A MATHEMATICAL MODEL FOR THE DETECTION OF DEEP SPACE OBJECTS

by

Susan R. Garrett

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

Mathematics

APPROVED:

May, 1987

Blacksburg, Virginia

A MATHEMATICAL MODEL FOR THE DETECTION OF DEEP SPACE OBJECTS

by

Susan R. Garrett

(ABSTRACT)

The problem of detecting deep space objects with certain probabilities was investigated. A mathematical model was then developed from given problem specifications that deals with the trade-off of various parameters involved in the detection problem.

A software package that allows the user to input data interactively was written to implement the model. The completed program as well as an analysis of the tested results are included.

ACKNOWLEDGEMENTS

This thesis is dedicated to my parents, Rev. and Mrs. William H. Garrett, and my fiancé, Karl Kalbaugh, who have been a constant source of encouragement, support and love.

I would like to thank Dr. Terry Herdman, my committee chairman and advisor, and Dr. John Burns for their help and encouragement throughout this project. I would also like to thank Dr. Margaret Murray and Dr. Eugene Cliff for consenting to be on my committee.

TABLE OF CONTENTS

DESCRIPTION OF THE PROBLEM

Introduction

The purpose of this project is to develop a numerical scheme that will effectively allow the user to trade-off various parameters involved in the detection of space objects. The space objects are referred to as targets because they are the target of detection. The targets can be objects in some earth orbit or they can be missiles with known launch and impact sites. In order to detect the targets, a visual range telescope is used. The telescope, referred to as the sensor, will be space-based, operating in orbit. Thus, we have a sensor in some orbit about the earth that can view objects in space. The view of the target through the sensor is assumed to be against a background of stars only. For simplicity, nothing else will be considered in the detection of the target other than the target and stars.

This view of the target against the starfield background is called the total field of view. Part of the actual hardware of the sensor, is a fine mesh grid through which the total field of view can be seen and divided. This is called the focal plane grid of detectors. Each division of the grid is a square of equal size to the other divisions. These divisions are called pixels and act as the detectors.

The sensor must have some device that does the actual detection of a target. The sensor is equipped with an amplifier that can collect or record the number of electrons of any object in the field of view. The number of electrons collected in each pixel is recorded and is assumed to follow a Gaussian distribution. This is done more than once. Each time electrons are collected it can be thought of as though a picture has been taken. Each one of these "pictures" is referred to as an exposure and is of some given time length. The whole problem comes down to finding out how many exposures are needed and how much time each one should take in order to adequately detect a target.

This method for detecting a target compares each subsequent exposure. Since we assumed that the background is a starfield only, the number of electrons collected from the target should move from one pixel to the next in each exposure while the number of electrons collected due to stars remains constant. It is assumed that the grid can be oriented so that the target moves in a straight line from one pixel to the next.

There are certain elements that come into play that make a target harder to detect. Up to this point, we have only discussed collecting electrons due to the target itself and to stars. Because they are recorded through an amplifier, it is possible to pick up electrons due to noise from the amplifier. This must be taken into account. It is also possible to detect what is called a false alarm. This happens when a star is mistaken for a target. False alarms must also be taken into account. One other thing that has not been mentioned is the idea of a threshold level. The threshold level is a number below which no electrons are recorded. This makes it possible to have a target and not detect it because the number of electrons collected from it falls below the threshold level. All of these things affect the accuracy with which a target can be detected.

An overview of what is physically taking place is this: There is a visual range telescope operating in an orbit about the earth. It has the ability to take timed exposures of objects in its field of view by collecting electrons from the objects. Each exposure is compared to the one preceding it and the one succeeding it to find out if the path of the target can be distinguished.

The rest of this chapter goes through the details of the problem and the mathematical equations that arise. Chapter two deals with the development of the computer program and the last chapter gives an analysis of the tested results. A copy of the computer code of the program can be found in Appendix D.

The Sensor

The sensor, operating in orbit about the earth, has the following known orbit data supplied as input by the user:

a) Altitude of sensor at apogee (NM: nautical miles)

b) Altitude of sensor at perigee (NM) or in place of a) and b)

- period of revolution (HR)
- eccentricity

c) Inclination (degrees)

d) Initial true anomaly (deg)

e) Longitude of ascending node with respect to Greenwich (deg)

f) Argument of perigee (deg) - if the orbit is non-circular

The aperture of the sensor is the diameter, D (meters), and is also supplied on input by the user. Other input data that will be needed includes the following:

1) Field of view, Ω (square degrees), of the sensor
2) Combined efficiency, ε, of the optics and detectors in the sensor
3) Total number of pixels, N, that make up the full focal plane array of detectors in the sensor. The focal plane grid is divided into squares which act as electron detectors.
4) Number of diffraction limited pixel diameters, K, that make up an actual pixel
5) The galactic latitude, Φ_g (deg)
6) Standard deviation of the number of electrons due to noise, σ_r
7) Visual magnitude of the target, M_{VT} or, if not known, the following may be input:

   Distance from the sensor to the target, R(m).

   Reflectivity - area product, σ_R(m).

   Sun angle subtended from sensor to target to sun, γ(deg).

M_{VT} can then be found using the following formula from [3]:

$$M_{VT} = -26.78 - 2.5 \log \left(\frac{\sigma_R \, F(\gamma)}{R^2} \right) \tag{1}$$

where

$$F(\gamma) \simeq \frac{2}{3\pi^2} \left[(\pi - \gamma) \cos \gamma + \sin \gamma \right] \tag{2}$$

The sensor will try to detect a target in the field of view that also contains stars. This is achieved by examining a time-sequenced row of pixels yielding a number of electrons from an amplifier read-out above a certain threshold level, n_o. In other words, an exposure is made and each pixel outputs the number of electrons exceeding the threshold level. For a detection to take place, subsequent exposures are made showing adjacent "flipped" pixels. A "flip" occurs when a pixel outputs electrons above the threshold level. The number of exposures needed, Q, depends upon given probabilities. The number of exposures needed is the major calculation of the program and is therefore an important output data item.

A detection is defined as a set of Q flips, one in each of the adjacent Q exposures forming a straight line. On input, the probability of detection for a target in the field of view, PD, will be required as input in order to obtain the probability of detection for a target in a pixel field of view, pd. A lower bound on pd is obtained by ignoring the enhancement effects of stars and readout noise. Thus,

$$PD = pd^Q \ . \tag{3}$$

The time for each exposure is referred to as the stare time, τ_{exp} (sec). The total stare time is denoted τ_{stare}. Since each exposure is of duration τ_{exp}, we have:

$$\tau_{stare} = Q\ \tau_{exp} \ . \tag{4}$$

During the total stare time, the stars remain fixed in the field of view and a target moves across the field of view. τ_{stare} and τ_{exp} are also given as output.

Since the total number of pixels, N, and the field of view of the sensor, Ω, are given as input, the pixel field of view, $\Delta\Omega$, can easily be found:

$$\Delta\Omega = \frac{\Omega}{N} \tag{5}$$

For any value of $\Delta\Omega$, τ_{exp} is to be computed so that the anticipated target just crosses the pixel width. Thus,

$$\frac{\sqrt{\Delta\Omega}}{\tau_{exp}} \cong \omega \tag{6}$$

where ω is the angular rate of the target across the field of view. ω is calculated from the orbital input data.

The following additional assumptions have been made in order to narrow down the scope of the problem:

1) The required statistics on the number of electrons relevant to the detection of targets, stars, and noise can be approximated by Gaussian distribution.

2) The focal plane grid geometry of the telescope is made up of an array of square detectors.

3) A combined efficiency of the optics is approximately constant over the wavelength.

4) The magnitude of the target does not vary significantly over the time length of an exposure.

False Alarms

A false alarm occurs during a stare when a detection occurs and a target is not present. This may be caused by some combination of stars and/or noise resulting in an electron readout above the threshold level. On input, the probability of a false alarm, P_{fa}, as well as a number, X, between 0 and 1 representing the ability to remove stars as false alarm candidates (1 = worst) will be needed.

False alarms caused by noise are due to the amplifier itself. A certain number of electrons will be added or subtracted by the amplifier. We assume that this noise has a mean of 0 and a standard deviation of σ_r. In dealing with false alarms due to stars, the average number of stars per steradian in the magnitude decrement dm_v is used (see Appendix A):

$$\left(\frac{dz}{dm_v}\right) dm_v \ . \tag{7}$$

Also, measured values of $\frac{dz}{dm_v}$ are given in Appendix A and are dependent on the visual magnitude, m_v, and the galactic latitude, Φ_g. Inserting the factor, X, (7) becomes

$$X \left(\frac{dz}{dm_v}\right) dm_v \ , \tag{8}$$

the effective number of stars per steradian in the magnitude decrement dm_v.

Threshold Setting

It was previously stated that a pixel is said to have "flipped" if the number of readout electrons exceeds some threshold value n_o.

The number of electrons is a combination of electrons from noise, stars, and the target.

In order to calculate n_o, we need to examine a few things. The number of electrons collected in a pixel due to the presence of an object of visual magnitude m_v will be denoted as $n(m_v)$ and the flux (photons/m^2/sec) at the sensor aperture will be denoted as F. Then, the flux of photons corresponding to an object of visual magnitude m_v is (see Appendix B):

$$F(m_v) = 5.76 \times 10^{10} \, e^{-.92 \, m_v} \, . \tag{9}$$

Thus, the mean number of electrons collected is:

$$n(m_v) = \frac{\pi D^2}{4} \, \varepsilon \, \tau_{exp} \, F(m_v) \tag{10}$$

or using equation (6):

$$n(m_v) = \frac{\pi D^2}{4} \, \frac{\sqrt{\Delta\Omega}}{\omega} \, \varepsilon \, F(m_v) \, . \tag{11}$$

Going back to one of the original assumptions regarding the electrons collected as following a Gaussian distribution, then:

$$pd = 1 - \frac{1}{\sigma_{eff}\sqrt{2\pi}} \int_{-\infty}^{n_o} e^{\frac{1}{2}\left(\frac{n_{TN} - \bar{n}_T}{\sigma_{eff}}\right)^2} dn_{TN} \, . \tag{12}$$

n_{TN} is the number of electrons collected due to a target plus noise during one exposure, $\bar{n}_T$ is the expected mean from equation (11) where the object is the target (i.e. $m_v = m_{VT}$), and σ_{eff} is the effective standard deviation of the number of electrons due to a target plus noise.

$$\sigma_{eff} = \sqrt{\sigma_T^2 + \sigma_r^2} \tag{13}$$

where

$$\sigma_T = \sqrt{\overline{n}_T} \ . \tag{14}$$

The lower limit on the integral in equation (12) should actually be zero but the approximation of using $-\infty$ is very small (no larger than that of assuming a Gaussian distribution). If ξ_{oT} is the number of standard deviations of n_o below $\overline{n}_T$, then

$$\xi_{oT} = \frac{\overline{n}_T - n_o}{\sigma_{eff}} \ . \tag{15}$$

We can now write pd as:

$$pd = 1 - \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\frac{\xi_{oT}}{\sqrt{2}}} e^{-x^2} dx \ . \tag{16}$$

Using the Gaussian error function:

$$\mathrm{erf}(x) \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \ , \tag{17}$$

and noting that:

$$\frac{2}{\sqrt{\pi}} \int_{-\infty}^0 e^{-t^2} dt = 1 \ , \tag{18}$$

pd becomes:

$$pd = 1 - \frac{1}{2} + \frac{1}{2} \mathrm{erf} \left(\frac{\xi_{oT}}{2} \right) \ . \tag{19}$$

Hence, from equation (3),

$$PD^{1/Q} = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left(\frac{\xi_{oT}}{\sqrt{2}}\right) \quad . \qquad (20)$$

Given the value of PD and Q, the value of ξ_{oT} can be found using the inverse error function. Once ξ_{oT} is known, n_o can be calculated using equation (15).

<u>Probability of Star Presence in a Pixel</u>

The effective number of stars per steradian in the magnitude decrement dm_{vs} is $X\left(\frac{dz}{dm_{vs}}\right) dm_{vs}$ (see (8)) where m_{vs} is the visual magnitude of a star. If we let the probability, P, be $\frac{1}{N}$ that a given star will be in a given pixel, we can find P_s, the probability of star presence in that pixel.

To calculate P_s, we add the probability of finding 1,2,...,20 stars in one pixel. The probabilities above 20 are extremely small and the resulting exponents extremely large to be handled easily; therefore, we need only go as high as 20. Thus,

$$P_s = \binom{m}{1} P(1-P)^{m-1} + \binom{m}{2} P^2(1-P)^{m-2} + \ldots + \binom{m}{20} P^{20}(1-P)^{m-20} \qquad (21)$$

where m is the total number of stars at the given galactic latitude.

Equation (21) gives the probability of star presence in a pixel where there are m stars in the total field of view. We will be integrating this quantity over visual star magnitudes so that m should be taken in a small magnitude decrement. Thus,

$$P_s(m_{vs})\, dm_{vs} = X\,(P_s)\, dm_{vs} \quad . \qquad (22)$$

It should be noted that m depends on m_{vs} so that P_s is actually a function of m_{vs}, $P_s(m_{vs})$.

Flip Probability of a Pixel with no Target Present

If we assume that no target is present in a pixel, the probability $P_{f\overline{T}}$ is:

$$P_{f\overline{T}} = \text{Prob (flip due to noise and no star)}$$
$$+ \text{ Prob (flip due to star presence plus noise)}$$
$$= P_{Nf}\,[1 - \int_0^\infty P_s(m_{vs})\,dm_{vs}] + \int_0^\infty P_{sNf}(m_{vs})\,P_s(m_{vs})\,dm_{vs}\ . \tag{23}$$

But, it must be noted that:

$$\int_0^\infty P_s(m_{vs})\,dm_{vs} = 1\ , \tag{24}$$

which makes the first term in equation (23) drop out. However, for extremely large values of m_{vs} (i.e. extremely dim stars), $P_s(m_{vs})$ should eventually equal 1. This would make the integral in equation (24) blow up (approach infinity). Therefore, once the star presence in a pixel is equal to 1, we are no longer interested in whether or not dimmer stars exist in the pixel. Thus, it suffices to find xx such that:

$$\int_0^{xx} P_s(m_{vs})\,dm_{vs} = 1\ . \tag{25}$$

Thus, equation (23) becomes:

$$P_{f\overline{T}} = \int_0^{xx} P_{sNf}(m_{vs})\, P_s(m_{vs})\, dm_{vs} \quad . \tag{26}$$

P_{sNf} is the probability of a flip due to star presence plus readout noise. We have already found an equation for P_s, now one must be found for P_{sNf}.

The number of standard deviations of n_o below $\overline{n}_s$ will be denoted as $\xi_{sNo}(m_{vs})$. Then:

$$\xi_{sNo}(m_{vs}) = \frac{n_o - \overline{n}_s(m_{vs})}{\sqrt{\sigma_{sN}}} \tag{27}$$

where $\overline{n}_s(m_{vs})$ is the mean number of electrons collected from a star with visual magnitude m_{vs} (see equation (11)) and σ_{sN} is the standard deviation of the number of electrons due to star presence plus noise:

$$\sigma_{sN} = \sqrt{\sigma_s^{\,2} + \sigma_r^{\,2}} \quad , \quad \sigma_s = \sqrt{\overline{n}_s} \quad . \tag{28}$$

Then, as in equation (19):

$$P_{sNf}(m_{vs}) = \frac{1}{2} - \frac{1}{2}\,\mathrm{erf}\left(\frac{\xi_{sNo}(m_{vs})}{\sqrt{2}}\right) \quad . \tag{29}$$

<u>Derivation of Q</u>

The probability of a false alarm will be denoted by P_{fa}. $P_{f\overline{T}}$ and P_{fa} are related by Q, the number of exposures needed, by the following equation:

$$P_{f\overline{T}} \cong \left(\frac{P_{fa}}{N}\right)^{1/Q} \quad . \tag{30}$$

We can now solve this equation for Q:

$$Q = \frac{\ell n\left(\frac{P_{fa}}{N}\right)}{\ell n(P_{f\overline{T}})} \quad . \tag{31}$$

The procedure for evaluating Q will be to first make an initial guess of Q. Then, using equation (31), iterate until convergence is met. The value or function $P_{f\overline{T}}$ can be considered a function of Q. Therefore, an initial guess is required in order to evaluate equation (31). One or more convergence schemes may be needed. Once Q has been found, calculate τ_{stare} from equation (4).

One other relationship should be mentioned. D, the diameter, and $\Delta\Omega$, the pixel field of view, can be related by diffraction theory. If we denote the field subtended by the entire central peak of the diffraction pattern of a point object as $\Delta\Omega_{DL}$, then:

$$\Delta\Omega_{DL} \cong \frac{4\ \overline{\lambda}^2}{\pi\ D^2/4} \tag{32}$$

where $\overline{\lambda}$ is the mean wavelength of the spectrum under consideration:

$$\overline{\lambda} \cong 0.5 \times 10^{-6} \quad . \tag{33}$$

Introducing K, the number of diffraction limited pixel diameters that make up an actual pixel, we get:

$$\Delta\Omega = K^2\ \Delta\Omega_{DL} \quad . \tag{34}$$

Thus,

$$\Delta\Omega = \frac{K^2\ 4\overline{\lambda}^2}{\pi\ D^2/4} \tag{35}$$

The computer program that will carry out all the calculations presented in this chapter, will allow the user to estimate some of the input parameters to see their effect on Q and τ_{stare}. The user will then be able to trade-off any input values for another to see the effects. The following items should be printed and presented to user on output:

Number of exposures needed - Q

Exposure time - τ_{exp}

Total stare time - τ_{stare}

Threshold level - n_o .

DEVELOPMENT OF THE PROGRAM

Major Calculations

The following are the major components of the program:

1) A main program structure

2) A subroutine for the purpose of allowing the user to input data interactively

3) Additional subroutines to partition the major evaluations into many shorter calculations

4) Output displayed for the user

5) The user should then be allowed to go back and alter any of the original input data to see what effects the changes make.

From Chapter One, the major calculation of the program is (see equation (31)):

$$Q = \frac{\ell n \left(\frac{P_{fa}}{N} \right)}{\ell n (P_{f\overline{T}})} \quad . \tag{36}$$

P_{fa} and N are entered on input so only $P_{f\overline{T}}$ needs to be calculated. In order to do this, the calculation of $P_{f\overline{T}}$ can be broken down and traced back through a series of equations. From the following list of equations, $P_{f\overline{T}}$ can be found from the input data:

$$P_{f\overline{T}} = \int_0^{xx} P_{sNf}(m_{vs})\; P_s(m_{vs})\; dm_{vs} \tag{37}$$

- $P_s(m_{vs})\ dm_{vs} = x\ P_s\ dm_{vs}$ (38)

- $$P_s = \binom{m}{1} P(1-P)^{m-1} + \ldots + \binom{m}{20} P^{20}(1-P)^{m-20} \quad (39)$$

- xx is such that $\int_0^{xx} P_s(m_{vs})\ dm_{vs} = 1$ (40)

- $$P_{sNf}(m_{vs}) = \frac{1}{2} - \frac{1}{2}\ \text{erf} \left(\frac{\xi_{sNo}(m_{vs})}{\sqrt{2}} \right) \quad (41)$$

- $$\xi_{sNo}(m_{vs}) = \frac{n_o - \bar{n}_s(m_{vs})}{\sigma_{sN}} \quad (42)$$

- $$\bar{n}_s(m_{vs}) = \frac{\pi D^2}{4} \frac{\sqrt{\Delta\Omega}}{\omega}\ \varepsilon\ F(m_{vs}) \quad (43)$$

- $$\Delta\Omega = \frac{K^2\ 4\bar{\lambda}^2}{\pi D^2/4} \quad (44)$$

- $\bar{\lambda} = 0.5 \times 10^{-6}$ (45)

- $F(m_{vs}) = 5.76 \times 10^{10}\ e^{-.92\ m_{vs}}$ (46)

- ω is calculated from orbital input data

- $$\sigma_{sN} = \sqrt{\sigma_s^2 + \sigma_r^2} \quad (47)$$

- $\sigma_s = \sqrt{\bar{n}_s}$ (48)

- $n_o = \bar{n}_T - \xi_{oT}\ \sigma_{eff}$ (49)

- $$\bar{n}_T = \frac{\pi D^2}{4} \frac{\sqrt{\Delta\Omega}}{\omega}\ \varepsilon\ F(m_{vT}) \quad (50)$$

$$\bullet\ \sigma_{eff} = \sqrt{\sigma_T^{\ 2} + \sigma_r^{\ 2}} \tag{51}$$

$$\bullet\ \sigma_T = \sqrt{\bar{n}_T} \tag{52}$$

$$\bullet\ \xi_{oT} = \sqrt{2}\ \mathrm{erf}^{-1}\ (2(PD)^{1/Q} - \frac{1}{2}) \tag{53}$$

$P_{f\bar{T}}$ is found using the following input data items:

PD, Q, σ_r, Ω, N, ε, m_{vT}, D, X, Φ_g .

m is also needed to find P_s but is not input by the user. It is calculated externally to the program itself and will be discussed in the next section. Solving the equations above requires the use of numerical integration and various convergence algorithms.

<u>Evaluation of m</u>

m is the number of stars at a given galactic latitude for some visual magnitude decrement. For each run of the program, Φ_g remains constant. Thus, m can be thought of as a function of visual star magnitude: $m \equiv m(m_{vs})$. Because we will be integrating over all visual star magnitudes in the integral of P_s, it will be helpful to find the function $m(m_{vs})$. For any given value of Φ_g, there is one function $m(m_{vs})$. Hence, in order to save computing time, a function $m(m_{vs})$ was calculated for specific values of Φ_g and stored. For this reason, the user is given a list of values for Φ_g to choose from. The list is reasonable and should enable the user to find an appropriate Φ_g. The choices are:

0°, ± 5°, ± 10°, ± 20°, ± 30°, ± 40°, ± 50°, ± 60°,

± 70°, ± 80°, ± 90° .

In order to find $m(m_{vs})$, the table of star numbers from [1] found in Appendix A was used. It shows the logarithm of the number of stars per square degree brighter than some photographic magnitude m (this m is not the same as the function $m(m_{vs})$). For each of the values in the table, 10^{value} was calculated to get the actual number of stars. These new values were then shifted up by .7 to take into account visual rather than photographic magnitudes. Then each value was subtracted from its immediate predecessor to obtain the number of stars per square degree within the brightness range $m + \frac{1}{2}$ to $m - \frac{1}{2}$. The logarithm of these values was then taken. It is known that the graph of the logarithm of these values forms a straight line. Thus, using a straight forward least squares routine, a straight line was fit to each set of values at each galactic latitude. The values at the extremes of the table were disregarded to alleviate possible error. The slope and intercept of each line for each galactic latitude are stored in the file LINES.DAT. At the start of the program, once the galactic latitude has been input, the appropriate slope and intercept are read into the variables SLOPE and YCEPT. Thus,

$$m(m_{vs}) = \text{SLOPE}(m_{vs}) + \text{YCEPT} . \tag{54}$$

With the value m as a known function, P_s is now a known function of the visual star magnitude. P_s is calculated in the subroutine PS (see Appendix D, D23).

Evaluation of $\int_0^{xx} P_s(m_{vs})\, dm_{vs}$

Once P_s is found as a function of m_{vs}, $\int P_s(m_{vs})\, dm_{vs}$ can be evaluated. xx needs to be found such that $\int_0^{xx} P_s(m_{vs})\, dm_{vs} \approx 1$. This was achieved in two steps. The subroutine LOWBND (see Appendix D, D20) is used to bracket the integral value 1 between two successive unit increments. Initially, $\int_0^{1} P_s(m_{vs})\, dm_{vs}$ is evaluated. The integral is then evaluated with an upper limit of 2. The upper limits are increased by 1 each time until the values of the integral at two successive upper limits bracket the value 1. Testing the integral at various galactic latitudes showed that, generally, a value between 10 and 20 for the upper limit yielded an integral value close to 1. Because these values for xx are visual star magnitudes this is expected. 20 is an extremely dim magnitude. The reason for bracketing the value 1 in this way is to insure better results when getting the integral to be sufficiently close to 1.

Newton's method is used in the subroutine UPLIMIT (see Appendix D, D22) to achieve convergence. Newton's method was chosen for two reasons. First, using the subroutine LOWBND gives a good initial estimate for Newton's method which greatly increases its rate of convergence. The lower of the two successive unit increments is used as the initial estimate. Secondly, Newton's method requires that the derivation of the function involved be calculated. Because we are dealing with a definite integral as our function, the derivative is easily found:

$$f(x) = \int_0^x P_s(m_{vs})\, dm_{vs} - 1 \tag{55}$$

and

$$f'(x) = P_s(x) \quad . \tag{56}$$

The added time involved using the subroutine LOWBND is minimal and is offset by the increase in the rate of convergence using Newton's method. The algorithm for Newton's method used in the subroutine UPLIMIT is adapted from [4].

In order to evaluate the integral, the subroutine GAUSS (see Appendix D, D21) is called. GAUSS uses the Gaussian Quadrature method of integration and was adapted from a previously written subroutine, GAUSSQ.

Once the root of f(x) (see equation (55)) has been found (i.e. convergence has been met) in the subroutine UPLIMIT, the root, X, is returned in the variable xx. xx is the upper limit of the integral. xx is then used in the evaluation of $\int_0^{xx} P_{sNf}(m_{vs})\, P_s(m_{vs})\, dm_{vs}$.

The subroutine GAUSS will be used in the evaluation of this integral also. P_{sNf} must now be evaluated.

Evaluation of $\int_0^{xx} P_{sNf}(m_{vs}) \, P_s(m_{vs}) \, dm_{vs}$

In order to evaluate $\int_0^{xx} P_{sNf}(m_{vs}) \, P_s(m_{vs}) \, dm_{vs}$, it is necessary to find a function for $P_{sNf}(m_{vs})$. There are many calculations involved in finding P_{sNf} as can be seen in the list of equations in the first section of this chapter. Some of these calculations are straight forward and need no explanation. The following is a list of variables and the subroutine or function where they are evaluated:

Variable	Routine	Appendix D
$\Delta\Omega$	PFIELDV	D5
ω	DOMEG	D17
$\bar{n}_T$	MEANT	D9
m_{VT}	TARMAG	D4
σ_{eff}	SDEV	D8
ξ_{oT}	SDEVOT	D6
n_o	THRESH	D7
$\bar{n}_s, \sigma_s, \sigma_{sN}$	PSNF	D24

Some of the variables used in the evaluation of P_{sNf}, such as ξ_{sNo} and F(M) are never actually defined in the program. Rather, their equivalent form is used and never named. For example, in place of the variable ξ_{sNo}, $\frac{n_o - \bar{n}_s}{\sigma_{sN}}$ is used and never named as a single value.

It should be noted that in order to calculate ω using DOMEG, the position and velocity vectors of both the sensor and the target are needed. The subroutine, SBSIN (see Appendix D, D15) prompts the user for orbital data for both the sensor and a non-missile target. It then calculates the position and velocity vectors for both. The requested input data is listed as a) through f) in the second section of the first chapter. In the case where the target is a missile, the subroutine MISSIL (see Appendix D, D16) prompts the user for launch and impact information and calculates the position and velocity vectors. The required input data is:

Launch site:	latitude (deg)
	longitude (deg)
	altitude (ft)
Impact site:	latitude (deg)
	longitude (deg)
	altitude (ft)

DOMEG, SBSIN, MISSIL are all previously written and tested subroutines that are used only in the calculation of ω.

The function SDEVOT calls the function INVERF (see Appendix D, D18) which evaluates the inverse error function. The function ERF (see Appendix D, D19) evaluates the error function. ERF is used in the function PSNF since $P_{sNf} = \frac{1}{2} - \frac{1}{2} \text{ erf } \left(\frac{\xi_{sNo}(m_{vs})}{\sqrt{2}} \right)$. Both INVERF and ERF were adapted from previously written and tested routines.

Evaluation of Q

Once P_{sNf} and P_s have been found, using an initial estimate for Q, the next value for Q can be evaluated. The main objective of the program from a mathematical viewpoint is to find the convergent value of Q. Because the program fails to yield any information if Q does not converge, two separate converging routines are used. The second is used in case the first fails. The two subroutines are CONVERG1 (see Appendix D, D11) and CONVERG2 (see Appendix D, D12).

CONVERG1 uses the x = G(x) method, also known as the method of iteration. This method was chosen because of the nature of the equation. It is already of the correct form:

$$Q = \frac{\ell n \left(\frac{P_{fa}}{N}\right)}{\ell n \, (P_{f\overline{T}})} = G(Q) \; . \qquad (57)$$

$P_{f\overline{T}}$ is dependent upon Q, therefore, using the notation, G(Q), is valid. CONVERG1 calls the function QFUN (see Appendix D, D13) to do the actual calculation of G(Q). QFUN then calls the other functions and subroutines previously mentioned in the evaluation of $P_{f\overline{T}}$. CONVERG1 was adapted from the program, PXGXIT, found in [4].

In case CONVERG1 fails to yield convergence, the subroutine CONVERG2 is used as a second attempt to reach convergence. This subroutine uses Newton's method with an initial estimate from the original input guess for Q. Since Q is expected to be approximately between 1 and 10, using the initial input guess provides an adequate estimate. The method used here is a similar adaptation of the one used earlier taken from [4].

Once a value of Q has been reached, the results are output and control is transferred back to the user.

Other Subroutines

Some of the subroutines that have not been mentioned yet do not directly effect the value of Q, although they do effect the output. The functions TEXP (see Appendix D, D10) and TSTARE (see Appendix D, D3) evaluate τ_{exp} and τ_{stare} respectively. These are both straight forward calculations (see equations (6) and (4)) and the results are included in the output.

The subroutine INPUT (see Appendix D, D2) is called from the main program to prompt the user for the needed input data. INPUT performs two initial calculations; that of m_{VT} and ω. Both m_{VT} and ω are calculated directly from the input data.

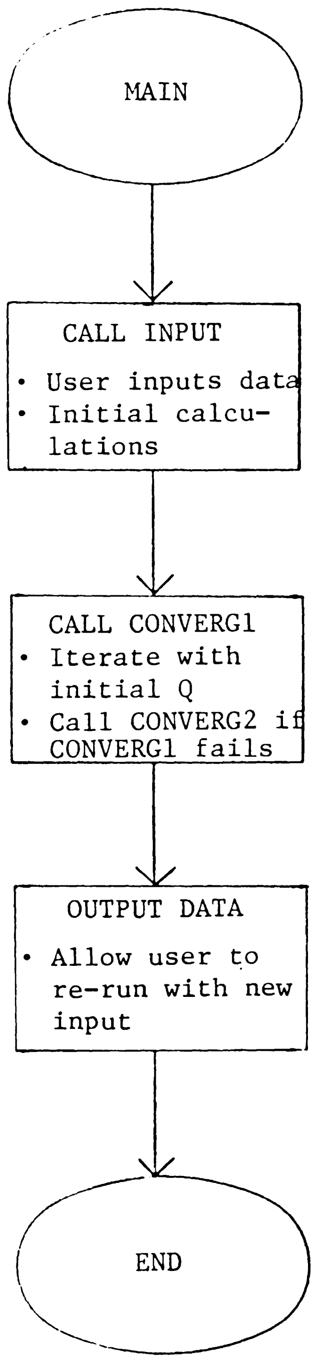
After the following items are displayed as output to the user:

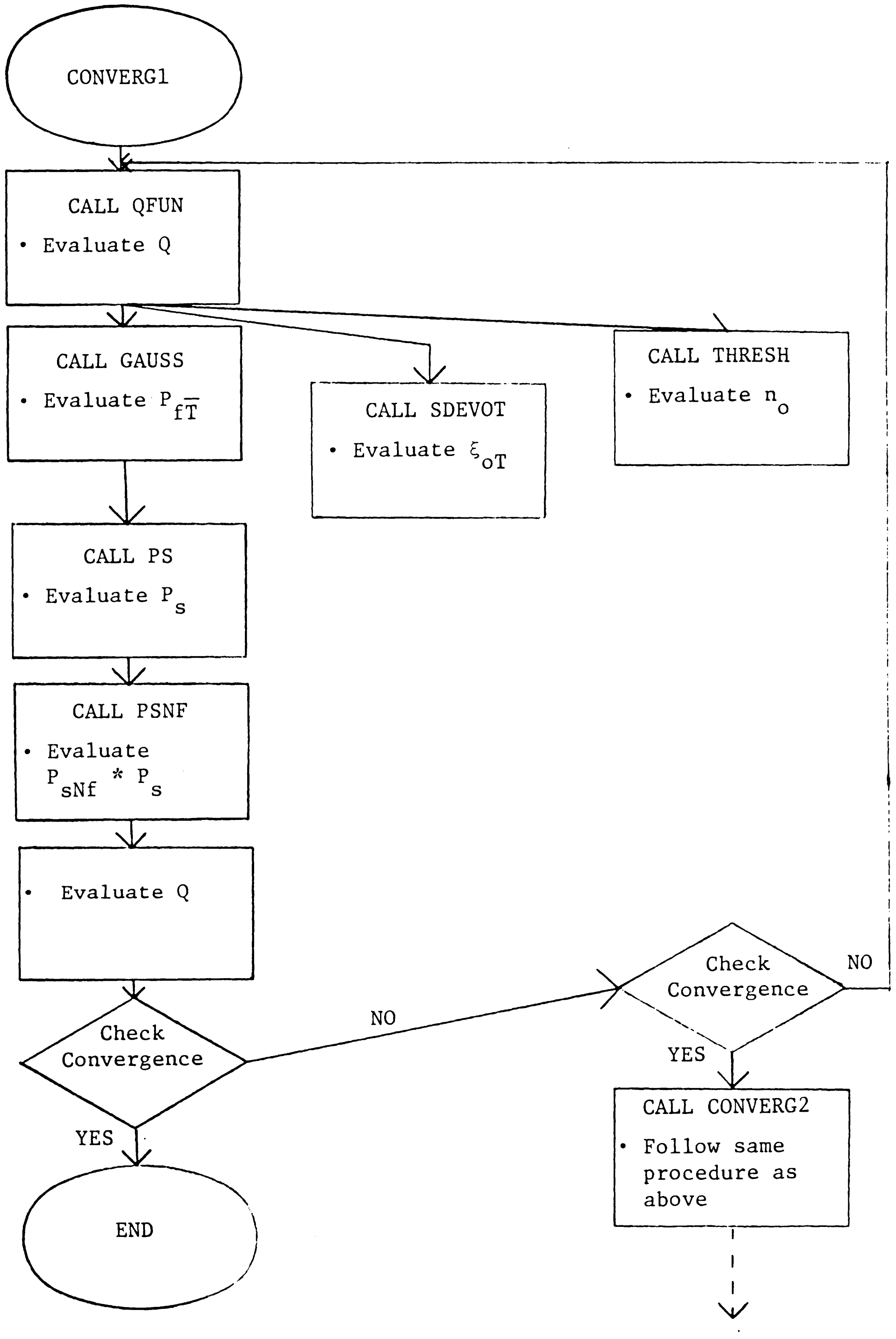
- number of exposures needed - Q
- exposure time - τ_{exp}
- total stare time - τ_{stare}
- threshold level - n_o ,

the user then has the opportunity to make any changes in the input data and run the program again.

The charts on the following pages show the transfer of control throughout the program. They briefly show the order in which the program flows and therefore all the subroutines are not present. A brief description of all the routines in the program follows.

Flow of Control




CONVERG1
CALL QFUN
• Evaluate Q
CALL GAUSS
• Evaluate $P_{f\overline{T}}$
CALL SDEVOT
• Evaluate ξ_{oT}
CALL THRESH
• Evaluate n_o
CALL PS
• Evaluate P_s
CALL PSNF
• Evaluate $P_{sNf} * P_s$
• Evaluate Q
Check Convergence
NO
YES
END
Check Convergence
NO
YES
CALL CONVERG2
• Follow same procedure as above

Description of Routines

Subroutines

INPUT	Prompts user for needed input data and performs preliminary calculations
CONVERG1	Performs method of iteration (or $x = G(x)$) to achieve convergence of Q
CONVERG2	Performs Newton's method if CONVERG1 fails
SBSIN	Prompts user for orbital data and calculates position and velocity vectors for both the sensor and a non-missile target - previously written
MISSIL	Prompts user for launch and impact data and determines position and velocity vectors for a missile target - previously written
LOWBND	Finds an initial estimate to be used with Newton's method in UPLIMIT
UPLIMIT	Performs Newton's method with an initial estimate from LOWBND to get $\int_0^{xx} P_s(m_{vs})\, dm_{vs}$ sufficiently close to 1.

Functions

TSTARE	Calculates total stare time - τ_{stare}
TARMAG	Calculates target magnitude, if not input - m_{VT}
PFIELDV	Calculates pixel field of view - $\Delta\Omega$
THRESH	Calculates threshold level - n_o

SDEVOT	Calculates the number of standard deviations of n_o below $\overline{n}_T - \xi_{oT}$
SDEV	Calculates the effective standard deviation of electrons due to target plus noise - σ_{eff}
MEANT	Calculates the mean number of electrons due to the target - $\overline{n}_T$
TEXP	Calculates exposure time - τ_{exp}
QFUN	Calculates a new value for Q at every iteration in the convergence subroutines. Calls functions to do other calculations affected by a new value of Q.
QDER	Calculates the derivative of QFUN - Q when Newton's method of convergence is used in CONVERG2.
PS	Calculates the probability of a flip due to star presence - P_s
PSNF	Calculates the probability of a flip due to star presence plus readout noise
DOMEG	Calculates the angular rate of motion of the target through the sensor view - previously written
INVERF	Calculates the inverse error function - previously written
ERF	Calculates the error function - previously written
GAUSS	Performs Gaussian Quadrature integration - previously written

ANALYSIS OF TEST RUNS

In order to test the validity of the output, the program was run numerous times with varying data. The computer used was an IBM PC XT. This XT was equipped with 640K memory, removable hard disk drives, two floppy disk drives, a math coprocessor chip, and the RM-Fortran Compiler (Ryan McFarland's version of Fortran). From one run to the next, all the input parameters were held constant except one. The purpose of this was to examine the effect that each individual parameter had on the output. This form of testing was chosen since the main goal of the program was to allow the user to trade-off various parameters and see their effect on the output.

The input data for the test cases is chosen from a range of realistic values. For example, the probability of false alarm, P_{fa}, should not realistically exceed .5. If it did, it would not be reasonable to expect to adequately detect a target regardless of the other input values. In each of the following cases only one of the input values varies from one run to the next. The resulting output and an analysis as to whether the output is physically reasonable are included with the input data.

Diameter - D

Initial Input Data

D	.2
K	5

m_{VT}	7
PD	.6
ε	.09
σ_r	3
Φ_g	0
X	.1
P_{fa}	.005
N	50,000
Q (initial guess)	7

<u>Orbital Data</u>	<u>Sensor</u>	<u>Target</u>
a) altitude at apogee	500	19,000
b) altitude at perigee	500	19,000
c) inclination	45	0
d) initial true anomaly	0	0
e) longitude of ascending node with respect to Greenwich	0	0

<u>Output</u> (D = .2)

τ_{exp}	0.171 sec
τ_{stare}	0.636 sec
Q	3.708 exposures
n_o	44,317.403 electrons

Output (D = 1.5)

τ_{exp}	0.023 sec
τ_{stare}	0.085 sec
Q	3.708 exposures
n_o	333,519.074 electrons

Increasing the diameter, D, of the sensor should make it easier to detect a target. As a detection becomes easier, it is reasonable to expect that less time is needed for each exposure. The fact that Q remained unchanged is not unreasonable since the same number of exposures but with smaller time lengths still yields a smaller total stare time, τ_{stare}. τ_{stare} is a direct result of τ_{exp} and Q. With an easier detection, the threshold level, n_o, would be expected to increase. A higher threshold level allows fewer false alarms to be detected. The threshold level, therefore, need not be as low with an easier target detection.

Target Magnitude - m_{VT}

Initial Input Data

D	1.5
K	41
m_{VT}	10
PD	.6
ε	.09
σ_r	3
Φ_g	0

X	.1
P_{fa}	.005
N	50,000
Q	7

Orbital Data	Sensor	Target
a)	500	19,000
b)	500	19,000
c)	45	0
d)	0	0
e)	0	0

Output (m_{VT} = 10)

τ_{exp}	0.187 sec
τ_{stare}	1.137 sec
Q	6.070 exposures
n_o	172,850.519 electrons

Output (m_{VT} = 18)

τ_{exp}	0.187 sec
τ_{stare}	2.135 sec
Q	11.396 exposures
n_o	91.682 electrons

If the visual magnitude of the target, m_{VT}, increases, it gets dimmer, harder to detect. As a result, it has fewer detectable electrons. (In relation to visual star magnitudes, 18 is an extremely dim magnitude.) The number of exposures, Q, increased, not unexpectedly, in order for a detection to occur with the given probabilities. As stated, the target with the larger magnitude has far fewer detectable electrons. Thus, in order for the target to be detected, the threshold level must be lowered. The exposure time remains unchanged mathematically, m_{VT} has no effect on τ_{exp}. The total stare time, τ_{stare}, increased as a direct result of the increase in Q.

Probability of Detection - PD

	Initial Input Data
D	1.5
K	41
m_{VT}	10
PD	.85
ε	.09
σ_r	3
Φ_g	0
X	.1
P_{fa}	.005
N	50,000
Q	7

Orbital Data	Sensor	Target
a)	500	19,000
b)	500	19,000
c)	45	0

d)	0	0
e)	0	0

<u>Output</u> (PD = .85)

τ_{exp}	0.187 sec
τ_{stare}	1.137 sec
Q	6.070 exposures
n_o	172,627.320 electrons

<u>Output</u> (PD = .999)

τ_{exp}	0.187 sec
τ_{stare}	1.137 sec
Q	6.070 exposures
n_o	171,938.249 electrons

Increasing the probability of detection, PD, resulted in a decrease in the threshold level but left the other input data unchanged. A higher value of PD implies that the user would like more assurance that the target will be detected. The resulting lower value of n_o, allows for this higher probability. It is not unreasonable that the number of exposures and the exposure time were unaltered. The lower threshold yielded the required probability of detection and no other adjustments were needed.

X

Initial Input Data	
D	1.5
K	41
m_{VT}	10
PD	.85
ε	.9
σ_r	40
Φ_g	90
X	.5
P_{fa}	.005
N	50,000
Q	7

Orbital Data	Sensor	Target
a)	500	19,000
b)	500	19,000
c)	45	0
d)	0	0
e)	0	0

Output (X = .5)	
τ_{exp}	0.187 sec
τ_{stare}	1.450 sec
Q	7.739 exposures
n_o	1,731,652.128 electrons

Output (X = .9)

τ_{exp}	0.187 sec
τ_{stare}	1.908 sec
Q	10.181 exposures
n_o	1,731,506.301 electrons

The value of X is an estimate of the ability to remove stars as false alarm candidates where 1 is the worst possible case. (X is between 0 and 1). An increase in the value of X means that more false alarms will show up in the detection. As a result, the threshold level was lowered. Lowering the threshold level allows more false alarms in the detection. More possible false alarms make it reasonable to expect that the value of Q would increase. The more false alarms with the same probability of a false alarm makes a detection harder. The more difficult a detection is, the more exposures it takes.

Conclusions

The above cases along with many other similar test runs lead to the conclusion that the output is indeed accurate. The output was also checked mathematically, i.e., the equations themselves were checked to see if they justified the increase or decrease of the output data. The results were not included in this paper. They were straight forward and do not require an explanation. An example of how the equations were checked can be seen in the first case in this chapter; the change in the diameter. From equation (44), we see that D is inversely proportional to $\Delta\Omega$. Thus, as D increases, $\Delta\Omega$ decreases.

$\Delta\Omega$ is proportional to τ_{exp} (see equation (6)) and therefore as decreases, τ_{exp} decreases. Hence, from this viewpoint, τ_{exp} is expected to decrease with an increase in D and this is indeed the case as can be seen in the output values of τ_{exp}.

BIBLIOGRAPHY

[1] C. W. Allen, Astrophysical Quantities, Athlone Press, University of London, 1963.

[2] K. R. Lang, Astrophysical Formulae, Springer Verlag, Berlin, 1974, pp. 559-560.

[3] G. A. McCue, J. G. Williams, and J. M. Morford, "Optical Characteristics of Artificial Satellites," Planetary and Space Science, Vol. 19, 1971, pp. 851-868.

[4] Curtis F. Gerald and Patrick O. Wheatley, Applied Numerical Analysis, Addison-Wesley Publishing Co., Reading, MA, 1984, pp. 62-67.

APPENDIX A: Star Numbers

Star Numbers

N_m = number of stars per square degree brighter than magnitude m

Variation of N_m with galactic latitude

log N_m

m	Galactic latitude											Mean 0° to 90°	Mean 0° to 90°
	0°	±5°	±10°	±20°	±30°	±40°	±50°	±60°	±70°	±80°	±90°		
	photographic magnitudes												visual
0·0													$\bar{5}$·7
1·0													$\bar{4}$·4
2·0		$\bar{5}$·18			$\bar{4}$·8				$\bar{4}$·7			$\bar{4}$·96	$\bar{4}$·99
3·0		$\bar{5}$·68			$\bar{3}$·27				$\bar{3}$·0			$\bar{3}$·40	$\bar{3}$·55
4·0	$\bar{2}$·25	$\bar{2}$·17	$\bar{2}$·12	$\bar{3}$·99	$\bar{3}$·86	$\bar{3}$·75	$\bar{3}$·70	$\bar{3}$·66	$\bar{3}$·64	$\bar{3}$·62	$\bar{3}$·60	$\bar{3}$·89	$\bar{2}$·11
5·0	$\bar{2}$·72	$\bar{2}$·64	$\bar{2}$·57	$\bar{2}$·44	$\bar{2}$·32	$\bar{2}$·24	$\bar{2}$·20	$\bar{2}$·17	$\bar{2}$·14	$\bar{2}$·12	$\bar{2}$·11	$\bar{2}$·37	$\bar{2}$·60
6·0	$\bar{1}$·18	$\bar{1}$·10	$\bar{1}$·03	$\bar{2}$·90	$\bar{2}$·78	$\bar{2}$·71	$\bar{2}$·66	$\bar{2}$·63	$\bar{2}$·62	$\bar{2}$·60	$\bar{2}$·58	$\bar{2}$·86	$\bar{1}$·07
7·0	$\bar{1}$·61	$\bar{1}$·54	$\bar{1}$·47	$\bar{1}$·34	$\bar{1}$·23	$\bar{1}$·16	$\bar{1}$·11	$\bar{1}$·08	$\bar{1}$·06	$\bar{1}$·04	$\bar{1}$·03	$\bar{1}$·31	$\bar{1}$·54
8·0	0·05	$\bar{1}$·99	$\bar{1}$·91	$\bar{1}$·78	$\bar{1}$·68	$\bar{1}$·60	$\bar{1}$·55	$\bar{1}$·52	$\bar{1}$·49	$\bar{1}$·47	$\bar{1}$·46	$\bar{1}$·75	0·00
9·0	0·52	0·43	0·36	0·22	0·12	0·04	$\bar{1}$·99	$\bar{1}$·94	$\bar{1}$·91	$\bar{1}$·89	$\bar{1}$·88	0·19	0·45
10·0	0·97	0·88	0·80	0·66	0·54	0·46	0·40	0·35	0·31	0·29	0·27	0·62	0·92
11·0	1·43	1·33	1·23	1·08	0·96	0·87	0·80	0·75	0·70	0·64	0·66	1·05	1·34
12·0	1·88	1·77	1·65	1·50	1·37	1·26	1·19	1·12	1·08	1·04	1·03	1·46	1·77
13·0	2·30	2·19	2·07	1·90	1·76	1·64	1·54	1·47	1·42	1·40	1·39	1·87	2·17
14·0	2·72	2·61	2·48	2·28	2·12	1·98	1·88	1·79	1·75	1·71	1·71	2·26	2·56
15·0	3·12	3·00	2·88	2·65	2·46	2·31	2·20	2·10	2·03	2·00	1·97	2·62	2·95
16·0	3·40	3·41	3·24	3·00	2·77	2·61	2·48	2·38	2·28	2·25	2·24	2·98	3·30
17·0	3·83	3·78	3·60	3·33	3·07	2·89	2·75	2·64	2·55	2·52	2·48	3·33	3·64
18·0	4·20	4·10	3·93	3·63	3·35	3·14	2·99	2·87	2·78	2·75	2·72	3·64	3·96
19·0	4·5	4·4	4·3	3·9	3·6	3·4	3·2	3·1	3·0	2·9	2·9	3·90	4·20
20·0	4·7	4·7	4·6	4·2	3·8	3·6	3·4	3·3	3·2	3·1	3·1	4·17	4·45
21·0	5·0	4·9	4·8	4·5	4·0	3·7	3·6	3·4	3·3	3·3	3·2	4·4	

APPENDIX B: Derivation of Flux

Let $S(\lambda)$ be the energy flux per unit wavelength (erg cm^{-2} sec^{-1} $\mathring{\text{A}}^{-1}$). Then, for an object of visual magnitude m_v, as fixed by the zero point of the magnitude system (see [2]),

$$\log S\ (5500\ \mathring{\text{A}}) = -\ 8.42 - 0.4\ m_v\ .$$

If one converts to metric units and specifies quantum rather than energy flux per unit wavelength (photons m^{-2} sec^{-1} m^{-1}), then

$$\log S\ (0.55 \times 10^{-6}\ \text{m}) = 17.02 - 0.4\ m_v\ .$$

The width (at half height) of the sunlight spectrum is $\Delta\lambda \cong 0.55 \times 10^{-6}$ m. Thus, the quantum flux of the sunlight spectrum (including nearby skirts) is:

$$\begin{aligned} F(m_v) &\cong S\ (0.55 \times 10^{-6}\ \text{m})\ \Delta\lambda \\ &= 5.76 \times 10^{-10}\ e^{-0.92\ m_v}\ (\text{photons/m}^{-2}\ \text{sec}^{-1})\ . \end{aligned}$$

APPENDIX C: Definition of Variables

DEFINITION OF VARIABLES

D	- Diameter of sensor (m)
K	- Number of diffraction limited pixel diameters that make up an actual pixel
m_{VT}	- Visual magnitude of target
γ	- Sun angle subtended from sensor to target to sun (deg)
σ_R	- Reflectivity - area product (m)
R	- Distance from sensor to target (m)
PD	- Probability of detection for a target in the field of view
ε	- Combined efficiency of optics and detectors in sensor
σ_r	- Standard deviation of the number of electrons due to noise
Φ_g	- Galactic latitude (deg)
X	- A number between 0 and 1 that represents ability to remove stars as false alarm candidates (1 = worst)
P_{fa}	- Probability of a false alarm
N	- Total numbef of pixels
Q	- Number of exposures needed
ω	- Angular rate of the target through the sensor view (rad/sec)
$\overline{n}_T$	- Expected mean of the number of electrons due to the target
n_o	- Threshold level (number of electrons above which anything is recorded)
$\Delta\Omega$	- Pixel field of view (steradians)
Ω	- Total sensor field of view (steradians)
σ_{eff}	- Effective standard deviation of the number of electrons due to a target plus noise

ξ_{oT}	- Number of standard deviations of n_o below $\overline{n}_T$
$P_{f\overline{T}}$	- Probability of a flip when no target is present
τ_{exp}	- Exposure time (sec)
τ_{stare}	- Total stare time (sec)
$\overline{\lambda}$	- Mean wavelength of the visible spectrum (m)
P_s	- Probability of a flip due to star presence
P_{sNf}	- Probability of a flip due to star presence plus noise

APPENDIX D: Program Code

*** D1 ***

```
      PROGRAM SURPAR
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL PFIELDV,SDEV,MEANT,TEXP,TSTARE,TARMAG
      DOUBLE PRECISION PFIELDV,SDEV,MEANT,TEXP,TSTARE,TARMAG
      DIMENSION SBSSEN(2,6)
      COMMON/ORB/SBSSEN
      COMMON/MAG/MVT
      COMMON/MAGS/SUNA,RCS
      COMMON/CONST1/PI,TPI,WL
      COMMON/CONST2/XTOL,FTOL,NLIM
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      COMMON Q
      COMMON/INPUTS/D,K,PD,EFF,SDR,GL,PFA,FOV
      COMMON/CALC/PFOV,ARATE,SDEFF,EOT,NT,NO,PFT
      COMMON/LINE/SLOPE,YCEPT
      COMMON/PROB/X,N
      LOGICAL CONV,CONVG,FLAG
      CHARACTER *1 ANSW,AN,AA
      INTEGER MM,GL
      DOUBLE PRECISION NT,NO,N,MVT
C
C * * * * * * * * * * * *  VARIABLE DEFINITIONS   * * * * * * * * * * * *
C                                                                        *
C -- VARIABLES --                                                        *
C ARATE:  Angular rate of motion of target through the sensor view      *
```

```
C D:      Aperture of the sensor (m)                               *
C EFF:    Combined efficiency of optics and detectors              *
C EOT:    Number of standard deviations of NO below NT             *
C FOV:    Field of view of sensor (steradians)                     *
C GL:     Galactic latitude (deg)                                  *
C K:      # of diffraction limited pixel diameters that make up an *
C         actual pixel                                             *
C MVT:    Visual magnitude of the target                           *
C N:      Number of pixel detectors                                *
C NO:     Threshold level (# of electrons above which anything is  *
C         recorded)                                                *
C NT:     Expected mean of the number of electrons due to the target *
C PD:     Probability of detection for a target in the field of view *
C PFA:    Probability of a false alarm                             *
C PFOV:   Field of view of a pixel (steradians)                    *
C PFT:    Probability of a flip assuming no target is present in a *
C         pixel                                                    *
C Q:      Number of exposures needed                               *
C R:      Distance from sensor to target (m)                       *
C RCS:    Reflectivity-area product (reflection cross section) (m) *
C SDEFF:  Effective standard deviation of the # of electrons due to a *
C         target plus noise                                        *
C SDR:    Standard deviation of the # of electrons due to noise    *
C SUNA:   Sun angle subtended from sensor to target to sun (rad)   *
C TEXP:   Exposure time (sec)                                      *
C TSTARE: Total stare time (sec)                                   *
```

```
C X:       A number between 0 and 1 that represents ability to remove  *
C          stars as false alarm candidates                              *
C -- FIXED PARAMETERS --                                                *
C CONV,CONVG,FLAG: Tell whether of not convergence has been met         *
C DEGRAD: Radians per degree                                            *
C DMU:     Gravitational constant (m**2/sec**3)                         *
C NLIM:    Upper limit on # of iterations in the convergent routines    *
C PI,TPI: Pi and 2*Pi                                                   *
C RADDEG: Degrees per radian                                            *
C RE:      Radius of the Earth (m)                                      *
C STERSD: Steradians per square degree                                  *
C WE:      Angular rate of the Earth (rad/sec)                          *
C WL:      Mean wavelength of spectrum under consideration (visible)    *
C XMNM:    Meters per nautical mile                                     *
C XTOL,FTOL: Used in convergence subroutines to test closeness to 0     *
C                                                                     . *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C Initialization
C
     PFT=0.0
C
C Call subroutine to ask user for input parameters
C
     CALL INPUT(ARATE,MVT)
C
```

```
      WRITE(6,*) 'Enter an initial guess for the number of exposures',
     &' that will be needed.   '
      READ(*,*) Q
      PRINT *, '   '
C
C  Do initial calculations
C
 101  NT=MEANT(MVT,D,K,EFF,ARATE)
      SDEFF=SDEV(NT,SDR)
      PFOV=PFIELDV(K,D)
      FOV=N*PFOV
      WRITE(6,*) 'THE STARTING VALUE OF Q IS:  ',Q
C
C  Read in slope and y-intercept for appropriate galactic level.  A Least
C  Squares routine was used to fit the log of the Amv values for each
C  galactic latitude to a straight line.
C

      IF (GL.EQ.0) THEN
         I=1
      ELSEIF (IABS(GL).EQ.5) THEN
         I=2
      ELSE
         I=IABS(GL)/10 + 2
      ENDIF
      OPEN (11,STATUS='OLD',FILE='LINES.DAT')
      DO 10 J=1,I-1
         READ(11,*)
 10   CONTINUE
      READ(11,*) SLOPE,YCEPT
      CLOSE (11)
      FLAG=.FALSE.
      CALL LOWBND(XX,FLAG,CONVG)
      WRITE(6,*) 'AFTER LOWBND XX =  ',XX
      IF (.NOT.FLAG) CALL UPLIMIT(XX,CONVG)
      IF (.NOT.CONVG) THEN
         WRITE(6,*) 'Integral did not converge to 1.  Re-run.'
         GO TO 999
      ENDIF
C
C Call subroutine to get a better estimate for the number of exposures
C
      CALL CONVERG1(CONV,XX)
      IF (.NOT. CONV) THEN
         CALL CONVERG2(CONV,XX)
         IF (.NOT. CONV) THEN
            WRITE(6,*) 'No convergence with two different methods.'
            WRITE(6,*) 'Do you want to try another value of your',
     &      ' initial guess of the number of exposures needed (Y/N)?   '
            READ(*,111) ANSW
            IF (ANSW.EQ.'Y') THEN
               WRITE(6,*) 'Enter guess   '
```

```
              READ(*,*) Q
              GO TO 101
           ENDIF
        ENDIF
     ENDIF
C
     IF (CONV) THEN
        TT=TEXP(PFOV,ARATE)
        WRITE(6,100) TT
 100    FORMAT (' ','The exposure time is |  ',F6.3,' sec')
        TS=TSTARE(Q,PFOV,ARATE)
        WRITE(6,200) TS
 200    FORMAT (' ','The stare time is |  ',F7.3,' sec')
        WRITE(6,*) 'SDEFF=  ',SDEFF,' NT=  ',NT,'EOT=  ',EOT
        NO=THRESH(SDEFF,EOT,NT)
        WRITE(6,300) NO
 300    FORMAT (' ','The threshold value is |  ',F12.3,' electrons')
        WRITE(6,400) Q
 400    FORMAT (' ','The number of exposures needed is |  ',F6.3/T3)
        WRITE(6,*) 'PFT=======  ',PFT
C
        WRITE(6,*) 'Do you want to run this program again with any new
    &parameters (Y/N)?  '
        READ(*,111) AN
        IF (AN .EQ. 'Y') THEN
 110       WRITE(6,*) 'Enter the number beside the parameter'
```

```
          WRITE(6,*) 'Aperture of sensor (meters) - 1   '
          WRITE(6,*) '# of diffraction limited pixel diameters - 2   '
          WRITE(6,*) 'Visible magnitude of target - 3   '
          WRITE(6,*) 'Sun angle subtended from sensor-target-sun',
     &    ' (degrees) - 4   '
          WRITE(6,*) 'Reflectivity-area product (meters) - 5   '
          WRITE(6,*) 'Probability of detection of target - 6   '
          WRITE(6,*) 'Combined efficiency of optics and detectors',
     &    ' - 7   '
          WRITE(6,*) 'Standard deviation of noise electrons - 8   '
          WRITE(6,*) 'Galactic latitude (degrees) - 9   '
          WRITE(6,*) 'A number between 0 and 1 that represents your'
          WRITE(6,*) '       ability to remove stars as false alarm'
          WRITE(6,*) '       candidates   (1=worst) - 10   '
          WRITE(6,*) 'The probability of false alarm - 11   '
          WRITE(6,*) 'The  # of pixels - 12  '
          WRITE(6,*) 'A new estimate of the number of exposures',
     &    ' needed - 13   '
          WRITE(6,*) 'Sensor orbital data - 14   '
          WRITE(6,*) 'Target orbital data (missile) - 15   '
          WRITE(6,*) 'Target orbital data (non-missile) - 16   '
          READ(*,*) MM
          IF (MM .LE. 13) THEN
             WRITE(6,*) 'Enter new value   '
          ENDIF
C
```

```
IF (MM .EQ. 1) THEN
   READ(*,*) D
ELSEIF (MM .EQ. 2) THEN
   READ(*,*) K
ELSEIF (MM .EQ. 3) THEN
   READ(*,*) MVT
ELSEIF (MM .EQ. 4) THEN
   READ(*,*) SUNA
   MVT=TARMAG(SUNA,RCS)
ELSEIF (MM .EQ. 5) THEN
   READ(*,*) RCS
   MVT=TARMAG(SUNA,RCS)
ELSEIF (MM .EQ. 6) THEN
   READ(*,*) PD
ELSEIF (MM .EQ. 7) THEN
   READ(*,*) EFF
ELSEIF (MM .EQ. 8) THEN
   READ(*,*) SDR
ELSEIF (MM .EQ. 9) THEN
   READ(*,*) GL
ELSEIF (MM .EQ. 10) THEN
   READ(*,*) X
ELSEIF (MM .EQ. 11) THEN
   READ(*,*) PFA
ELSEIF (MM .EQ. 12) THEN
   READ(*,*) N
```

```
        ELSEIF (MM .EQ. 13) THEN
           READ(*,*) Q
        ELSEIF (MM .EQ. 14) THEN
           CALL SBSIN(SBSSEN,1)
           ARATE=DOMEG(SBSSEN)
        ELSEIF (MM. EQ. 15) THEN
           CALL MISSIL(SBSSEN,2)
           ARATE=DOMEG(SBSSEN)
        ELSE
           CALL SBSIN(SBSSEN,2)
           ARATE=DOMEG(SBSSEN)
        ENDIF
        IF (MM .GE. 14) THEN
           WRITE(6,*) 'If you have previously entered the magnitude'
           WRITE(6,*) ' of the target directly, enter Y, otherwise',
&' enter N   '
           READ(*,111) AA
           IF (AA .EQ. 'N') MVT=TARMAG(SUNA,RCS)
        ENDIF
        WRITE(6,*) 'Do you want to change any other parameters '
        WRITE(6,*) 'before this program is run again (Y/N)?  '
        READ(*,111) AN
        IF (AN .EQ .'Y') GO TO 110
        GO TO 101
     ENDIF
 ENDIF
```

```
999 CONTINUE
111 FORMAT (A1)
    STOP
    END
```

```
***  D2  ***
      SUBROUTINE INPUT(ARATE,MVT)
C
C  This subroutine asks the user for the required input parameters
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL TARMAG,DOMEG
      DOUBLE PRECISION TARMAG,DOMEG
      DIMENSION SBSSEN(2,6)
      COMMON/MAGS/SUNA,RCS
      COMMON/ORB/SBSSEN
      COMMON/INPUTS/D,K,PD,EFF,SDR,GL,PFA,FOV
      COMMON/PROB/X,N
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      CHARACTER ANS*1
      INTEGER IM,GL
      DOUBLE PRECISION N,MVT
      WRITE(6,*) 'Enter the aperture of your sensor (meters)   '
      READ(*,*) D
      WRITE(6,*) 'Enter the # of diffraction limited pixel diameters ',
     & 'that make up an actual pixel    '
      READ(*,*) K
      WRITE(6,*) 'Do you know the visual magnitude of the target (Y/N)'
      READ(*,112) ANS
      IF (ANS.EQ.'Y') THEN
         WRITE(6,*) 'Enter the value    '
```

```
           READ(*,*) MVT
        ELSE
           WRITE(6,*) 'Enter the sun angle subtended from sensor to target
     & to sun (degrees)  '
           READ(*,*) SUNA
           WRITE(6,*) 'Enter the reflectivity-area product (reflection ',
     & 'cross section) (meters)  '
           READ(*,*) RCS
        ENDIF
        WRITE(6,*) 'Enter the probability of detection for a target in ',
     & 'the field of view   '
        READ(*,*) PD
        WRITE(6,*) 'Enter the combined efficiency of optics and',
     & ' detectors   '
        READ(*,*) EFF
        WRITE(6,*) 'Enter the standard deviation of the # of electrons due
     & to noise   '
        READ(*,*) SDR
        WRITE(6,100)
 100    FORMAT(' ','Enter the galactic latitude, choosing one of the ',
     &'following:'/T3,'0,5,-5,10,-10,20,-20,30,-30,40,-40'/T3,'50',
     &',-50,60,-60,70,-70,80,-80,90,-90   ')
        READ(*,*) GL
        WRITE(6,*) 'Enter a number between 0 and 1 that represents your '
        WRITE(6,*) 'ability to remove stars as false alarm candidates '
        WRITE(6,*) '(1 = no discrimination between stars and a target)  '
```

```
      READ(*,*) X
      WRITE(6,*) 'Enter the probability of a false alarm    '
      READ(*,*) PFA
      WRITE(6,*) 'Enter the # of pixels  '
      READ(*,*) N
      WRITE(6,*) 'Input the following orbit data with respect to your',
     &' sensor '
      CALL SBSIN(SBSSEN,1)
      WRITE(6,*) 'If your target is a missile - input 1'
      WRITE(6,*) 'otherwise - input 2    '
      READ(*,*) IM
      IF (IM .EQ. 1) CALL MISSIL(SBSSEN,2)
      IF (IM .EQ. 2) CALL SBSIN(SBSSEN,2)
      ARATE=DOMEG(SBSSEN)
      IF (ANS .EQ. 'N') MVT=TARMAG(SUNA,RCS)
 112  FORMAT (A1)
      RETURN
      END
```

```
***  D3  ***
      DOUBLE PRECISION FUNCTION TSTARE(Q,PFOV,ARATE)
C
C  This function caluculates the stare time given the calculated exposure
C  time and the current value of Q
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL TEXP
      DOUBLE PRECISION TEXP
      T=TEXP(PFOV,ARATE)
      TSTARE=Q*T
      RETURN
      END
```

*** D4 ***

```
      DOUBLE PRECISION FUNCTION TARMAG(SUNA,RCS)
C
C This function calculates the visual magnitude of the target given the
C sun angle (sensor-target-sun), the reflectivity-area product and the
C sensor-target distance
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST1/PI,TPI,WL
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      COMMON/ORB/SBSSEN
      DIMENSION SBSSEN(2,6)
      XDIF=SBSSEN(1,1)-SBSSEN(2,1)
      YDIF=SBSSEN(1,2)-SBSSEN(2,2)
      ZDIF=SBSSEN(1,3)-SBSSEN(2,3)
      R=DSQRT(XDIF*XDIF+YDIF*YDIF+ZDIF*ZDIF)
      SUNA=SUNA/RADDEG
      F=2/(3*PI*PI)*((PI-SUNA)*DCOS(SUNA)+DSIN(SUNA))
      TARMAG=-26.78-2.5*DLOG10((RCS*F)/(R*R))
      RETURN
      END
```

```
***  D5  ***
      DOUBLE PRECISION FUNCTION PFIELDV(K,D)
C
C  This function calculates the field of view of the pixel in steradians
C  given the aperture of the sensor, the mean wavelength of
C  the spectrum and the value of K
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST1/PI,TPI,WL
      PFIELDV=K*K*16.*WL*WL/(PI*D*D)
      RETURN
      END
```

```
*** D6 ***
      DOUBLE PRECISION FUNCTION SDEVOT(PD,Q)
C
C This function calculates the standard deviation of NO below NT given
C the probability of detection of the target and the current value of Q
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL INVERF
      DOUBLE PRECISION INVERF
      SDEVOT=INVERF(2.0*PD**(1.0/Q)-1.0)*SQRT(2.0)
      RETURN
      END
```

*** D7 ***

```
      DOUBLE PRECISION FUNCTION THRESH(SDEFF,EOT,NT)
C
C This function calculates the threshold value given the effective
C standard deviation of electrons due to target plus noise, the mean
C number of electrons due to the target
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION NT
      THRESH=-SDEFF*EOT+NT
      RETURN
      END
```

```
***  D8  ***
      DOUBLE PRECISION FUNCTION SDEV(NT,SDR)
C
C  This function calculates the effective standard deviation of electron
C  due to target plus noise given the mean number of electrons due to th
C  target and the standard deviation of elctrons due to the target alone
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION NT
      SDEV=DSQRT(NT+SDR*SDR)
      RETURN
      END
```

```
***  D9  ***
      DOUBLE PRECISION FUNCTION MEANT(MVT,D,K,EFF,ARATE)
C
C  This function calculates the mean number of electrons due to the
C  target given the diameter of the sensor, the value of K, the wave-
C  length of the spectrum, the angular rate of the sensor, the combined
C  efficiency and the visual magnitude of the target
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST1/PI,TPI,WL
      DOUBLE PRECISION MVT
      E=DEXP(-.92*MVT)
      MEANT=DSQRT(PI)*D*K*WL*EFF/ARATE*(5.76E10)*E
      RETURN
      END
```

```
***  D10  ***
      DOUBLE PRECISION FUNCTION TEXP(PFOV,ARATE)
C
C  This function calculates the exposure time given the field of view
C  and the angular rate of the sensor
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      WRITE(6,*) 'IN TEXP, PFOV= ',PFOV,' ARATE= ',ARATE
      TEXP=DSQRT(PFOV)/ARATE
      RETURN
      END
```

```
***  D11  ***
      SUBROUTINE CONVERG1(CONV,XX)
C
C  This subroutine uses the x=G(x) method (or the method of iteration) of
C  convergence using the initial estimated input  value of Q
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST2/XTOL,FTOL,NLIM
      COMMON Q
      EXTERNAL QFUN
      DOUBLE PRECISION QFUN
      LOGICAL CONV
      CONV=.TRUE.
      J=1
      SAVEQ=Q
      Q=QFUN(Q,XX)
      DEL1=DABS(SAVEQ-Q)
      IF (DEL1 .LE. XTOL) RETURN
      DO 20 J=2,NLIM
         SAVEQ=Q
         WRITE(6,*) 'A NEW Q IS NOW BEING EVALUATED'
         Q=QFUN(Q,XX)
         DEL2=DABS(Q-SAVEQ)
         IF (DEL2 .LE. XTOL) RETURN
         IF (J .EQ. 2) THEN
            IF (DEL1 .LE. DEL2) THEN
```

```
                CONV=.FALSE.
                RETURN
             ENDIF
          ENDIF
 20    CONTINUE
C
C  If NLIM is exceeded CONV returns a value of false
C
       CONV=.FALSE.
       RETURN
       END
```

```
*** D12 ***
      SUBROUTINE CONVERG2(CONV,XX)
C
C  This subroutine uses Newton's method of convergence if the first
C  method fails with the original input value of Q (INIT)
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST2/XTOL,FTOL,NLIM
      COMMON/CALC/PFOV,ARATE,SDEFF,EOT,NT,NO,PFT
      COMMON Q
      DOUBLE PRECISION NT,NO
      LOGICAL CONV
      EXTERNAL QFUN,QDER
      DOUBLE PRECISION QFUN,QDER
      CONV=.TRUE.
      QQ=QFUN(Q,XX)
      QX=QQ-Q
      DO 30 J=1,NLIM
         DELQ=QX/QDER(QQ,XX,PFT)
         Q=Q-DELQ
         QQ=QFUN(Q,XX)
         QX=QQ-Q
         IF (DABS(DELQ) .LE. XTOL) RETURN
         IF (DABS(QX) .LE. FTOL) RETURN
 30   CONTINUE
C
```

```
C  If NLIM is exceeded CONV returns a value of false
C
      CONV=.FALSE.
      RETURN
      END
```

```
*** D13 ***
      DOUBLE PRECISION FUNCTION QFUN(Q,XX)
C
C This function calculates Q as a function of Q to be used in the
C convergent routines
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CALC/PFOV,ARATE,SDEFF,EOT,NT,NO,PFT
      COMMON/INPUTS/D,K,PD,EFF,SDR,GL,PFA,FOV
      COMMON/CONST1/PI,TPI,WL
      COMMON/PROB/X,N
      DOUBLE PRECISION NOT,NT,NO
      EXTERNAL THRESH,SDEVOT,GAUSS,PSNF
      DOUBLE PRECISION THRESH,SDEVOT,GAUSS,PSNF
      INTEGER GL
      DOUBLE PRECISION N
      EOT=SDEVOT(PD,Q)
      NO=THRESH(SDEFF,EOT,NT)
      PFT=GAUSS(50,PSNF,0.,XX)
      QFUN=DLOG(PFA/N)/DLOG(PFT)
      RETURN
      END
```

```
*** D14 ***
      DOUBLE PRECISION FUNCTION QDER(QQ,XX,PFT)
C
C This function evaluates the derivative of QFUN-Q to be used only if
C the CONVERG2 routine is needed
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL GAUSS,PSNF
      DOUBLE PRECISION GAUSS,PSNF
      DPFT=PSNF(XX)
      QDER=-QQ*DPFT/(DLOG(PFT)*PFT)-1.
      RETURN
      END
```

```
*** D15 ***
      SUBROUTINE SBSIN(SBSSEN,L)
C
C  This subroutine requests input for determining the orbits
C  of satellites when the Keplerian Propagator will be used.
C  It has been previously written.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION SBSSEN(2,6),ORB(6),X(3),Y(3),H(3),XMNAN(2)
      COMMON/CONST1/PI,TPI,WL
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      ORB(6)=0.0
      WRITE(6,100)
  100 FORMAT (' '/T3,'Six pieces of data are needed to propagate the ',
     &'satellites.'/T3,'For Circular orbits, only five pieces of data ',
     &'are input.'/T3,'Input 1 if you wish to input apogee and perigee',
     &' altitudes,'/T3,'2 if you wish to input period and eccentricity')
      WRITE(6,*) ' '
      READ(*,*) ITYPE
      IF (ITYPE .EQ. 1) THEN
         WRITE(6,120)
  120    FORMAT (' '//T3,'Input Altitude at ',
     &  'Apogee (NM)'/T3,'Altitude at Perigee (NM)'/T3,'Inclination (deg
     &)')
      ELSE
         WRITE(6,110)
```

```
110     FORMAT (' ',/T3,'Input Period (hours),',
   &    ' Eccentricity, Inclination (degrees)')
     ENDIF
     WRITE(6,130)
130  FORMAT (' '/T3,'Initial true anomaly (deg)'/T3,'Longitude of',
   & ' ascending node with respect to Greenwich (deg)'/T3,
   & 'Input last ascending node before Epoch'/T3)
     READ(*,*) (ORB(I),I=1,5)
     IF (DABS(ORB(2)-ORB(1)) .LT. 1.E-2 .OR. ORB(2).LT. 1.E-5) GO TO 35
     WRITE(6,*) ' Input argument of Perigee (degrees), ',
   & '-180<=argument<180'
     WRITE(6,*) ' '
     READ(*,*) ORB(6)
35   IF (ITYPE .EQ. 2) THEN
        ORB(1)=((ORB(1)*3600./2./PI)**2*DMU)**(1./3.)
     ELSE
        ORB(1)=ORB(1)*XMNM
        ORB(2)=ORB(2)*XMNM
        A=(ORB(1)+ORB(2)+RE*2.)/2.
        ORB(2)=(ORB(1)+RE)/A-1.
        ORB(1)=A
     ENDIF
     DO 10 K=3,6
        ORB(K)=ORB(K)/RADDEG
10   CONTINUE
     B=DCOS(ORB(3))
```

```
      C=DSIN(ORB(3))
      D=DCOS(ORB(4))
      E=DSIN(ORB(4))
      F=DCOS(ORB(5))
      G=DSIN(ORB(5))
      P=DCOS(ORB(6))
      Q=DSIN(ORB(6))
      R=ORB(1)*(1.-ORB(2)**2)/(1.+ORB(2)*D)
      V=DSQRT(2.*(DMU/R-DMU/(2.*ORB(1))))
      ANGMOM=DSQRT((1.-ORB(2)**2)*DMU*ORB(1))
      H(1)=ANGMOM*G*C
      H(2)=-ANGMOM*F*C
      H(3)=B*ANGMOM
      XW=R*D
      YW=R*E
      PX=P*F-Q*G*B
      PY=P*G+Q*F*B
      PZ=Q*C
      QX=-Q*F-P*G*B
      QY=-Q*G+P*F*B
      QZ=P*C
      RX=XW*PX+YW*QX
      RY=XW*PY+YW*QY
C
C  Find Z-component of position vector
C
```

```
      SBSSEN(L,3)=XW*PZ+YW*QZ
C
C  Consider earth rotation between epoch and ascending node.
C
      TRUEA=DABS(ORB(6))
      DO 20 I=1,2
         RPER=ORB(1)*(1.-ORB(2)*ORB(2))/(1.+ORB(2)*DCOS(TRUEA))
         IF (DABS(ORB(2)).LT.0.0001) THEN
            ECCAN=TRUEA
         ELSE
            ECCAN=DACOS((1.-RPER/ORB(1))/ORB(2))
         ENDIF
         IF (TRUEA .GT. PI) ECCAN=TPI-ECCAN
         XMNAN(I)=ECCAN-ORB(2)*DSIN(ECCAN)
         TRUEA=ORB(4)
 20   CONTINUE
      C1=DSQRT(DMU/ORB(1)**3)
      ANG=ORB(4)+ORB(6)
      IF (ANG .GE. TPI) THEN
         DMEAN=XMNAN(1)+XMNAN(2)-TPI
      ELSE
         IF (ORB(6) .LT. 0.) DMEAN=XMNAN(2)-XMNAN(1)
         IF (ORB(6) .GE. 0.) DMEAN=XMNAN(2)+XMNAN(1)
      ENDIF
      DT=DMEAN/C1
      EROT1=DCOS(WE*DT)
      EROT2=DSIN(WE*DT)
C
C  Find X,Y-components of position vector
C
      SBSSEN(L,1)=RX*EROT1+RY*EROT2
      SBSSEN(L,2)=RY*EROT1-RX*EROT2
      HO=H(1)
      H(1)=HO*EROT1+H(2)*EROT2
      H(2)=H(2)*EROT1-HO*EROT2
      GAM=DACOS(ANGMOM/(V*R))
      DO 30 I=1,3
         Y(I)=SBSSEN(L,I)/R
 30   CONTINUE
      X(1)=SBSSEN(L,3)*H(2)-SBSSEN(L,2)*H(3)
      X(2)=SBSSEN(L,1)*H(3)-SBSSEN(L,3)*H(1)
      X(3)=SBSSEN(L,2)*H(1)-SBSSEN(L,1)*H(2)
      DO 40 I=1,3
         X(I)=X(I)/(XMNM**3)
 40   CONTINUE
      XMAG=DSQRT(X(1)**2+X(2)**2+X(3)**2)
      DO 50 I=1,3
         X(I)=X(I)/XMAG
 50   CONTINUE
      VCOSG=V*DCOS(GAM)
      VSING=V*DSIN(GAM)
C
```

```
C  Find velocity vector
C
      SBSSEN(L,4)=VCOSG*X(1)+VSING*Y(1)
      SBSSEN(L,5)=VCOSG*X(2)+VSING*Y(2)
      SBSSEN(L,6)=VCOSG*X(3)+VSING*Y(3)
      RETURN
      END
```

```
*** D16 ***
      SUBROUTINE MISSIL(TRGT,L)
C
C This subroutine determines orbit elements for missile launches.
C Inputs are locations of launch and impact.  The routine
C determines the orbit from two position vectors, using the minimum
C energy trajectory.
C REFERENCE:  Cornelisse, Schoyer, Wakker:  Rocket Propulsion and
C   Spaceflight Dynamics,  CH. 13.
C It has been previously written.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION TRGT(2,6),ORB(6),RL(3),RI(3)
      COMMON/CONST1/PI,TPI,WL
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      WRITE(6,*) ' Note that the missile orbit determination is ',
     & 'Keplerian.'
      WRITE(6,*) ' Use the Keplerian Propagator only.'
      WRITE(6,100)
  100 FORMAT (' '/T3,'Input the Latitude (deg), Longitude (deg) of the',
     & ' launch site,'/T3,'Altitude of launch site (ft).',
     &//T3,'Southern latitudes and/or Western longitudes',
     & ' should be input with a '/T3,'minus sign.'/T3)
      READ(*,*) ORB(1),ORB(2),ORB(3)
      WRITE(6,110)
  110 FORMAT (' '/T3,'Input the Latitude (deg), Longitude (deg) of the',
     & ' impact site,'/T3,'Altitude of impact site (ft)'/T3)
      READ(*,*) ORB(4),ORB(5),ORB(6)
      ORB(1)=ORB(1)/RADDEG
      ORB(2)=ORB(2)/RADDEG
      ORB(3)=ORB(3)*.3048
      ORB(4)=ORB(4)/RADDEG
      ORB(5)=ORB(5)/RADDEG
      ORB(6)=ORB(6)*.3048
      RLCH=RE+ORB(3)
      RL(1)=RLCH*DCOS(ORB(1))*DCOS(ORB(2))
      RL(2)=RLCH*DCOS(ORB(1))*DSIN(ORB(2))
      RL(3)=RLCH*DSIN(ORB(1))
      TOLD=0.
      DELTAT=0.
C
C  Iterate to find exact parameters with rotating earth.
C
 10   ORB(5)=ORB(5)+WE*DELTAT
      RIMP=RE+ORB(6)
      RI(1)=RIMP*DCOS(ORB(4))*DCOS(ORB(5))
      RI(2)=RIMP*DCOS(ORB(4))*DSIN(ORB(5))
      RI(3)=RIMP*DSIN(ORB(4))
C
C  Length of chord between launch and target sites.
C
```

```
      D=DSQRT((RL(1)-RI(1))**2+(RL(2)-RI(2))**2+(RL(3)-RI(3))**2)
C
C  Lenth of major axis.
C
       S=(D+RIMP+RLCH)/2.
C      S=D+RIMP+RLCH
C
C  Compute minimum energy flight time.
C
      B=1.-D/S
      C=DSQRT(B)
      TF=DSQRT((S/2.)**3/DMU)*(PI-2.*DASIN(C)+2.*C*DSQRT(1.-B))
      DELTAT=TF-TOLD
      IF (DABS(TOLD-TF) .LT. 1.E-1) GO TO 20
      TOLD=TF
      GO TO 10
 20   A=S/2.
      VI=DSQRT(2.*(DMU/RLCH-DMU/S))
      SIG=DACOS((RLCH**2+RIMP**2-D**2)/2./RIMP/RLCH)
      GAMI=(PI-SIG)/4.
      E=TAN(GAMI)
C
C  Following derivation taken from Escobal, Methods of Orbit
C  Determination, p. 197.
C
      ECC1=DACOS((1.-RLCH/A)/E)
      ECC2=2.*PI-DACOS((1.-RIMP/A)/E)
      DELTAE=ECC2-ECC1
      F=1.-A/RLCH*(1.-DCOS(DELTAE))
      G=TF-DSQRT(A**3/DMU)*(DELTAE-DSIN(DELTAE))
      DO 30 K=1,3
         TRGT(L,K)=RL(K)
         TRGT(L,K+3)=(RI(K)-F*RL(K))/G
30    CONTINUE
      RETURN
      END
```

```
*** D17 ***
      DOUBLE PRECISION FUNCTION DOMEG(SBSSEN)
C
C  This function calculates the angular rate of motion
C  It has been previously written.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION SBSSEN(2,6),DR(3),DV(3)
      DO 10 I=1,3
         DR(I)=SBSSEN(2,I)-SBSSEN(1,I)
         DV(I)=SBSSEN(2,I+3)-SBSSEN(1,I+3)
 10   CONTINUE
      DRMAG=DSQRT(DR(1)*DR(1)+DR(2)*DR(2)+DR(3)*DR(3))
      DVMAG=DSQRT(DV(1)*DV(1)+DV(2)*DV(2)+DV(3)*DV(3))
      DRDOTDV=DR(1)*DV(1)+DR(2)*DV(2)+DR(3)*DV(3)
      P=DRDOTDV/DRMAG/DVMAG
      THETA=DASIN(DABS(P))
C
C  Rate of apparent motion of target in focal plane
C
      RAM=DVMAG*DCOS(THETA)
      DOMEG=RAM/DRMAG
      RETURN
      END
```

```
***  D18  ***

      DOUBLE PRECISION FUNCTION INVERF(C)
C
C  This function evaluates the inverse error function.  It is adapted
C  from a previously written function.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL ERF
      DOUBLE PRECISION ERF
      DIMENSION Y(7)
      DATA Y/0.0,.842700793,.995322265,.99997910,.999999984,1.0,1.0/
      IF (C .GE. 1.0) THEN
         INVERF=6.0
      ELSEIF (C .LE. 0.0) THEN
         INVERF=0.0
      ELSE
         DO 10 I=1,7
            IF (Y(I)-C) 10,20,30
 10      CONTINUE
 20      INVERF=FLOAT(I-1)
         GO TO 50
 30      XC=I-2
         DO 40 K=1,20
            A=ERF(XC)
            TEMP=XC+(C-A)*(0.886226925*DEXP(XC**2))
            B=ERF(TEMP)
```

```
          Z=C-B
          XC=TEMP
          IF (Z-1.E-10 .LT. 0.0) THEN
             INVERF=TEMP
             GO TO 50
          ENDIF
40      CONTINUE
      ENDIF
50    RETURN
      END
```

```
***  D19  ***

      DOUBLE PRECISION FUNCTION ERF(W)
C
C  This function evaluates the error function.  It was previously
C  written
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION A(25),B(30)
      DATA A/16443152242714D-13,-9049760497548D-13,643570883797D-13,
     *       196418177368D-13,-1244215694D-13,-9101941905D-13,
     *       -1796219835D-13,139836786D-13,164789417D-13,39009267D-13,
     *       -893145D-13,-3747896D-13,1298818D-13,136773D-13,77107D-13,
     *       46810D-13,11844D-13,-5D-13,-1384D-13,-652D-13,145D-13,
     *       10D-13,24D-13,11D-13,2D-13/
      M=24
      X=DABS(W)
      XERR=1.0
      IF (X .GT. 9.306) THEN
         CERR=1.0-XERR
      ELSEIF (X .GE. 0.010) THEN
         Z=(X-1.0)/(X+1.0)
         DO 10 I=1,30
            B(I)=0.0
 10      CONTINUE
         DO 20 I=1,M
            M1=(M+1)-I
```

```
              B(M1)=2.0*Z*B(M1+1)-B(M1+2)+A(M1+1)
20          CONTINUE
            F=-B(2)+Z*B(1)+0.5*A(1)
            XERR=1.0-(1.0/1.77245385)*(DEXP(-(X**2)))*F
            CERR=1.0-XERR
         ELSE
            XERR=2.0/(3.0*1.77245385)*X*(3.0-X**2)
            CERR=1.0-XERR
         ENDIF
         IF (W .GE. 0.0) THEN
            ERF=XERR
         ELSE
            ERF=CERR
         ENDIF
         RETURN
         END
```

```
*** D20 ***
      SUBROUTINE LOWBND(XX,FLAG,CONVG)
C
C This subroutine evaluates the integral of Ps(Mvs) and returns a value
C of TRUE in FLAG if the integral is sufficiently close to 1 and the
C upper limit of the integral in XX.  Flag remains false if the value
C of the integral is not sufficiently close to 1 and less than 1 and
C XX returns the value of the upper limit
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/PROB/X,N
      EXTERNAL PS,GAUSS
      DOUBLE PRECISION PS,GAUSS
      DOUBLE PRECISION N
      LOGICAL FLAG,CONVG
      HK=0.
      DO 10 K=0,100
         B=DFLOAT(K)
         VAL=GAUSS(50,PS,B,B+1)
         HK=VAL+HK
         IF (DABS(HK-1.).LT.0.001) THEN
            XX=DFLOAT(K)
            FLAG=.TRUE.
            RETURN
         ENDIF
         IF (HK .GT. 1.) THEN
```

```
            XX=DFLOAT(K-1)
            RETURN
         ENDIF
         IF (DABS(VAL).LT.0.1D-15) THEN
            XX=DFLOAT(K-1)
            FLAG=.TRUE.
            CONVG=.TRUE.
            RETURN
         ENDIF
10    CONTINUE
      WRITE(6,*) 'FINAL INTEGRAL VALUE =  ',VAL
      WRITE(6,*) 'K VALUE EXCEEDED  K=  ',K
      XX=K
      RETURN
      END
```

```
***  D21  ***
      DOUBLE PRECISION FUNCTION GAUSS(NARG,F,A,B)
C
C This function uses Gaussian Quadrature to evaluate the integral of F
C from A to B using NARG as the number of values of F to be used.  It
C previsouly written.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/PROB/X,N
      DOUBLE PRECISION N
      DIMENSION U(71),H(71),IV(17),TCEL(16)
      DATA U/.28867513,0.,.38729833,
     & .16999052,.43056816,0.,.26923466,.45308992,
     & .11930959,.33060469,.46623476,0.,.20292258,
     & .37076559,.47455396,.09171732,.26276620,
     & .39833324,.48014493,0.,.16212671,.30668572,
     & .41801555,.48408012,.07443717,.21669769,.33970478,
     & .43253168,.48695326,0.,.13477158,.25954806,
     & .36507600,.44353130,.48911433,.06261670,
     & .18391575,.29365898,.38495134,.45205863,.49078032,
     & 0.,.11522916,.22424638,.32117467,
     & .40078905,.45879920,.49209153,.05402747,.15955618,
     & .25762432,.34364645,.41360066,.46421744,.49314190,
     & 0.,.10059705,.19707567,.28548609,.36220887,.42410329,
     & .46863670,.49399626,.04750625,.14080178,.22900839,
     & .30893812,.37770220,.43281560,.47228751,.49470047/
```

```
 DATA H/.50000000,.44444445,.27777778,
& .32607258,.17392742,.28444444,.23931434,.11846344,
& .23395697,.18038079,.08566225,.20897959,.19091503,
& .13985270,.06474248,.18134189,.15685332,.11119052,
& .05061427,.16511968,.15617354,.13030535,.09032408,
& .04063719,.14776211,.13463336,.10954318,.07472567,
& .03333567,.13646254,.13140227,.11659688,.09314511,
& .06279018,.02783428,.12457352,.11674627,.10158371,
& .08003916,.05346966,.02358767,
& .11627578,.11314159,.10390802,
& .08907299,.06943676,.04606075,.02024200,
& .10763193,.10259923,.09276920,.07860158,.06075929,
& .04007904,.01755973,.10128912,.09921574,.09308050,
& .08313460,.06978534,.05357961,.03518302,.01537662,
& .09472531,.09130171,.08457826,.07479799,.06231449,
& .04757926,.03112676,.01357623/
 DATA IV/0,1,2,4,6,9,12,16,20,25,
& 30,36,42,49,56,64,72/
 DATA ZERO/0./
 I=MIN(16,NARG)
 NN=MAX(2,I)
 M1=IV(NN)
 M2=IV(NN+1)-1
 I=1
 J=M1
 V=U(J)
```

```
241  T=(B-A)*V+(A+B)/2.
     TCEL(I)=F(T)
     IF (I.LT.NN) THEN
        I=I+1
        IF (V.LE.ZERO) THEN
           J=J+1
           V=U(J)
        ELSE
           V=-V
        ENDIF
        GO TO 241
     ELSE
       IF (U(M1).EQ.ZERO) THEN
          S=H(M1)*TCEL(1)
          J=2
       ELSE
          S=H(M1)*(TCEL(1)+TCEL(2))
          J=3
          IF (J.GT.NN) THEN
             GAUSS=(B-A)*S
             RETURN
          ENDIF
       ENDIF
     ENDIF
     I=M1+1
     DO 10 J1=I,M2
```

```
      S=S+H(J1)*(TCEL(J)+TCEL(J+1))
      J=J+2
10    CONTINUE
      GAUSS=(B-A)*S
      RETURN
      END
```

```
*** D22 ***
      SUBROUTINE UPLIMIT(XX,CONV)
C
C  This subroutine is used if LOWBND returns a value of FALSE in FLAG .
C  It uses Newton's Method for convergence in order to get the integral
C  of Ps(Mvs) sufficiently close to 1.
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/LINE/SLOPE,YCEPT
      COMMON/CONST2/XTOL,FTOL,NLIM
      COMMON/PROB/X,N
      EXTERNAL PS,GAUSS
      DOUBLE PRECISION PS,GAUSS
      LOGICAL CONV
      DOUBLE PRECISION N
      CONV=.TRUE.
      VAL=GAUSS(50,PS,0.,XX)
      PSX=VAL-1.
      DO 20 J=1,NLIM
         DELP=PSX/PS(XX)
         XX=XX-DELP
         VAL=GAUSS(50,PS,0.,XX)
         PSX=VAL-1.
         WRITE(6,*) 'THE LAST VALUE AFTER THE LAST ONE OF THESE'
         WRITE(6,*) 'STATEMENTS IS INT PS  ',VAL
         IF (DABS(DELP).LE.XTOL) RETURN
         IF (DABS(PSX).LE.FTOL) RETURN
 20   CONTINUE
C
C  If NLIM is exceeded CONV returns a value of false
C
      CONV=.FALSE.
      RETURN
      END
```

```
***  D23  ***
      DOUBLE PRECISION FUNCTION PS(P)
C
C This function evaluates Ps (the probability of a flip due to star
C presence) at P
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      COMMON/LINE/SLOPE,YCEPT
      COMMON/PROB/X,N
      DOUBLE PRECISION M,N,NN
      DIMENSION YA(20),Y(20)
      M=10.**(SLOPE*P+YCEPT)
      M=M/STERSD
C     WRITE(6,*) '# STARS/PIXEL=  ',M/N
      NN=(N-1.)/N
      YA(1)=M
      Y(1)=YA(1)/N*NN**(M-1.)
      PS=Y(1)
      DO 10 J=2,20
         S=DFLOAT(J)
         YA(J)=(M-(S-1.))/S*YA(J-1)
         Y(J)=YA(J)/N**J*NN**(M-S)
         PS=PS+Y(J)
 10   CONTINUE
      PS=X*PS
```

```
RETURN
END
```

```
*** D24 ***
      DOUBLE PRECISION FUNCTION PSNF(P)
C
C  This function evaluates Psnf (the probability of a flip due to star
C  presence plus readout noise) * Ps at P
C
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      EXTERNAL PS,ERF
      DOUBLE PRECISION PS,ERF
      COMMON/CONST1/PI,TPI,WL
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      COMMON/INPUTS/D,K,PD,EFF,SDR,GL,PFA,FOV
      COMMON/CALC/PFOV,ARATE,SDEFF,EOT,NT,NO,PFT
      COMMON/LINE/SLOPE,YCEPT
      COMMON/MAG/MVT
      DOUBLE PRECISION NT,NO,MVT
      E=DEXP(-.92*(P-MVT))
      XE=NT*E
      UL=(NO-XE)/(DSQRT(XE+SDR*SDR)*SQRT(2.))
C     WRITE(6,*) 'AFTER THIS, IGNORE #STARS/PIXEL'
C     READ(*,*)
      PSNF=PS(P)*0.5*(1.-ERF(UL))
C     WRITE(6,*) 'PSNF=  ',PSNF,'UL=  ',UL
      RETURN
      END
```

```
***  D25  ***
      BLOCK DATA
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      COMMON/CONST1/PI,TPI,WL
      COMMON/CONST2/XTOL,FTOL,NLIM
      COMMON/CONST3/XMNM,RE,DMU,WE,DEGRAD,RADDEG,STERSD
      DATA PI,TPI,WL/3.14159265,6.2831853,0.5D-6/
      DATA XTOL,FTOL,NLIM/.00001,.00001,100/
      DATA XMNM,RE,DMU/1852.,6371000.,3.981D14/
      DATA WE,DEGRAD,RADDEG/7.292115856D-5,1.74532925D-2,57.29577958/
      DATA STERSD/3.04617424D-4/
      END
C
C ************************  End of Program  *****************************
```