0
         do i = 1,lptr                ! find half power point
         if(pt(i).gt.ptmax/2.d0.and.lq.eq.0) then
            a1 = pt(i-1)
            da = pt(i) - a1
            tm1 = float(i-1) * tinc + ((0.5 - a1) * tinc) / da
            lq = 1
         elseif(pt(i).lt.ptmax/2.d0.and.lq.eq.1) then
            a1 = pt(i-1)
            da = pt(i) - a1
            tm2 = float(i-1) * tinc + ((0.5 - a1) * tinc) / da
            exit
         endif
      end do
      do i = 1,n                    ! put data into complex array
         if(i.lt.n/2-lp2+1.or.i.gt.n/2+lp2+1) then
            x(i) = (0.d0 , 0.d0)
         else
            x(i) = dcmplx( pt(i - n/2 + lp2) , 0.d0 )
         endif
      end do
      wdth = tm2 - tm1
      var2 = -((wdth / 2.d0)**2 / 2.d0) / dlog(0.5d0)
      offset = dble(n/2) * tinc
      stdev = sig1
      sig1 = dsqrt(var1 + var2)
      shift = 4.d0 * sig1
      do i = 1,n/2                 ! Compute gaussian for sea state
         if(i.lt.n/4) then
            t = dble(n/2+i-1) * tinc
         y(i) = dcmplx( dexp(-0.5d0*((t-offset)/stdev)**2) , 0.d0)
            t = dble(n/2+i) * tinc
         y(n-i+1) = dcmplx(dexp(-0.5d0*((t-offset)/stdev)**2),0.d0)
         else
            y(i) = (0.d0 , 0.d0)
            y(n-i+1) = (0.d0 , 0.d0)
         endif
      end do
      call fft(x(1),m-1)            ! FFT both functions
      call fft(y(1),m-1)
      do i = 1,n                    ! Multiply FFT's together
         y(i) = dconjg(x(i) * y(i))
      end do
      call fft(y(1),m-1)            ! Inverse FFT the result
      do i = 1,n                    ! Scale the answer
         y(i) = y(i) * tinc / dble(n)
         if(cdabs(y(i)).gt.zmax) zmax = cdabs(y(i))
      end do
      if(n.ge.ns) then             ! Place answer in required array
         do i = 1,ns
            z(i) = cdabs(y((n-ns)/2+i)) / zmax
            write(9,50) float(i) * sngl(tinc),z(i)
         end do
         else
            do i = 1,ns
               if(i.le.(ns-n)/2.or.i.ge.n-(ns-n)/2+1) then
                  z(i) = 0.d0
               else
                  z(i) = cdabs(y((ns-n)/2+i)) / zmax
               endif
               write(9,50) float(i) * sngl(tinc),z(i)
            end do
         endif
         dmax = ptx * zmax / (dsqrt(twopi) * stdev)  ! Compute the max
c
c  Method using gaussians for both functions.
c
      else
         offset = dble(ns/2) * tinc
         var2 = -((wdth / 2.d0)**2 / 2.d0) / dlog(.5d0)
         stdev = dsqrt(var1 + var2)    ! Gaussian convolution
         shift = 4.d0 * stdev
         dmax = ptx * dsqrt(var2) / stdev
         do i = 1,ns                   ! Compute the combined function
            t = dble(i) * tinc
            z(i) = dexp(-.5d0*((t-offset)/stdev)**2)
            write(9,50) float(i) * sngl(tinc),z(i)
         end do
      endif
      k = idnint(shift / tinc)
      write(*,52) dmax
   50 format(1x,f7.3,2x,f11.8)
   51 format(1x,f7.3,2x,d22.15)
   52 format(/' Maximum of the combined distribution is: ',d22.15)
      close(9)
      deallocate( x, y, pt, stat = ier )
      return
      end
```

## A.2.4 FFT CONVOLUTION SUBROUTINE

```
      subroutine convf(mr,mark,tinc,ishft,pm,dm,am,amark,k)
c  1/13/90  M. Newkirk
c  Performs the FFT convolution of PFS and DIST.
c  Subroutine called:
c    FFT   : Computes the Fast Fourier Transform of a data set.
c
      double precision pfs(1),tinc,dm,pm,amax,am
      complex*16 resp[huge](1),ans[huge,allocatable](:)
      dimension conv(1)
      common /pcom/ pfs
      common /rcom/ resp
      common /fcom/ istrt,idif,conv
c
      lfft = 2 ** mr
      allocate( ans(lfft) )
c
c  Using the known maximum of pfs, move pfs into an array of length
c  lfft to be used in the convolution.  Also, set zeros at
c  appropriate places for zeropadding.
c
      lmin = lfft/4
      lmax = 3*lfft/4+1
      lmid = lfft/2
      do i = 1,lfft
         if(i.lt.lmin.or.i.gt.lmax) then
            ans(i) = dcmplx(0.d0,0.d0)
         elseif(mark-lmid+i.le.0) then
            ans(i) = dcmplx(0.d0,0.d0)
         else
            ans(i) = dcmplx(pfs(mark-lmid+i),0.d0)
         endif
      end do
c
c  FFT the FSIR
c
      call fft(ans(1),mr)
c
c  Multiply the result with the RESPNS function, the FT of the point
c  target response / sea state distribution, and conjugate the result
c  in preparation for the inverse Fourier transform. See Brigham, CH 1
c
      do i = 1,lfft
         ans(i) = dconjg(ans(i) * resp(i))
      end do
c
c  FFT the result (actually an inverse FFT) to get the true return
c  waveform
c
```

```
      call fft(ans(1),mr)
c
c  Scale the result to get true amplitude.  (Multiply by T, the
c  sampling interval, and divide by N, the FFT length.  See Brigham,
c  CHs 9 & 13
c
      do i = 1,lfft
         ans(i) = (tinc / dble(lfft)) * ans(i)
      end do
c
c  Find maximum of ans for normalization
c
      amax = 0.d0
      do i = 1,lfft
         if(cdabs(ans(i)).gt.amax) then
            amax = cdabs(ans(i))
            marka = i
         endif
      end do
c
c  Multiply the normalization factors together to get the total
c  normalization factor
c
      amark = float(mark-lmid+ishft+marka+istrt-1) * sngl(tinc)
      am = amax * pm * dm
c
c  Take the answer and shift it back to the true position in time,
c  by knowing the position of its maximum, 'mark' in real time.
c  also normalize CONV to unity using AM from above.
c
      do i = 1,lfft
         ltst = mark-lmid+ishft+i+1
         if(ltst.le.0) cycle
         conv(ltst) = cdabs(ans(i))/sngl(amax)
      end do
      deallocate( ans, stat=ier )
c
c  Make sure that all of the answer is put into output file by testing
c  the magnitude of the last entry. This yields K, the shift that
c  is necessary to get all of the answer into the file.
c
      k = 0
   10 if(conv(idif+k).gt.0.001) then
         k = k + 50
      else
         goto 11
      endif
      goto 10
   11 return
      end
```

## A.2.5 NUMERICAL INTEGRATION CONVOLUTION SUBROUTINE

```
      subroutine convi(lcon,tinc,ishft,pm,dm,am,amark,k)
c   1/15/90  M. Newkirk
c   Performs the numerical integration of the convolution of F and G.
c   No other routines are called
c
      double precision pfs(1),dist(1),h[allocatable,huge](:)
      double precision temp,tinc,amax,am,pm,dm
      dimension conv(1)
      common /pcom/ pfs
      common /dcom/ dist
      common /fcom/ istrt,idif,conv
      allocate( h(8402), stat=ier )
c
c   Convolution by Integration
c
      write(*,*)
      i = 1
      h(1) = 0.d0         ! First point can't be integrated.
      j = 2               ! Begin with second point.
      m2 = lcon/2
   40 h(j) = 0.d0
   20 temp = pfs(j+1-i) * dist(i)
      if(i.eq.1.or.i.eq.j) then
         h(j) = h(j) + temp / 2.d0      ! Trapezoidal Rule
         goto 5
      else
         h(j) = h(j) + temp
      endif
    5 i = i+1
      if(i.gt.j.or.i.gt.lcon) goto 10
      goto 20
   10 h(j) = h(j) * tinc
      j = j + 1
      write(*,25) j
   25 format('+j = ',i5)
      if(j.ge.lcon+idif) goto 30
      if(j.gt.idif) then
         i = j - idif + 1
      else
         i = 1
      endif
      goto 40
   30 continue
c
c   Find maximum of the resulting convolution
c
      amax = 0.d0
      do i = 1,lcon+idif
```

```
         if(h(i).gt.amax) then
            amax = h(i)
            marka = i
         endif
      end do
c
c  Multiply the normalization factors together to get the total
c  normalization factor
c
      amark = float(marka-m2+ishft+istrt-1) * sngl(tinc)
      am = amax * pm * dm
c
c  Place result in output array
c
      do i = 1,lcon+idif
         conv(i) = h(i) / amax
      end do
      deallocate( h, stat=ier )
c
c  Make sure that all of the answer is put into output file by testing
c  the magnitude of the last entry. This yields K, the shift that
c  is necessary to get all of the answer into the file.
c
      k = 0
   60 if(conv(idif+k).gt.0.001) then
         k = k + 50
      else
         goto 50
      endif
      goto 60
   50 return
      end
```

## A.2.6 FFT SUBROUTINE

```
      subroutine fft(x,m)
c  11/7/88  FFT algorithm acquired from A A Beex in EE4980 Fall '87
c  X is a complex vector with data to be FFTed. M defines the length
c  of the vector as follows: N = 2**M. The resulting transformed
c  vector is returned in X.
c
      complex*16 x(1),u,w,t
      double precision pi
c
      n = 2**m
      nv2 = n/2
      nm1 = n-1
      j = 1
      do 7 i=1,nm1
         if(i.ge.j) goto 5
         t = x(j)
         x(j) = x(i)
         x(i) = t
    5    k = nv2
    6    if(k.ge.j) goto 7
         j = j-k
         k = k/2
         goto 6
    7 j = j+k
      pi = 2.d0*dacos(0.d0)
      do 20 l = 1,m
         le = 2**l
         le1 = le/2
         u = dcmplx(1.d0,0.d0)
         w = dcmplx(dcos(pi/dble(le1)),-dsin(pi/dble(le1)))
         do 20 j = 1,le1
            do 10 i = j,n,le
               ip = i+le1
               t = x(ip)*u
               x(ip) = x(i)-t
   10       x(i) = x(i)+t
   20 u = u*w
      return
      end
```

## A.2.7 BESSEL FUNCTION SUBROUTINE

```
      double precision function inot(x)
c  3/05/90  M. H. Newkirk
c  This FUNCTION computes the value of exp(-x)*Io(x) depending on the
c  magnitude of x.  The resulting value is returned in INOT.
c
      double precision x,y,t
c
      t = x/3.75d0
      if(t.le.1.d0) then
c
c  Equation 9.8.1, p.378, Abramowitz & Stegun
c
         y = (1.d0 + 3.5156229d0*t**2   + 3.0899424d0*t**4
     1             + 1.2067492d0*t**6  + 0.2659732d0*t**8
     2             + 0.0360768d0*t**10 + 0.0045813d0*t**12) * dexp(-x)
      else
c
c  Equation 9.8.2, p.378, Abramowitz & Stegun
c
         y = (0.39894228d0+0.01328592d0/t+0.00225319d0/t**2
     1       - 0.00157565d0/t**3+0.00916281d0/t**4-0.02057706d0/t**5
     2       + 0.02635537d0/t**6-0.01647633d0/t**7+0.00392377d0/t**8)
     3       / dsqrt(x)
      endif
      inot = y
      end
```

## A.3 RPM OUTPUT FILE

```
Information for files: pfsf.0   and rpwf.0

Look angle:    .00 degrees; Height:  3048.00 m; Time incr:   .0200 ns

Bandwidth:  152.28 MHz; Noise power: -118.09 dBW

Beamwidth modelled as unequal

Beta value for this look angle:  .413120033126327D-02

Convolution is by FFT, FFT length: 4096

Gain at this look angle is: 46.00 dB

Sigma0:  -5.00 dB; Losses:  10.00 dB

Range of FSIR evaluation time: [    .0000, 12.0000] ns

The leading coefficient for the FSIR =   .92908252D-11

Maximum FSIR value is: .58375976D-01 nW at tau =    .000000 ns

Maximum RPW value is:  .11961356D-01 μW at tau =  12.340000 ns

Maximum S/N is:   38.87 dB

3 dB width of RPW is:   6.897295 ns
```

# B. COMPUTING THE ERROR FUNCTION

In Chapter 5, a test case was developed to convolve a decaying exponential with a Gaussian distribution. The exponential represented the flat surface impulse response for a pointing angle of zero, while the Gaussian was used to approximate the convolved height probability density function of the specular points and radar system point target response. In performing the convolution of these two functions, use was made of an analytical formula found in [1]. This formula allowed the convolution to be completed analytically, providing a convenient check for numerical convolution algorithms.

The only problem associated with this analytical evaluation method is that the resulting expression is very susceptible to errors when evaluated on a digital computer. Embedded in the expression are rapidly varying exponential and error functions that when multiplied together can give widely varying results. Possible solution methods will be investigated, and the best technique for this application will be identified.

## B.1 THE PROBLEM

In Section 5.2, a decaying exponential function, representing $P_{FS}(\tau)$ at $\xi = 0^\circ$, was analytically convolved with a Gaussian distribution, which represented the combined sea state and radar system point target responses, $r(\tau)$. The result of this convolution was shown to be:

$$P_R(\tau) = C\,exp\left\{2\sigma_t^2\left[\left(\frac{2c}{\gamma h}+\frac{2}{\sigma_t}\right)^2 - \frac{2c\tau}{\gamma h\sigma_t^2}\right]\right\}$$

$$\cdot\left[erf\left\{\frac{\tau}{\sqrt{2}\sigma_t} - \sqrt{2}\sigma_t\left(\frac{2c}{\gamma h}+\frac{2}{\sigma_t}\right)\right\}+1\right], \qquad \text{(B.1-1)}$$

where

$$C = \frac{G_o^2\lambda^2 c\sigma^o P_T\sigma_p e^{-8}\sqrt{2\pi}}{8(4\pi)^2 L_P h^3}.$$

The variables $a$, $b$, c, $A$, and $B$ have been substituted into (5.2-10) to obtain (B.1-1).

Careful examination shows that this equation is essentially the product of a decaying exponential and an error function with a constant amplitude term. At first glance this may not seem very troublesome, but noting that $2c/\gamma h \simeq 2.25$ and $\sigma_t \simeq 3.08$ for the MARA system, when $\tau$ is small the exponential is very large while the error function is very small. This can produce unpredictable results when the two are multiplied together by computer, since this operation is done by adding the exponents and multiplying the mantissa portions of the machine representations of these numbers. As $\tau$ becomes larger, this problem becomes less imposing, but $\tau$ must be on the order of 35 ns or more before the exponential and error functions become comparable in magnitude. This is much later than where the peak of the convolved result occurs ( $\sim$ 12.56 ns, see Figure 5.8), meaning that in the neighborhood of where the computation is most important, the result is susceptible to unknown error. In fact, when (B.1-1) was coded into a computer using a polynomial approximation [1] for the error function evaluation, with the expression accurate to 7 decimal places, the result had no resemblance to the expected $P_R(\tau)$ waveform. This illustrated the need for more careful computation of (B.1-1).

## B.2 A POSSIBLE REMEDY

One way to circumvent the problem of evaluating (B.1-1) without dealing with the multiplication of numbers that are several orders of magnitude different is to examine some of the available formulas for approximating $erf(x)$. For example, equation 7.1-26 of [1] is a polynomial approximation;

$$erf(x) = 1 - \left[ \sum_{n=1}^{5} a_n t^n \right] e^{-x^2} + \epsilon(x), \quad x \geq 0 \qquad \text{(B.2-1)}$$

where

$$|\epsilon(x)| \leq 1.5 \times 10^{-7} \qquad t = \frac{1}{1+px}$$

| | |
|---|---|
| $p = 0.3275911$ | $a_1 = 0.254829592$ |
| $a_2 = -0.284496736$ | $a_3 = 1.421413741$ |
| $a_4 = -1.453152027$ | $a_5 = 1.061405429.$ |

Temporarily neglecting the error term $\epsilon(x)$, (B.2-1) can be rearranged as

$$e^{x^2}\left[1 - erf(x)\right] \simeq \sum_{n=1}^{5} a_n t^n, \tag{B.2-2}$$

and noting that $erf(-x) = -erf(x)$ [1], (B.1-1) can be rewritten as

$$P_R(\tau) = C\,exp\left\{2\sigma_t^2\left[\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right)^2 - \frac{2c\tau}{\gamma h \sigma_t^2}\right]\right\} \cdot \left[1 - erf\left\{\sqrt{2}\sigma_t\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right) - \frac{\tau}{\sqrt{2}\sigma_t}\right\}\right]. \tag{B.2-3}$$

Now it can be seen that (B.2-3) looks similar to the left-hand side of (B.2-2). In fact, if the argument of the error function in (B.2-3) is squared and compared to the exponential argument, the terms that create the form of the left-hand side of (B.2-2) can be collected and substituted with the summation, yielding

$$P_R(\tau) \simeq C\,exp\left\{\frac{\tau}{2\sigma_t^2}(8\sigma_t - \tau)\right\}\left[\sum_{n=1}^{5} a_n t^n\right], \quad \tau \le 2\sigma_t^2\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right), \tag{B.2-4}$$

with $t$ and the $a_n$'s are given in (B.2-1). This result behaves functionally as expected in that the exponential peaks for some value of $\tau$ then decays back to zero. There is no multiplication of terms that are several orders in magnitude different, and the limit on $\tau$ is large enough to accommodate the evaluation

($\tau \lesssim 55$ ns).

This evaluation was coded into a computer and evaluated, yielding a waveform having the expected shape and peak location in $\tau$, but with a larger peak amplitude than expected when compared to the numerical results in Chapter 5. Figure B.1 shows the difference in the numerical results (FFT convolution) and the evaluation of (B.2-4).

There is a very subtle reason for this difference. When (B.2-4) is evaluated, it is very important to remember that this expression makes use of a polynomial *approximation*, whose error is guaranteed to be less than $1.5 \times 10^{-7}$[1]. However, when the rearranging of (B.2-1) is made to obtain (B.2-2), the error in the right-hand side of (B.2-2) becomes comparable to its magnitude, as $x \to \infty$. For this reason, the solution given by (B.2-4) is suspect.

## B.3 A SOLUTION

The idea of exploiting the form of the polynomial approximation for $erf(x)$ given in (B.2-1) is a sound one, but the equation itself has been pushed beyond its capabilities. Fortunately, [1] provides several more methods to compute $erf(x)$ and $erfc(x)$ (the complimentary error function). Noting that $erfc(x) = 1 - erf(x)$, equation 7.1.23 of [1] provides an asymptotic approximation, i.e.

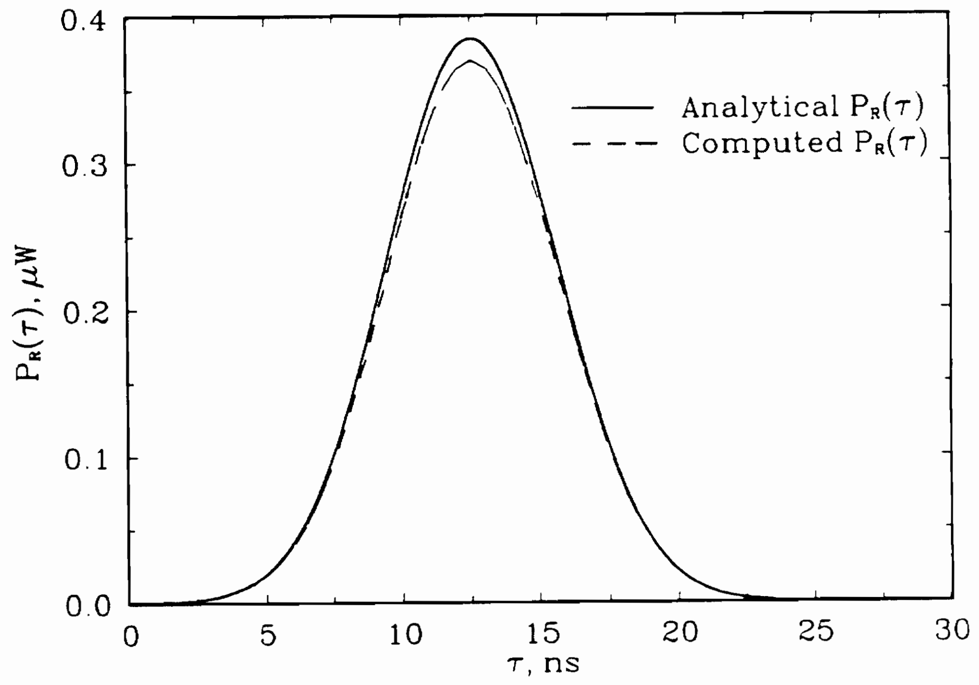


**Figure B.1.** Return power for a nadir pointing angle obtained by FFT and the incorrect analytical convolution methods.

$$\sqrt{\pi}\, x e^{x^2}(1 - erf(x)) \sim 1 + \sum_{m=1}^{\infty} (-1)^m \frac{1 \cdot 3 \cdots (2m-1)}{(2x^2)^m}, \tag{B.3-1}$$

for $x \to \infty$ (large $x$). Where (B.2-1) breaks down, (B.3-1) holds very well. This can be seen by examining both equations as $x \to \infty$; (B.2.1) becomes

$$\lim_{x\to\infty} \sqrt{\pi}\, x e^{x^2}(1 - erf(x)) = \lim_{x\to\infty} \sqrt{\pi}\, x \sum_{n=1}^{5} a_n t^n$$

$$= \frac{\sqrt{\pi} x}{px} = \frac{\sqrt{\pi}}{p} = 5.41057 \tag{B.3-2}$$

and for (B.3-1);

$$\lim_{x\to\infty} \sqrt{\pi}\, x e^{x^2}(1 - erf(x)) = \lim_{x\to\infty} \left\{ 1 + \sum_{m=1}^{\infty} (-1)^m \frac{1 \cdot 3 \cdots (2m-1)}{(2x^2)^m} \right\}$$

$$= 1. \tag{B.3-3}$$

Thus, (B.2-1) does not go to the proper limit as $x \to \infty$, so for large $x$, (B.3-1) is the more reliable approximation of $erf(x)$.

To evaluate (B.1-1), or its alternate form in (B.2-3), once again exploit the $e^{x^2}(1 - erf(x))$ form of (B.3-1) to obtain;

$$P_R(\tau) = \frac{G_o^2 \lambda^2 c \sigma^o P_T \sigma_p e^{-8}\sqrt{2}}{8(4\pi)^2 L_P h^3} \; \frac{exp\left\{\frac{\tau}{2\sigma_t^2}(8\sigma_t - \tau)\right\}}{\sqrt{2}\sigma_t\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right) - \frac{\tau}{\sqrt{2}\sigma_t}}$$

$$\cdot \left\{ 1 + \sum_{m=1}^{\infty} \frac{(-1)^m [1 \cdot 3 \cdots (2m-1)]}{\left[2\left(\sqrt{2}\sigma_t\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right) - \frac{\tau}{\sqrt{2}\sigma_t}\right)^2\right]^m} \right\} \tag{B.3-4}$$

for $\tau < 2\sigma_t^2\left(\frac{2c}{\gamma h} + \frac{2}{\sigma_t}\right)$. As in (B.2-4), the function shows a peaking in the exponential, but now modified by an infinite series summation where (B.2-4) required a summation of only five terms.

The series in (B.3-4) must be evaluated with care due to the behavior of the individual terms. The terms begin large in magnitude, alternating positive and negative, decaying to smaller terms. However, as the number of terms increases, their magnitude grows toward infinity, still alternating between positive and negative. To illustrate this, consider evaluating the series terms for (B.3-1) for $x = 3$ and $x = 5$. Table B.1 shows the first thirty terms of (B.3-1) for $x = 3$ and the first forty terms for $x = 5$, along with the cumulative sum for each. Note that for $x = 3$, the magnitude of these terms decays to a small value after only the fifth term, but after the tenth term, they begin to grow again. It is interesting to note that as the magnitude of $x$ increases, the number of terms that must be summed before this divergence occurs also increases. The values in Table B.1 for $x = 5$ show how the terms decay to a smaller magnitude than for $x = 3$, but they do not begin to diverge until several terms later. Thus as $x$ becomes larger, the sum converges to a more accurate approximate value, while the number of terms before the series diverges increases. In the MARA system case, the equivalent value of $x$ ranges from roughly 12.6 when $\tau = 0$ ns to about 6.9 when $\tau = 25$ ns, so the divergence problem is not an imposing problem.

An important point to stress is that (B.3-1) is an asymptotic series, meaning that only a finite number of terms should be used in the summation. As was seen in Table B.1, asymptotic series first converge to some value then eventually diverge. Fortunately, deciding which term should be the last when

Table B.1. Comparison of series terms and cumulative sums in (B.3-1) for $x = 3$ and $x = 5$.

| | $x = 3$ | | $x = 5$ | |
|---|---|---|---|---|
| $m$ | Term Value | $1 + \text{Sum}$ | Term Value | $1 + \text{Sum}$ |
| 1 | -.05555555555 | .94444444444 | -.02000000000 | .98000000000 |
| 2 | .00925925925 | .95370370370 | .00120000000 | .98120000000 |
| 3 | -.00257201646 | .95113168724 | -.00012000000 | .98108000000 |
| 4 | .00100022862 | .95213191586 | .00001680000 | .98109680000 |
| 5 | -.00050011431 | .95163180155 | -.00000302400 | .98109377600 |
| 6 | .00030562541 | .95193742696 | .00000066528 | .98109444128 |
| 7 | -.00022072946 | .95171669750 | -.00000017297 | .98109426830 |
| 8 | .00018394122 | .95190063872 | .00000005189 | .98109432019 |
| 9 | -.00017372226 | .95172691645 | -.00000001764 | .98109430255 |
| 10 | .00018337350 | .95191028996 | .00000000670 | .98109430926 |
| 11 | -.00021393575 | .95169635420 | -.00000000281 | .98109430644 |
| 12 | .00027336234 | .95196971655 | .00000000129 | .98109430774 |
| 13 | -.00037966992 | .95159004662 | -.00000000064 | .98109430709 |
| 14 | .00056950489 | .95215955152 | .00000000035 | .98109430744 |
| 15 | -.00091753566 | .95124201586 | -.00000000020 | .98109430723 |
| 16 | .00158020030 | .95282221616 | .00000000012 | .98109430736 |
| 17 | -.00289703389 | .94992518227 | -.00000000008 | .98109430728 |
| 18 | .00563312145 | .95555830372 | .00000000005 | .98109430734 |
| 19 | -.01157919410 | .94397910962 | -.00000000004 | .98109430729 |
| 20 | .02508825388 | .96906736351 | .00000000003 | .98109430733 |
| 21 | -.05714546718 | .91192189633 | -.00000000002 | .98109430730 |
| 22 | .13651417159 | 1.04843606793 | .00000000002 | .98109430732 |
| 23 | -.34128542899 | .70715063893 | -.00000000002 | .98109430730 |
| 24 | .89113417571 | 1.59828481464 | .00000000002 | .98109430732 |
| 25 | -2.42586525610 | -.82758044145 | -.00000000002 | .98109430730 |
| 26 | 6.87328489229 | 6.04570445083 | .00000000002 | .98109430732 |
| 27 | -20.23800551619 | -14.19230106536 | -.00000000002 | .98109430730 |
| 28 | 61.83835018838 | 47.64604912302 | .00000000002 | .98109430732 |
| 29 | -195.82144226321 | -148.17539314019 | -.00000000002 | .98109430730 |
| 30 | 641.85917186274 | 493.68377872255 | .00000000003 | .98109430733 |
| 31 | | | -.00000000003 | .98109430729 |
| 32 | | | .00000000004 | .98109430734 |
| 33 | | | -.00000000006 | .98109430728 |
| 34 | | | .00000000008 | .98109430736 |
| 35 | | | -.00000000011 | .98109430724 |
| 36 | | | .00000000016 | .98109430741 |
| 37 | | | -.00000000024 | .98109430717 |
| 38 | | | .00000000036 | .98109430753 |
| 39 | | | -.00000000055 | .98109430697 |
| 40 | | | .00000000087 | .98109430785 |

truncating the series is not difficult. The algorithm that computes this series should look at each term as it is computed and compare that value to the magnitude of the previous term. Truncating the series when that term exceeds the previous term in magnitude will result in a very close approximation to the sum, accurate to several decimal places.

Using this method, the resulting $P_R(\tau)$ waveform agrees quite well with the numerical results, as was seen in Section 5.2.3. Figure 5.8 gave the comparison of this analytical result with the numerical evaluations, and as seen there, the three methods give virtually identical output. Thus, accurate evaluation of the error function can be obtained with careful use of the approximate formulas.

# VITA

Michael Newkirk was born in Elmer, New Jersey on July 6, 1965, where he worked on the family dairy farm through high school. He graduated in 1983 from Woodstown High School in New Jersey and obtained his Bachelor of Science degree in Electrical Engineering from Virginia Tech in 1988.

He has worked during the summers of 1988 and 1989 at the U. S. Naval Research Laboratory in Washington, DC, in the Space Sensing Branch of the Space Systems division. While there he performed testing and data aquisition on a 95 GHz short pulse radar altimeter.

He is currently continuing his graduate work on multibeam radar altimeters at Virginia Tech as a Bradley Fellow, pursuing a Doctorate degree in Electrical Engineering.

Michael H. Newkirk