Technical Note 1980-19

Simulation Study on Detection and Estimation of Closely Spaced Optical Targets

18 March 1980

Prepared for the Department of the Army under Electronic Systems Division Contract F19628-80-C-0002 by

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

Approved for public release; distribution unlimited.
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LINCOLN LABORATORY

SIMULATION STUDY ON DETECTION AND ESTIMATION
OF CLOSELY SPACED OPTICAL TARGETS

M. J. TSAI
Group 32

TECHNICAL NOTE 1980-19

18 MARCH 1980

Approved for public release; distribution unlimited.

LEXINGTON MASSACHUSETTS
ABSTRACT

In this report the detection and estimation of closely spaced optical targets are studied using simulation. The observed signal which originates from two point targets may exhibit only one apparent peak when they are located within one detector width. The Akaike information criterion and a maximum likelihood estimator are used to detect and estimate such unresolved targets. For target separation between 3/4 and 1 detector width the detection rate is high, the estimator is unbiased and the estimation variance is close to the Cramer-Rao bound. The performance degrades greatly when the separation becomes smaller. This loss in performance is attributed to the increasing interference between the two targets and the difficulty in providing a "good" initial guess for the estimator.
CONTENTS

I. INTRODUCTION 1

II. PROBLEM STATEMENT 3

III. METHODS 10
   3.1 Akaike Information Criterion 10
   3.2 Maximum Likelihood Estimator 13

IV. SIMULATION RESULTS AND DISCUSSIONS 20
   4.1 Detection Performance 20
   4.2 Estimation Performance 29
   4.3 Discussion 36

V. CONCLUSION 38

APPENDIX A: COMPUTATIONAL ASPECTS 39
   A.1 Top-level Flow Charts 39
   A.2 Quasi-Newton Method 42
   A.3 Computations of $s_0(t)$ and $s_0(t)$ 45
   A.4 Choice of Initial Guess 47

APPENDIX B: PROGRAM LISTINGS 50

ACKNOWLEDGMENTS 71

REFERENCES 72
I. INTRODUCTION

In the hierarchy of BMD (Ballistic Missile Defense) systems functions, closely spaced object (CSO) resolution occurs early in the sequence of events and consequently influences the performance of the subsequent functions to different degrees. It is very important to assess the CSO resolution capability of the sensor systems employed in every BMD system study before one can determine the overall BMD system performance. The CSO resolution capability is clearly dependent upon the sensor system and the threat characteristics considered in the BMD system study. Current attention has been focused on a variety of passive optical sensor systems employed in the Layered Defense System against threats at long ranges with high angular density. Several studies have been completed recently in assessing the CSO resolution performance for various optical systems [1-4]. The CSO parameter estimation performance can either be predicted by theoretical lower bounds, say the Cramer-Rao lower bounds [1-4], or be evaluated by the Monte-Carlo simulation of specific algorithms. In the earlier studies [1-4], the estimation accuracy for the intensity and position of the CSO's were presented for various lens apertures and noise models under the assumption that the exact number of targets present is known. The CSO detection performance was not presented in these studies.

It is the purpose of this report to address the following
issues through a simulation study. Is the above-mentioned theoretical lower bound achievable in practice? Can the number of targets present in the CSO cluster be determined with certainty so that the assumption made is true? In the simulation, a maximum likelihood estimator is implemented for the CSO parameter estimation and the Akaike information criterion (AIC) is employed to determine the number of targets. A specific sensor and noise model as well as the detector scanning mode is selected for this study. For other sensors and noise models and detector patterns, a similar CSO detection and estimation algorithm can be implemented very easily. This work is in its initial stages. The findings reported here are interesting but not necessarily complete and conclusive. Further investigations are currently in progress and will be reported in future reports.

The problems concerned in this study are first stated in section 2. The models of signal and noise in a single scanning detector environment are outlined in this section. The methods for detection and estimation are described in section 3. Section 4 presents the Monte-Carlo simulation results of detection and estimation performance. The estimation performance thus obtained is also compared with the theoretical result. Some details of the computational aspects of the algorithm and the program listing are attached in the appendices.
II. PROBLEM STATEMENT

For the purpose of this simulation study, the optical sensor system can be simplified and is represented in Fig. 1. The optical point targets which are in the field-of-view of the sensor will form an image on the focal plane. This image is often referred to as the point spread function (PSF). An optical detector is usually employed to scan the image in a fixed direction. The spatial structure of the optical image is thus converted into a temporal electrical signal. This signal is then subject to amplification, filtering and analog-to-digital conversion before entering the signal processor. Noise sources which can be introduced at various points of the system include the background radiation noise, scanning noise, optical-electrical conversion noise, amplifier noise and quantization noise. For simplicity, it is assumed in this study that these noise sources can be lumped together and represented by an additive white gaussian noise (WGN), \( n(t) \). The observations available to the detection and estimation processor can then be written as

\[
y(t_k) = s_d(t_k) + n(t_k) \quad k=1, 2, \ldots, K
\]  

(1)

where \( s_d(t) \) is the desired signal. Given these observations the processor is then required to perform the following two functions:

1. Determine how many targets are embedded in the
Fig. 1. Optical detection/estimation system.
observed signal (detection problem)

2. Obtain intensities and positions of the targets
   (estimation problem)

It is the purpose of this report to describe an algorithm for
such a processor and to evaluate its performance.

Suppose that there are \(n\) point targets which lie along the
scanning direction of the detector. The desired signal \(s_d(t)\)
is simply given by

\[
s_d(t) = \sum_{i=1}^{n} a_i s_0(t - \tau_i)
\]  

(2)

where \(a_i\) and \(\tau_i\) are the intensity and position of the \(i\)th target;
\(s_0(t)\) is the basic target response generated by a target of unit
intensity lying on the optical axis. The shape of \(s_0(t)\) depends
upon the PSF of the particular aperture and the scanning response
function of the detector. Suppose the aperture is annular with
50% obscuration; the detector response function is uniform and
equal to unity within a rectangular gate and equal to zero
elsewhere. Then \(s_0(t)\) is given by [6],

\[
s_0(t) = \int_{-\beta_{y}/2}^{\beta_{y}/2} \int_{\alpha t - \beta_{x}/2}^{\alpha t + \beta_{x}/2} s_f(x, y) \, dx \, dy
\]  

(3a)

*This assumption has to be made because the cross-scan position
of a target can not be resolved by a single detector.
where

\[ s_{\theta}(x,y) = \left( \frac{3}{\pi} \frac{J_1(\pi(x^2+y^2)^{1/2})}{\sqrt{x^2+y^2}^3} - \frac{4}{3} \frac{J_1[\pi(x^2+y^2)^{1/2}]}{\pi(x^2+y^2)^{3/2}} \right)^2 \]  

(3b)

Here, \( J_1(*) \) is the Bessel function of first kind; \( \alpha \) is the angular scanning rate normalized by the optical diffraction limit \( \lambda/d \); \( \beta_x \) and \( \beta_y \) are the in-scan and cross-scan angular dimensions of the detector normalized by \( \lambda/d \). Without loss of generality, the scanning rate can be assumed equal to 1 and thus the time variable \( t \) is equivalent to the angular variable \( \theta \). These two variables will be used interchangeably throughout this report.

The basic optical pulse, \( s_0(t) \), is depicted in Fig. 2 for a detector with \( \beta_x = 2 \) and \( \beta_y = 6 \). Note that there is a slight overshoot near the center of the pulse. Fig. 3 illustrates several sample waveforms, \( y(t) \), from detectors of identical size for the case of 2 CSO's with various intensities and positions. The variance of the WGN is equal to 1. It can be seen that the 2 targets which are separated by 2.5 \( \lambda/d \) (Fig. 3a) correspond to two-peaks of \( y(t) \). In this case they can be detected and estimated with a matched filter followed by a peak detector. However, in Figs. 3(b) - 3(d), the two targets which are located within one detector width (2\( \lambda/d \)) interfere with each other to such a degree that only one apparent

*The signal \( s_d(t) = a s_0(t) \) has the peak at the center \( (t=0) \) equal to 10 for \( a = 9.08 \).
Fig. 2. Typical pulse shape, Eq. (3).
Fig. 3(a-d). Some examples of the observed signal, \( y(t) \), in the case of two CSO's; the variance of the WGN is equal to 1.
peak is observed in the resulting noisy waveform. A simple peak detector may not be capable of resolving satisfactorily the two targets in these examples. Other more sophisticated detection/estimation schemes are required for this purpose.
III. METHODS

3.1 Akaike Information Criterion

One approach for determining the number of targets imbedded in the observed data is to apply the generalized likelihood ratio (GLR) test. There are some difficulties in applying this method. First, the distribution of the GLR is hard to find so that the behavior of the test may not be known exactly. Second, since the test can only be applied to two classes at one time, multiple application of the test is required for the present problem. Third, the choice of the test threshold is usually very subjective.

Akaike has advocated a new approach, termed the Akaike information criterion (AIC), for determining the order of the correct model (for the problem here, the order is the number of targets present) [5]. This information criterion is based on an extension of the maximum likelihood principle starting from the fundamental notation of entropy in statistical mechanics and the Kullback-Leibler information quantity. The final statistic used to optimally choose the order is defined by

\[ \text{AIC} = (-2) \log \text{(maximum likelihood)} + 2 \times \text{(number of free parameters)} \]  

The correct model is that which minimizes this criterion.
Note that the first term in the definition of the AIC represents a penalty of "poor fit" and the second term characterizes increased unreliability. This second term is essential because the maximum value of the likelihood function with the higher order model (model with more parameters) is usually greater than that of the smaller model and therefore without this term, the model with higher order would be favored. Qualitatively speaking, the AIC provides a mathematical formulation of the principle of parsimony in model building.

The AIC has found many applications in various fields, particularly in the autoregressive model fitting of time-series analysis. Since the theory is general, it is easily applied to the detection problem stated in the last section. Two obvious advantages for using the AIC in this problem can be seen from Eq. (4):

1. The AIC is easy to apply; Eq. (4) is very simple.

2. The AIC combines the detection problem with the estimation problem. For detection (determining the order of the model) the information from estimation (finding the maximum likelihood function) is needed. Once the detection is done, the maximum likelihood estimates of parameters in the correct model are also available without additional effort.

Although the AIC is derived from the ideas of information
theory, there seems to be no particular basis for the penalty factor 2. Some investigators, applying the AIC in their particular problems, reported that this factor should be between 3.5 and 4 [7]. It can be shown that the AIC decision rule is equivalent to the hypothesis testing procedure at an appropriate significance level in the case of two classes. Using different values for the penalty factor is analogous to adjusting the significance level. The effect of this factor will be reported in this study. For this purpose, the AIC can be rewritten as

$$AIC(i) = (-2)\log_e(\text{maximum likelihood}) + \eta(i)$$

(5)

where $\eta$ is the penalty factor and $i$ is the number of targets present.

It is reasonable to assume that $AIC(i)$ is a discrete unimodal function of $i$ for a fixed value of $\eta$. Therefore the detection procedure is simply as follows:

Step 1: Start with $i=0$, compute $AIC = AIC(0)$.
Step 2: Increment $i$ by 1.
Step 3: Compute $AIC(i)$.
Step 4: If $AIC(i)$ is greater than $AIC$, go to Step 5. Otherwise, set $AIC = AIC(i)$ and go back to Step 2.

Two parameters, intensity and position, are associated with each target.
Step 5: Stop; the number of targets present is equal to (i-1).

3.2 Maximum Likelihood Estimator

The maximum likelihood estimator (MLE) is implied in the AIC procedure. This estimator possesses several nice properties. It can be shown that the MLE, under rather general conditions, is asymptotically unbiased and efficient; it yields the same results as the least-square method for the case of additive white gaussian noise.

For the signal model described in section 2, the likelihood function can be written as

\[
\mathcal{L}(\theta, \sigma) = \frac{1}{(2\pi)^{k/2} \sigma^k} \exp \left( - \frac{1}{2\sigma^2} (y - \Omega_n \theta)^T (y - \Omega_n \theta) \right)
\]  

(6)

where

\[
\Omega_n = \begin{pmatrix}
s_0(t_1 - \tau_1) & \cdots & s_0(t_1 - \tau_n) \\
\vdots & & \vdots \\
s_0(t_k - \tau_1) & \cdots & s_0(t_k - \tau_n)
\end{pmatrix}
\]  

(7)

\[
y = (y(t_1), y(t_2), \ldots, y(t_k))^T
\]

\[
x = \begin{pmatrix}
\theta \\
\tau
\end{pmatrix} = (a_1, \ldots, a_n, \tau_1, \ldots, \tau_n)^T = (x_1, \ldots, x_n, x_{n+1}, \ldots, x_{2n})^T
\]
and \( \sigma \), considered as an unknown parameter, is the standard deviation of the WGN. The maximum likelihood estimates of \( x \) and \( \sigma \), denoted by \( \hat{x} = (\hat{x}, \hat{\sigma}) \) and \( \hat{\sigma} \) are the values of \( x \) and \( \sigma \) which maximize \( l(x, \sigma) \). Since the logarithm is a strictly increasing function, the maximization of \( l(x, \sigma) \) and \( \log_e l(x, \sigma) \) are equivalent. Let

\[
J(x, \sigma) = \log_e l(x, \sigma) = -\frac{k}{2} \log_e (2\pi) - \log_e \sigma - \frac{1}{2\sigma^2} (y - Q_n \alpha)^T (y - Q_n \alpha)
\]

(8)

The maximum value of this function is the first term needed to evaluate the AIC.

It is usually not necessary to estimate the unknown \( \sigma \) explicitly and it can be dropped from the expression for \( J \). By taking the derivative of \( J \) with respect to \( \sigma \) and setting it to zero, \( \hat{\sigma} \) is obtained as

\[
\hat{\sigma} = \left( \frac{2}{k} (y - Q_n \alpha)^T (y - Q_n \alpha) \right)^{\frac{1}{2}}
\]

(9)

By replacing \( \sigma \) with \( \hat{\sigma} \), Eq. (8) becomes

\[
J(x) = -\frac{k}{2} \left\{ \log_e (2\pi) + 1 + \log_e \left( \frac{1}{k} (y - Q_n \alpha)^T (y - Q_n \alpha) \right) \right\}
\]

(10)

Using this logarithmic likelihood function with the first two nuisance terms discarded, the AIC given by Eq. (5) can be
written as

\[ \text{AIC}(i) = k \log_2 \bar{C}^2 + 2ni. \]  \hspace{1cm} (11)

It is clear that as far as the maximization is concerned, \( J(x) \) in Eq. (10) can be replaced with

\[ J(x) = -(x - C_n a)^T (x - C_n a). \]  \hspace{1cm} (12)

By setting the gradient of \( J(x) \) with respect to \( a \) equal to zero, the intensity estimate, \( \hat{A} \), can be obtained as

\[ \hat{A} = (C_n^T C_n)^{-1} C_n^T x. \]  \hspace{1cm} (13)

Substituting \( \hat{A} \) in Eq. (12), \( J \) can be rewritten as

\[ J(\hat{x}) = x^T [C_n (C_n^T C_n)^{-1} C_n^T - I] x. \]  \hspace{1cm} (14)

The maximization of the likelihood function can be done with respect to expressions given by either Eq. (12) or Eq. (14). The former involves \( 2n \) parameters and the latter involves only \( n \) parameters but requires matrix computations which usually need more computer time. Experience seems to indicate that it is easier to use Eq. (12).
It should be noted that the J function is nonlinear on the unknown parameters. Its maximization is implemented here by using the Quasi-Newton method [8] which is an iteration procedure starting with an initial guess. This method is appealing because it possesses the quadratic convergence property near the maximum of the criterion function (like the Newton method) and avoids the difficulty involved in computing the inverse of the Hessian matrix at each iteration (unlike the Newton method). At each iteration, the gradient, $\nabla J(x) = \frac{\partial J(x)}{\partial x_i}$, $i=1,\ldots,2n$, is required, which, from Eq. (12), is given by

$$
\frac{\partial J(x)}{\partial x_i} = \sum_{k=1}^{K} \left( y(t_k^i) - s_d(t_k^i) \right) \frac{\partial s_d(t_k^i)}{\partial x_i}.
$$

(15)

Here, from Eq. (2),

$$
\frac{\partial s_d(t)}{\partial x_i} = \begin{cases} 
  s_o(t-t_i) & x_i=a_i \\
  -a_i s_o(t-t_i) & x_i=t_i 
\end{cases}
$$

(16)

and, from Eq. (3),

$$
\hat{s}_o(t) = \alpha \int_{-\beta/2}^{\beta/2} \left[ s_f(at+\beta/2, y) - s_f(at-\beta/2, y) \right] dy.
$$

(17)

The unknown parameters are not entirely free but subject to two types of constraints. The intensity of any target, $a_i$, is physically constrained to be non-negative. This constraint can
be released by substituting \( b_1^2 \) for \( a_1 \). By doing so, nothing is changed except that the parameter set becomes \( x = (b_1, \ldots, b_n, \tau_1, \ldots, \tau_n)^T \) and Eq. (16) is replaced by

\[
\frac{\partial s_d(t)}{\partial x_i} = \begin{cases} 
2b_i s_o(t-\tau_i) & x_i = b_i \\
-b_i^2 \frac{\partial s_o(t-\tau_i)}{\partial t} & x_i = \tau_i 
\end{cases} (18)
\]

The final estimate \( \hat{a}_i \) can be obtained by setting \( \hat{a}_i = \hat{b}_i^2 \).

Suppose now the sequence of observations, \( y \), is obtained in the range from \( \theta_o - \gamma/2 \) to \( \theta_o + \gamma/2 \) where \( \gamma \) is the angular distance covered by the observations and \( \theta_o \) is the central point of the range. In practice, it can be assumed that the admissible range for \( \tau_i \), center of the \( i \)th target response, is also within this range. In other words, only targets with center falling in this range will be identified. This additional range constraint will be incorporated into the Quasi-Newton method.

The choice of initial guess is a crucial step for the Quasi-Newton method. A good initial guess can lead the iteration process to converging to the correct maximum in a relatively small number of iterations. On the other hand, for a bad initial guess, the iteration process might converge to a local maximum, oscillate around the maximum or not converge at all.

One method for generating the initial guess is the pure ran-
dom search. This method consists of computing $J(\bar{r})$ at $N$ random points drawn from a probability distribution uniform over the entire parameter space and selecting the point with the greatest value of $J(\bar{r})$ as the initial guess. If it is assumed that each parameter can vary between 0 and 100% and that the optimal parameter set which correspond to the global maximum of $J(\bar{r})$ is to be located within 10% of each parameter then the probability of locating the optimum in $N$ trials is [9]

$$p = 1 - (1 - 10^{-n})^N.$$  \hspace{1cm} (19)

Conversely the number of trials required to have a 90% probability of locating the optimum is

$$N = 1/\log(\frac{1}{1 - 10^{-n}}) \approx 2.3 \times 10^n.$$ \hspace{1cm} (20)

Obviously, the required number of trials increases rapidly with the number of unknown parameters, $n$. If there exists a single target ($n=1$) the pure random search seems able to yield a "good" initial guess within a reasonable computation time. However for more than one target this method becomes impractical.

A more practical approach is to use the pure random search in conjunction with a priori knowledge. As pointed out earlier,

*For the purpose of selecting an initial guess, Eq. (14) instead of Eq. (12) is used.*
in applying the AIC procedure to determine the number of targets, the statistics $AIC(i)$, $i = 1, 2, \ldots$, are computed in ascending sequence. At each step the likelihood function is maximized iteratively starting with an initial guess. It seems feasible to select the initial guess of the $(i+1)^{th}$ step in such a way that the initial values of the first $i$ parameters are equal to the $i$ estimates from the previous step and the initial value of the remaining parameter is chosen from pure random search. Thus, the initial guess for any model (number of target) is obtained by performing a one-parameter search which is easier numerically.

This approach guarantees that the maximum likelihood function of the $(i+1)^{th}$ step is no less than that at the $i^{th}$ step. However, if, for the model of $i+1$ targets, the first $i$ components of the true optimum in the $i+1$ parameter space are far from the optimal point determined in step $i$, the initial guess obtained in this way is usually not "good" for the $(i+1)^{th}$ step. The actual implementation of this approach is described in the appendices.
IV. SIMULATION RESULTS AND DISCUSSIONS

The performance of detecting the number of targets present and estimating the parameters of these targets is evaluated by Monte-Carlo simulation. The simulation algorithm has been described in the previous section and more details are given in the appendices. In presenting the results, some system parameters must be specified. It is assumed that the signal available to the processor is observed in an angular range of $\pm 6.3 \lambda/d$ from the center of the focal plane. The signal is sampled uniformly in this range at an interval of $0.2 \lambda/d$. The total number of observations is 64. The signal is the product of an annular aperture with 50% obscuration and a scanning photo-detector with in-scan and cross-scan dimensions equal to $2 \lambda/d^* \times 6 \lambda/d$ respectively. The number of Monte-Carlo runs is fixed at 100 for every case discussed in the following.

4.1 Detection Performance

First consider the case where at most one target may exist. One must decide between two hypotheses: $H_0$: no-target and $H_1$: one-target present. For this case it is only necessary to compute $AIC(0)$ and $AIC(1)$, and the decision rule accepts $H_0$ ($H_1$) if $AIC(0)$ ($AIC(1)$) is the smaller of the two. The detection performance

*The in-scan dimension of the detector is approximately equal to the diameter of the blur size (diameter of the first dark ring) of the PSF.*
curve (false alarm rate vs. leakage rate) is shown in Fig. 4.
The false alarm rate \( P_f \) is the probability of accepting \( H_1 \) given that \( H_0 \) is true and the leakage rate \( P_L \) is the probability of accepting \( H_0 \) given that \( H_1 \) is true. The parameter \( d \) in the figure is signal-to-noise ratio related and defined as:

\[
d = \left[ \sum_{k=1}^{K} \frac{a^2}{\sigma^2} \right]^{1/2}.
\]

Here \( \sigma \) is set equal to 1 and \( a \) varies among 0.905, 0.454, 0.227 and 0.114 for the four curves shown. Each point on the performance curve corresponds to one value of the penalty factor \( n \) in the AIC defined by Eq. (5). The value of \( n \) is established by the actual operating point which is determined by the requirements for \( P_L \) and \( P_f \).

Now consider the case where there are two CSO's (\( n=2 \)) separated by \( \Delta \theta \) which is less than or equal to \( 2\lambda/d \) (one detector width). It is always assumed that the two targets are located at \( \tau_1 = -\Delta \theta/2 \) and \( \tau_2 = \Delta \theta/2 \), and \( \sigma \) is equal to 1. Fig. 5 shows the probabilities of identifying correctly, from a given observation, 2 targets, \( P(2) \), less than 2 targets, \( P(<2) \), and more than 2 targets, \( P(>2) \). The observation comes from two equally strong targets with signal-to-noise ratio (SNR) of 10. The penalty factor, \( n \) is chosen as 3 here. The curves shown are drawn by connecting the

*The SNR of a target is defined as the intensity of the target at the center of its pulse shape divided by the RMS of the noise. The interference noise is not included.
Fig. 4. Operation characteristic curve for the case where at most one target may exist.
Fig. 5. Probabilities of identifying 2 targets $P(2)$, less than 2 targets $P(<2)$, and more than 2 targets $P(>2)$, while the separation between the actual targets is varied.
finite simulation points. The probability of correctly identifying 2 targets is as high as 90% for separation, $\Delta \theta$, of $2\lambda/d$ (one detector width). However it is less than 25% for separations smaller than $0.5\lambda/d$ (quarter of the detector width). The error is mainly because the 2 CSO's are identified as a single target. For separation, $\Delta \theta$, between $0.5\lambda/d$ and $\lambda/d$, the correct detection rate is between 25% and 65%, and mis-detection is attributed to both under and over identifying the number of targets. The data at no separation ($\Delta \theta=0$) can be viewed from another angle. At $\Delta \theta=0$, the 2 CSO's are coincident and indistinguishable from a single target. Therefore, the data indicates that a single target with SNR=20 has 95% probability of being identified as a single target and 5% probability as 2 CSO's.

Fig. 6 illustrates the effect of choosing different values for the penalty factor $\eta$. It can be seen that using larger values of $\eta$ can increase the detection rate for separation $\Delta \theta>\lambda/d$, but, at the expense of poorer performance for $\Delta \theta<\lambda/d$. There appears to be no obvious way to select $\eta$ optimally. This does not contradict Akaike's theory because no proof for the optimality of using 2 for $\eta$, as proposed by Akaike, has been given. Although not shown in the figure, it is worthwhile to point out that for $\eta=0$, $P(2)$ is equal to 0% for any separation, $\Delta \theta$. In other words, the number of targets present is never correctly identified as 2 when the penalty term of the AIC statistics vanishes.
Fig. 6. Effect of choosing different values for the penalty factor in Eq. (11).
Fig. 7 shows $P(2)$ as a function of $\Delta \theta$ for several different sets of target intensity. The variance of noise remains equal to 1. It is observed that the correct detection rate is generally higher for greater intensity except for the sudden drop of the top curve at $\Delta \theta=\lambda/d$. This drop in detection rate is due to the fact that in many of the 100 Monte Carlo runs for this case, the initial guess provided by the simulation algorithms causes the Quasi-Newton method to oscillate or to converge to a local maximum for the 2-target model but leads to a final estimation for the 3-target model, which includes 2 targets close to the true ones and a third target of insignificant intensity. Note that an experimentor-supplied "good" initial guess for the 2-target model can bring up the detection rate at this point. As to why the algorithm fails at this particular point, no satisfactory explanation has been found.

In Fig. 5 the true model has 2-target with $a_1=a_2=9.08$ and $\tau_1=-\tau_2=-\Delta \theta/2$. Suppose the initial guess of the Quasi-Newton procedure is supplied by the experimentor instead of the algorithm itself and the experimentor intelligently selects $a_1^0=a_2^0=9.08$, $\tau_1^0=-\tau_2^0=-\Delta \theta/2$ for the 1-target model, $a_1^0=a_2^0=9.08$, $a_3^0=1$, $\tau_1^0=-\tau_2^0=-\Delta \theta/2$, $\tau_3^0=0$ for the 3-target model. The resulting detection performance is shown in Fig. 8. A comparison of Figs. 5 and 8 shows that the "good" initial guess increases the correct
Fig. 7. Correct detection rate as function of target separation for several different sets of target intensities.
Fig. 8. Comparison of detection performance between using a "good" initial guess and using the initial guess provided by the simulation algorithm.
detection rate significantly. For separations as small as \(0.8 \lambda/d\) (two fifths of the detector width) the detection performance is almost perfect. The incorrect detection for even smaller separations is because only one target is recognized. The chance of identifying more than 2 targets is almost nil in the entire range of separations considered.

### 4.2 Estimation Performance

The performance of an estimator is usually evaluated in terms of estimation bias and variance. From the \(N\) simulation runs, the variance of an estimate is computed as

\[
\sigma^2_{x_1} = \frac{1}{N} \sum_{j=1}^{N} (x_1^{(j)} - \overline{x}_1)^2
\]  

(22)

where \(x_1^{(j)}\) is the estimate of the parameter \(x_1\) in the \(j\)th run and

\[
\overline{x}_1 = \frac{1}{N} \sum_{j=1}^{N} x_1^{(j)}
\]  

(23)

is the mean of the estimate. The bias of the estimate is then given by

\[
b_{x_1} = \overline{x}_1 - x_1.
\]  

(24)

This sample variance can be compared with the Cramer-Rao
bound (CRB) which is a lower bound on the variance (or covariance matrix) of an unbiased estimator. In the case of a single target in additive white gaussian noise, the CRB for the estimation of parameter vector $\mathbf{x}$ can be obtained by

$$c(x_i) = (F)_{ii}^{-1}$$ (25)

$$F_{ij} = \frac{1}{\sigma^2} \sum_{k=1}^{k} \frac{\partial s_d(t_k)}{\partial x_i} \frac{\partial s_d(t_k)}{\partial x_j}$$ (26)

where $F$ is the Fisher information matrix, $\sigma^2$ the variance of the noise and $s_d(t)$ the desired signal. A set of discrete observations is assumed. This bound is usually tight when the signal-to-noise ratio is high.

Figs. 9 (a) and (b) compare the square roots of the $c(x_i)$ and $c_{xx_i}$ for the intensity and position respectively, in the case of two targets with SNR's both equal to 10. The angular separation between the two targets is the control variable. The simulation is only run at certain values of target separation. For the data shown in these figures the detection procedure is omitted by assuming that the number of targets present is known exactly in advance. That is to say a 2-target model is always assumed. The initial guess for this model is provided in two different ways for comparison. The first uses the procedure described in section 3.2 and the second relies on an "intelligent"
Fig. 9(a-b). Estimation performance and comparison; (a) intensity estimation, (b) position estimation; one example.
experimentor who selects the initial value of the parameter in the neighborhood of the true one. Note that since \( a_1 = a_2 \) and \( |\tau_1| = |\tau_2| \), the concerned statistics for the first target should be very close to those for the second target and so no distinction between them is made in this figure.

From Fig. 9, it can be seen that when the initial guess is provided by the algorithm automatically, the estimate variance agrees with the CRB very well at target separations of \( 1.5\lambda/d \), \( 2\lambda/d \) and \( 2.5\lambda/d \). Recall that the observed signal exhibits two peaks for separation of \( 2.5\lambda/d \) as shown in Fig. 3(a). The sample variance is significantly larger than the CRB for separations less than \( 1.2\lambda/d \). When the "good" initial guesses are employed, the estimation variance becomes quite close to the CRB except at the point where separation, \( \Delta\theta = 0.5\lambda/d \). This once again demonstrates the importance of the initial guess in an iterative optimization algorithm.

Figs. 10(a) and (b) show the same types of comparison as that of Figs. 9(a) and (b) for two targets with SNR's both equal to 15. Similar behaviors are observed. It should be pointed out the \( \sqrt{C(\hat{a})/a} \) and \( \sqrt{C(\hat{\tau})} \) of this example are smaller than those of the former example by a factor of 1.5, which is exactly equal to the ratio of the two corresponding SNR's.

In the first case, the estimation procedure starts with the one-target model and ends with the two-target model. In the second case, it is only applied to the two-target model.
Fig. 10(a-b). Estimation performance and comparison, example.
Based on the same simulation as used in Fig. 9, the biases and the RMS errors* of the intensity and position estimations are shown in Fig. 11(a) and (b). Here the initial guess is generated by the estimator itself. It can be seen that the estimator is actually biased in the region where the sample variance deviates significantly from the Cramer-Rao bound. Meanwhile, the intensity bias, $b_i$, is relatively small compared to the corresponding sample standard deviation (roughly by an order) over the entire CSO region. However, at very small separation ($<\lambda/d$), the position bias is significant and contributive to total RMS error. If only those runs in which the number of targets is identified as 2 are used in computing the statistics, the bias, sample variance and RMS error can be reduced slightly.

*The mean-square error $e^2$ is the sum of the variance and the squared bias $e^2 = \sigma^2 + b^2$. 

34
Fig. 11(a-b). Estimation bias and RMS error; (a) intensity estimation, (b) position estimation.
4.3 Discussion

In this simulation study, the pulse shape of the response of the photo-detector to a unit-intensity point target is assumed known exactly in advance. What the detection/estimation processor does is to adjust the intensities and positions of a certain number of ideal pulse shapes to best match this "synthesized" signal with the given observation. If the pulse shape is different from the one assumed in this report due to differences in aperture shapes, detector responses, detector configurations etc., the simulation program is still applicable as long as the pulse shape and its derivative can be made available. However, the performance of the detection and estimation process may vary significantly with different pulse shapes.

The algorithm described in this report is designed to process the CSO segments rather than the entire observation. It seems feasible to use this algorithm as the second stage of a two-stage signal processor which first process the original signal to identify the isolated "resolved" targets and to separate them from the potential CSO's. This approach may be quite efficient computationally if the probability of occurrence for CSO's is much lower than that of isolated targets.

One version of the Quasi-Newton method is implemented here for the optimization of the likelihood function. There exist some
other methods for this purpose, which can be divided into two general categories: the random method and the gradient method. Usually, no single method is best for all types of nonlinear optimization problems. No evaluation of different methods on the present CSO problem is undertaken in this report. However it is the author's feeling that selecting a particular optimization method may not be as important as devising an efficient and promising way to provide the initial guess.

A single scanning detector is assumed in this report. Since this detector can not resolve the cross-scan component of a target's position, the targets are assumed to lie along the in-scan direction. To determine the in-scan and cross-scan positions of a target, other detector configurations such as the chevron (a pair of detectors oriented in different directions) should be used.

In this simulation study, an additive white Gaussian noise is assumed. This assumption is valid when the thermal noise is the dominant noise source in the optical sensor system. However this noise model becomes inaccurate in the so-called shot-noise limited case where the noise level is dependent on the signal [3]. Even for this case, the detection/estimation scheme presented in this report can apply except that the likelihood function and its gradient should be reformulated. The mathematics involved is, of course, more complicated but still tractable.
V. CONCLUSION

In this report the detection and estimation problems of closely spaced optical point targets are studied using simulation. An annular aperture of 50% obscuration and a scanning photo-detector with in-scan and cross-scan dimensions equal to $2\lambda/d$ and $6\lambda/d$, respectively, are employed. The observed signal which originates from two point targets exhibits a single apparent peak when they are separated by less than one detector width. To detect and estimate such "unresolved" targets, the Akaike information criterion and a particular maximum likelihood estimator are utilized. The estimator is in fact a nonlinear estimation algorithm.

Several examples have been examined. It is found that for target separation between $3/4$ and $1$ detector width, the correct detection rate is fairly high, the estimator unbiased and the sampled variance close to the Cramer-Rao bound. However, the detection and estimation performances degrade significantly for smaller separations. This is unavoidable because when the two targets get closer they become more indistinguishable from a single target, particularly in the presence of noise. More importantly, the difficulty in providing algorithmically a "good" initial guess for the estimator contributes to the poor performance. Undoubtedly, the performance of the algorithm can be improved if a more intelligent way of choosing the initial guess can be devised.
APPENDIX A: COMPUTATIONAL ASPECTS

Programs have been written in accordance with the principles described in section 3 in order to simulate the detection/estimation processor. The performance of the processor is closely related to the particular implementations adopted. Therefore, before presenting the simulation results, the program actually used has to be specified more carefully. The top-level flow charts and details of some subroutines are given in this the following appendices.

A.1 Top-level Flow Charts

For practical reasons, the simulation is done in two steps by two different main programs. The first one (TABLE) is used to create the standard pulse shapes of \( s_o(t) \) and \( s_0(t) \) in advance for use in the second one (CSOMCS) which is the program performing detection and estimation.

The top-level flow charts of these two programs are depicted in Figs. A1(a) and (b) respectively while their listings are given in the appendix. In these charts the functions of some blocks are implemented by subroutines. The names of these subroutines are put down beside the associated blocks. Some of them will be further explained in the following subsections. Note that the detection statistics are the output of the flow chart in Fig. A1.
Fig. A1(a-b). Top-level flow charts of the simulation programs, CSOMCS (a) and TABLE (b).
Fig. Al(a-b). Continued.
It can be modified easily to yield the estimation statistics.

A.2 Quasi-Newton Method

There are several versions of the so called Quasi-Newton method [8]. One particular version has been written by Dr. R. W. Miller in the Laboratory. This version is modified here to meet the requirements of the present simulation study. The procedures are as follows:

Step 1: Set $k=0$; read in the initial parameter values, $\mathbf{x}^0$, and the number of targets, $n$.

Step 2: Compute $J(\mathbf{x}^0)$ and $\nabla J(\mathbf{x}^0)$ according to Eqs. (12), (15) and (18). Find $H^0$ by inverting the matrix

$$ G^0(i,j) = \frac{1}{d} \left[ \frac{\partial J}{\partial x_j}(x_1^0, \ldots, x_i^0 + d, x_{i+1}^0, \ldots, x_{2n}^0) - \frac{\partial J}{\partial x_j}(\mathbf{x}^0) \right] \quad i, j = 1, 2, \ldots, 2n $$

where $d$ is a fixed perturbation. If $G^0$ is uninvertable, $H^0$ is chosen as a diagonal matrix with

$$ H^0(i,i) = d/||\nabla J(\mathbf{x}^0)|| $$

where $||\cdot||$ is the Euclidean norm.

Step 3: Compute the increment of $\mathbf{x}^k$.

$$ \Delta \mathbf{x}^k = H^k \nabla J(\mathbf{x}^k) $$

and its size,

$$ c = ||\Delta \mathbf{x}^k||. $$
If $c$ is less than a preset step size $\alpha$, go to step 4. Otherwise, multiply $\Delta x^k$ with $\alpha/c$ and set $c = \alpha$.

**Step 4:** Compute the increment of $J(x^k)$,

$$\Delta J(x^k) = \Delta x^k \cdot \nabla J(x^k).$$

If $\Delta J(x^k)$ is positive, go to step 5. Otherwise, replace $\Delta x^k$ with

$$\Delta x^k = \beta \frac{\nabla J(x^k)}{||\nabla J(x^k)||},$$

where $\beta$ is a preset number.

**Step 5:** Update $x$,

$$x^{k+1} = x^k + \Delta x^k$$

**Step 6:** Apply the range constraints, $-\frac{\pi}{2} \leq x_{n+j} \leq \frac{\pi}{2}$, $j=1, \ldots, n$. For any $j$, if $|x_{n+j}^{k+1}| \leq \frac{\pi}{2}$ go to step 7. Otherwise, set

$$x_{n+j}^{k+1} = \begin{cases} \frac{\pi}{2} & x_{n+j}^{k+1} < -\frac{\pi}{2} \\ \frac{\pi}{2} & x_{n+j}^{k+1} > \frac{\pi}{2} \end{cases}$$

and

$$\Delta x_{n+j}^k = x_{n+j}^{k+1} - x_{n+j}^k.$$
Recompute the magnitude of the increment,

\[ c = ||\Delta x^k|| \]

Step 7: Compute \( J(x^{k+1}) \), \( \nabla J(x^{k+1}) \) and

\[ H^{k+1} = H^k - \frac{(\Delta x^k + H^k \nabla x^k) \cdot (\Delta x^k)^T H_k}{(\Delta x^k)^T H_k \nabla x^k} \]

provided the denominator is not equal to zero. Here,

\[ y^k = \nabla J(x^{k+1}) - \nabla J(x^k). \]

Step 8: If \( c \leq c_o \), the iteration converges and \( x^k \) is the desired estimate. Stop. Otherwise proceed. Note that \( c_o \) is the threshold value of the stop criterion.

Step 9: If \( k < k_o \), set \( k = k + 1 \) and go to step 3. Otherwise, proceed to the following step. Here, \( k_o \) is a preset value for the maximum number of iterations.

Step 10: Among \( k_o \) iterations find the one which has the greatest value of \( J(x^k) \). Take the estimate of this iteration as the final estimate. Then stop.

The above procedures are implemented in subroutines GNEW and ITERAT. The necessary constants are preset for the simulation as \( d = .005, \ a = .5, \ b = .15, \ c_o = 10^{-5}, \) and \( k_o = 50. \)
A.3 Computations of \( s_o(t) \) and \( \dot{s}_o(t) \)

The Quasi-Newton procedure requires, at each iteration, the computations of \( J(x) \) and \( VJ(x) \) which, in turn, requires those of \( s_o(t_k-t_i) \) and \( \dot{s}_o(t_k-t_i) \) for \( k=1,2,\ldots,k \) and \( i=1,2,\ldots,n \). It is very time-consuming to directly carry out, for every iteration, the associated single and double integrals. Although the computation time can be reduced significantly by using the high-speed convolution method, it is still quite noticeable. An alternative approach is to compute in advance samples of \( s_o(t) \) and \( \dot{s}_o(t) \) as well as their spline coefficients, and store them in files. At the beginning of the simulation, those data are first retrieved and later on, whenever needed, \( s_o(t_k-t_i) \) and \( \dot{s}_o(t_k-t_i) \) can be computed by simple interpolations which take almost no time. If the number of stored samples is large enough, the interpolation would provide sufficient accuracy.

Since \( s(t) \) and \( \dot{s}_o(t) \) would be only computed once and off-line the computer time required is not critical. They are computed by directly carrying out the double and single integrals of Eqs. (3) and (17) using the Gaussian-Legendre quadrature formula [10, 11]. This formula gives for the single integral,

\[
I = \int_{a}^{b} f(u) \, du \tag{A.1}
\]
the following approximation

\[ I = \frac{b-a}{2} \sum_{i=1}^{n} \left\{ A_i^{(n)} \cdot f\left( \frac{b-a}{2} x_i^{(n)} + \frac{b+a}{2} \right) \right\} \] (A.2)

where the weights \( A_i^{(n)} \) and abscissas \( x_i^{(n)} \) can be found from a standard mathematical table. The formula is exact whenever \( f(u) \) is a polynomial of degree \( \leq 2n-1 \). The double integral of the form

\[ I = \int_{a}^{b} \int_{\phi(v)}^{\psi(u)} f(u,v) \, dudv \] (A.3)

is approximated by

\[ I = \frac{b-a}{2} \sum_{i=1}^{n} \left[ A_i^{(n)} \cdot \frac{\psi(c_i) - \phi(c_i)}{2} \sum_{j=1}^{m} \left[ A_j^{(m)} \cdot \left( \frac{\psi(c_i) - \phi(c_i)}{2} x_j^{(m)} + \frac{\psi(c_i) + \phi(c_i)}{2} \right) \right] \right] \]

with

\[ c_i = \frac{b-a}{2} x_i^{(n)} + \frac{b+a}{2} \]  (A.4)

Here, both \( n \) and \( m \) are chosen equal to 16 for the current application, which is shown experimentally to be sufficient. Two subroutines QMULT2 and QMULT1 have been written for this purpose.

The spline coefficients used to interpolate the set of points from \( s_o(t) \) (or \( \dot{s}_o(t) \)) are computed from the Quasi-Cubic Hermite splines [12]. The cubic spline representing the function between each pair of given points is determined by the coordinates and slopes at the two points. The slope at each point is determined
locally by the point in question and two points on its each side. The resulting curve passes through all the given points. The subroutine IQHSCU which is available in the IMSL package [13] is employed for this computation. The subroutine SSDOT called by the simulation main program performs the necessary interpolations to yield values of \( s_d(t_2) \) (Eq. (2)) and \( \frac{\partial s_d(t_2)}{\partial s_i} \) (Eq. (16)), for \( i=1, 2, \ldots, k \) and \( j=1, 2, \ldots, 2n \) given any \( a \) and \( T \). Using the output of SSDOT the subroutine JFCN computes the desired \( J(x) \) and \( \nabla J(x) \).

A.4 Choice of Initial Guess

The subroutine IC is employed to provide the initial guess for the Quasi-Newton procedure. The associated principal logic has been described in subsection 3.2. In this subsection, the details of the program and the flow chart are given.

At the \((i+1)\)th step of the AIC procedure, the initial guesses of the first \( i \) targets are set equal to the estimates from the \( i \)th step and the initial guess for the \((i+1)\)th target is found through the pure random search. From Eq. (12), it is clear that maximizing \( J(x) \) is equivalent to minimizing the following function

\[
\phi_1 = \sum_{j=1}^{k} \left[ y^{(j)} - \sum_{j=1}^{i+1} a_j s_o(t_2 - \tau_j) \right]^2
\]  

(A.5)

over \( a_j \) and \( \tau_j \), \( j=1, \ldots, i+1 \).
Since $a_j$ and $\tau_j$, $j=1,\ldots,i$, are already fixed, it also becomes equivalent to minimizing

$$\phi_2 = \sum_{l=1}^{k} \left[ y_1(t^*_l) - a_{i+1} s_o(t^*_l - \tau_{i+1}) \right]^2$$  \hspace{1cm} (A.6)

over $a_{i+1}$ and $\tau_{i+1}$, where

$$y_1(t^*_l) = y(t^*_l) - \sum_{j=1}^{i} a_j s_o(t^*_l - \tau_j)$$  \hspace{1cm} (A.7)

is the residue of $y(t)$ after being fit by the fixed i targets. Further assume that $(a_{i+1}, \tau_{i+1})$ lies on the curve imposed by

$$\frac{\partial \phi_2}{\partial a_{i+1}} = 0,$$

i.e.,

$$a_{i+1} = \frac{\sum_{l=1}^{k} s_o(t^*_l - \tau_{i+1}) y_1(t^*_l)}{\sum_{l=1}^{k} s_o^2(t^*_l - \tau_{i+1})}$$ \hspace{1cm} (A.8)

This leaves $\tau_{i+1}$ the only parameter to be searched for the minimization of $\phi_2$. The number of random trials in searching for $\tau_{i+1}$ is selected equal to 50 in the program. The flow chart is shown in Fig. A2.

*Eq. (A.8) is the same as Eq. (13) with $n=1.$

48
Assume \((i+1)\) targets present. Input the given observation, \(y_i\). Input, from the previous step, the estimates \(a_j\) and \(r_j\) \(j=1,\ldots,i\) and the residual sum square \(e_i\). Read in the number of random search, ITRL.

\[
(i+1)=1 \quad \text{No}
\]

Set \(a_j^* = a_j\), \(r_j^* = r_j\) \(j=1,\ldots,i\). Set \(e_0^* = e_1\). Compute \(y_1\) according to Eq. (27).

\[
j=1
\]

Randomly pick a value for \(r_{i+1}\) within the admissible region.

Compute \(a_{i+1}\) according to Eq. 25

Compute \(e_{i+1}\) according to Eq. 26.

\[
-2 \leq e_{i+1} \leq 0
\]

No

Yes

Set \(e_0 = e_0^*\), \(a_{i+1} = a_{i+1}^*\), \(r_{i+1} = r_{i+1}^*\), \(i+1 = i+1\)

\[
j = i+1
\]

No

Yes

\[
j = \text{ITRL}
\]

RETURN

Fig. A2. Flow chart of the initial-guess generating subroutine.
The whole simulation algorithm is composed of two main programs, TABLE and CSOMCS. The important input and output variables as well as the listings of these programs are given in this appendix. As for the detailed logic of the algorithm, readers should refer to the context of the report. The comments incorporated in the program will be also helpful in understanding the program.

**Important Variables**

**TABLE:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>indicator of the type of aperture; 1 for square aperture and 2 for annular aperture</td>
</tr>
<tr>
<td>E</td>
<td>obscuration factor of the annular aperture; $0 &lt; E &lt; 1$</td>
</tr>
<tr>
<td>ANGLE</td>
<td>orientation of the detector, specified by the angle (degree) between the central line of the detector and the cross-scan direction; $0 &lt; \text{ANGLE} &lt; 90$.</td>
</tr>
<tr>
<td>NPOINT</td>
<td>number of points where values of $s_o(t)$ and $s_0(t)$ are computed</td>
</tr>
<tr>
<td>TDELTA</td>
<td>interval between a pair of successive points in units of $\lambda/d$</td>
</tr>
<tr>
<td>TBEGIN</td>
<td>position of the first point, equal to $(\text{NPOINT}-1) \times \text{TDELTA}/2$</td>
</tr>
<tr>
<td>ARG</td>
<td>array containing positions of the NPOINT points</td>
</tr>
<tr>
<td>VAL1</td>
<td>array containing the corresponding NPOINT values of $s_0(t_1)$</td>
</tr>
</tbody>
</table>
VAL2 = array containing the NPOINT values of $\hat{s}_o(t_i)$
SPLIN1 = array of dimension NPOINTx3, containing the spline coefficients of $\hat{s}_o(t_i)$
SPLIN2 = array of dimension NPOINTx3, containing the spline coefficients of $\hat{s}_o(t_i)$
FILEN = name of the file which keeps the necessary data
(FILEN, TBEGIN, TDELTA, NPOINT, ARG, VAL1, VAL2, SPLIN1 and SPLIN2) for use in the simulation program CSOMCS

CSOMCS:

FILEN, TBEGIN, TDELTA, NPOINT, ARG, VAL1, VAL2, SPLIN1, SPLIN2 as defined above for the program TABLE.

The following parameters are specified in order to generate artificial noisy signal.

IX = seed of the random number generator
NLOOP = number of Monte Carlo simulation runs
NTO = number of targets present
NS = number of samples
DT = sampling interval
AMP = array containing intensities of the NTO targets
THETA = array containing positions of the NTO targets

STD = standard deviation of the WGN

RT = position of the first sample, equal to \((NS-1) \times DT/2\)

The following parameters are used in the detection/estimation procedure:

NT = number of targets assumed

N = total number of free parameters, equal to \(2 \times NT\)

X = vector of length \(N\). The intensity and position estimates of the \(i^{th}\) target are kept in \(x(i)\) and \(x(i+NT)\).

NM = preset upper bound of the number of assumed target

ETA = value of the AIC penalty constant

AIC = value of the Akaike information criterion

SSQ = value of \(J(x)\) in Eq. (12)

SSQN = value of SSQ under the assumption that no target present

S = vector of length NS containing values of \(s_d(t)\) taken in the NS points
SDOT = vector of length NSxN containing partial derivatives of \( s_d(t) \);
\[
\frac{\partial s_d(t_1)}{\partial a_1}, \ldots, \frac{\partial s_d(t_1)}{\partial a_{NT}}, \frac{\partial s_d(t_2)}{\partial a_1}, \ldots, \frac{\partial s_d(t_2)}{\partial a_{NT}}, \ldots, \frac{\partial s_d(t_L)}{\partial a_1}, \ldots, \frac{\partial s_d(t_L)}{\partial a_{NT}}, \frac{\partial s_d(t_L)}{\partial \tau_1}, \ldots, \frac{\partial s_d(t_L)}{\partial \tau_{NT}}
\]
are stored sequentially in the locations beginning at SDOT((\(L-1\)) *N+1)

XBAR = vector containing mean values of the estimates in all the models assumed

XVAR = vector containing variances of the estimates

ICOUNT = array whose \( i^{th} \) element containing the probability of identifying \( i \) targets

EPS = smallest increment used in convergence test (for QNEW subroutine)

LIMIT = allowed largest number of iterations (for QNEW subroutine)

ITRL = number of the random searches used in subroutine IC for choosing the initial guess

Listings
This program computes and stores the optical pulse shape and its first derivative with respect to angular position for future use. The single integral of Eq. (17) is evaluated by using Gauss-Legendre formula. Subroutine OLO16 provides a table of the 16-point Gauss-Legendre formula.

The double integral of Eq. (3) is computed using the same method.

EXTERNAL FCNF, FUP, FLO, FC1
DOUBLE PRECISION DX(16), DA(16)
DIMENSION X(16), A(16), FILEN(3)
DIMENSION VAL1(1024), VAL2(1024), ARG(1024), SPLIN1(1024, 3), SPLIN2(1024, 3)
COMMON /PSF/ IP, E, /DETOR/, CENTER, ANGLE, BETAX
WRITE(6, 10)
10 FORMAT(/10X, '*** TABLE ***', //)
WRITE(6, 20)
20 FORMAT(1X, 'READ IN FILE NAME')
READ(5, 30) FILEN
30 FORMAT(3A4)
WRITE(6, 40) FILEN
40 FORMAT(1X, 'FILE NAME-', 3A4)
TDELTA = .025
NPOINT = 1024
RANGE = (NPOINT-1) * TDELTA
TBEGIN = -RANGE /2.
BETAX = 7.
BETAY = 6.
BETAYL = BETAY /2.
BETAYH = -BETAYH
WRITE(6, 50)
50 FORMAT(1X, 'READ IN IP: 1 FOR RECTANGULAR, 2 FOR ANNULAR')
READ(5, *) IP
WRITE(6, 60) IP
60 FORMAT(1X, 'IP=', I2)
IF (IP .EQ. 1 ) GO TO 80
WRITE(6, 65)
65 FORMAT(1X, 'READ IN OBSCURATION FACTOR')
READ(5, *) E
WRITE(6, 70) E
70 FORMAT(1X, 'E=', F4.2)
WRITE(6, 90)
80 FORMAT(1X, 'READ IN DETECTOR ANGLE(DEGREE)')
READ(5, *) ANGLE
WRITE(6, 100) ANGLE
100 FORMAT(1X, 'ANGLE=', F6.2)
C
OBTAIN THE GAUSS-LEGENDRE WEIGHTS
C
CALL OLO16(DX, DA, -1, DO, 1, DO)
DO 110 I = 1, 16
  X(I) = DX(I)
110  A(I) = DA(I)
MM = 16
C
COMPUTE THE PULSE SHAPE, VAL1 AND ITS DERIVATIVE VAL2
C
DO 120 I = 1, NPOINT
  CENTER = TBEGIN + TDELTA * (I-1)
ARG(I) = CENTER
VAL1(I) = OMULT2(FCN,PETAYL,PETAYH,FUP,FLD,X,A,MM)
VAL2(I) = OMULT1(FCT,PETAYL,PETAYH,X,A,MM)
120 CONTINUE
C
C COMPUTE THE SPLINE COEFFICIENT OF VAL1, SUB IGHSCU IS IN IMSL.
C
CALL IGHSCU(ARG,VAL1,NPOINT,SPLIN1,NPOINT,IER)
C
C COMPUTE THE SPLINE COEFFICIENT OF VAL2,
C
CALL IGHSCU(ARG,VAL2,NPOINT,SPLIN2,NPOINT,IER)
C
WRITE OUT THE DATA
C
WRITE(6,130) TBEGIN,TDELTA,NPOINT
130 FORMAT(/I4,12E14.7*8/) DO 140 I=1,NPOINT
WRITE(6,140) I,ARG(I),VAL1(I),VAL2(I),(SPLIN1(I,J),J=1,3),(SPLIN2(I,J),J=1,3)
140 FORMAT(I4,6E14.5) CONTINUE
C
STORE DATA UNDER THE GIVEN FILE NAME
C
WRITE(8,160) FILEN,TBEGIN,TDELTA,NPOINT
DO 170 I=1,NPOINT
WRITE(8,180) ARG(I),VAL1(I),VAL2(I),(SPLIN1(I,J),J=1,3),
1 (SPLIN2(I,J),J=1,3)
170 CONTINUE
160 FORMAT(3A4,2E14.7,16) 180 FORMAT(9E14.7) STOP
END
C
FUNCTION OMULT2
C
COMPUTE THE DOUBLE INTEGRAL SS F(X,Y) DXDY, AA,LE,Y,LE,BB,
C
FUNCTION OMULT2(FCN,AA,BB,FU,FL,X,A,MM)
DIMENSION X(I),A(I)
HI = (BB-AA)/2.
G1 = (BB+AA)/2.
Q1 = 0.
DO 4 I=1,MM
UI = HI*X(I) + G1
AI = H1*A(I)
D = FU(UI)
C = FL(UI)
H = (D-C)/2.
G = (D+C)/2.
Q = 0.
DO 2 J=1,MM
VJ = H*X(J) + G
Q = Q + A(J) * FCN(VJ,UI)
2 Q = Q + A(J) * FCN(VJ,UI)
4 Q1 = Q1 + AI*Q
OMULT2 = Q1 RETURN
END
FUNCTION FCN
C COMPUTE F(X, Y) WHICH IS USED IN QMULT2
C
FUNCTION FCN(X, Y)
DOUBLE PRECISION MMBSJI, DR
COMMON /PSF/ IP, E
SCALE = 3.141593
GO TO (10, 20), IP
10 IF (X .EQ. 0.) GO TO 32
   X1 = X * SCALE
   FX = (SIN(X1) / X1)**2
   GO TO 36
32 FX = 1.
36 IF (Y .EQ. 0.) GO TO 42
   Y1 = Y * SCALE
   FY = (SIN(Y1) / Y1)**2
   GO TO 46
42 FY = 1.
46 FCN = FX * FY
   RETURN
20 R = SQRT(X**2 + Y**2) * SCALE
   IF (R .NE. 0.) GO TO 50
   FCN = 1.
   RETURN
50 DR = R
C MMBSJI COMPUTES THE BESSEL FUNCTION OF FIRST KIND, EXISTING IN IMSL.
C
BJ1 = MMBSJI(DR, IER1)
IF(E .EQ. 0.) GO TO 60
   DR = R * E
BJ2 = MMBSJI(DR, IER2)
GO TO 70
60 IER2 = 0
   BJ2 = 0.
70 IF (IER1 .NE. 0 .OR. IER2 .NE. 0) GO TO 80
   TEMP = 2. / ((1. - E**2)**R)
   FCN = ((BJ1 - E*BJ2) * TEMP)**2
   RETURN
80 WRITE(6, 90)
90 FORMAT('SUB MMBSJI ERROR ')
   CALL EXIT
END

FUNCTION FUP
C THE UPPER BOUND USED IN QMULT2
C
FUNCTION FUP(Y)
COMMON /DETO/ CENTER, ANGLE, BETAX
   FUP = CENTER + BETAX / 2.
   IF(ANGLE .EQ. 0.) GO TO 10
   FUP = FUP + Y*TAN(ANGLE*, 0.1745329)
10 RETURN
END
C* FUNCTION FLO
C THE LOWER BOUND USED IN OMULT2
C
FUNCTION FLO(Y)
COMMON /DETOR/ CENTER,ANGLE,BETAX
FLO = CENTER - BETAX/2.
IF(ANGLE.EQ.0.) GO TO 20
FLO=FLO+Y*TAN(ANGLE*.01745329)
20 RETURN
END

C* FUNCTION OMULTI
C COMPUTE THE SINGLE INTEGRAL S F(Y) DY
C
FUNCTION OMULTI(FCT,AA,BB,X,A,MM)
DIMENSION X(1),A(1)
HI=(BB-AA)/2.
G1=(BB+AA)/2.
Q1=0.
DO 4 I=1,MM
UI=HI*X(I)+G1
AI=HI*A(I)
4 Q1=Q1+AI*FCT(UI)
OMULT1=Q1
RETURN
END

C* FUNCTION FCT
C COMPUTE F(Y) WHICH IS REQUIRED BY OMULT1
C
FUNCTION FCT(Y)
COMMON /DETOR/ CENTER,Y,ANGLE,BETAX
X=CENTER
IF(ANGLE.EQ.0.) GO TO 30
X = X + Y*TAN(ANGLE * .01745329)
30 X1 = X + BETAX/2.
X2 = X - BETAX/2.
FCT = FCN(X1,Y) - FCN(X2,Y)
RETURN
END

C* SUBROUTINE GL016
C PREPARE COEFFICIENTS OF THE 16-POINT GAUSS-LEGENDRE FORMULA
C
SUBROUTINE GL016(XA,CA,CB)
DOUBLE PRECISION C,D,X(1),A(1),XX(8),AA(8)
DATA XX/
& .9894009349916499325961541734D0 ,
& .9445750230732325760779884155D0 ,
& .865631202387617438804670977D0 ,
& .755404408350030338951011948D0 ,
& .61787624402643748446647640D0 ,
& .458016777657227386342349429D0 ,
& .2816035507792589132304605014D0 ,
& .19501250987673744018531933542D-1 /
DATA AA/
& .2715245941175409485178057245D-1,
& .622532393864789286284383699D-1,
& .9515851168249278480992510760D-1,
& .1246289712555338720524762821D0,
& .149595988165767320815017305D0,
& .1691565193950025381893120790D0,
& .182603415044923588667636679D0,
& .1894506104550684962853967232D0 /
 DMC = .5D0* (D-C)
 DPC = .5D0*(D+C)
 DO 2 I=1,8
 NI = 17-I
 X(I) = -DMC*XX(I) + DPC
 X(NI) = DMC*XX(I) + DPC
 A(I) = DMC*AA(I)
 A(NI) = DMC*AA(I)
 RETURN
END
MAIN CSOMCS

DESCRIPTION:

This program is a Monte-Carlo simulation of the intensity and angular location estimation of closely spaced optical targets. A set of artificial observations are first generated for a given configuration (no. of targets, intensities and locations). Using these data and assuming no. of CSO's, a Gauss-Newton algorithm is employed to search for the parameter set which maximums the likelihood function. The means and variances of the estimates are also computed. Akaike information criterion is computed for various models.

INTEGER I_COUNT(5)
REAL*4 AMP(4),THETA(4),X(8),S(64),SDOT(512)
REAL*4 XBAR(20),XVAR(20),AML(100,5),DUM(8),TH(10),FILEN(3)
COMMON /DATA/Y(64),NS,STD /SAMPLE/BTPDT /WORK/SDOT
COMMON /SPLINE/ TBEGIN,TDELTA,NPOINT,ARG(1024),VAL1(1024),VAL2(1024)
 WRITE(6,10)
 WRITE(6,20)
10 FORMAT(/5XP'***** RESULTS FROM CSOMCS *****')
20 FORMAT('CSO MONTE CARLO SIMULATION')
WRITE(6,30)
30 FORMAT(/12XP'FILE NAME? '9A4/) C
READ(6,50) FILEN,TBEGIN,TDELTA,NPOINT
DO 40 I=1,NPOINT
READ(6,60) ARG(I),VAL1(I),VAL2(I),(SPLIN1(I,J),J=1,3),
1 (SPLIN2(I,J),J=1,3)
40 CONTINUE
50 FORMAT(3A4,2E14.7,I6)
60 FORMAT(9E14.7)
WRITE(6,70) FILEN
70 FORMAT('/12X,'FILE NAME!','9A4/) C
WRITE(6,80) TBEGIN,TDELTA,NPOINT
C DO 12 I=1,NPOINT
C WRITE(6,90) ARG(I),VAL1(I),VAL2(I),(SPLIN1(I,J),J=1,3),
C (SPLIN2(I,J),J=1,3)
12 CONTINUE
C WRITE(6,100)
80 FORMAT(1X,'ENTER THE PENALTY CONSTANT OF AIC')
C READ(5,*0) ETA
READ(5,110) ETA
90 FORMAT('2X,F5.2') WRITE(6,120)
140 FORMAT(' ENTER SEED FOR RANMOD NUMBER GENERATOR')
C READ(5,*0) IX
WRITE(6,130) IX
150 FORMAT('3X,19') WRITE(6,140)
160 FORMAT(' ENTER NO. OF MONTE-CARLO LOOPS')
C READ(5,*0) ILOOP
WRITE(6,150) ILOOP
170 FORMAT('3X,14')
WRITE(6,180)
180 FORMAT(2' ENTER DATA TO GENERATE ARTIFICIAL SIGNAL: /
  5X,'NT,NS,DELTA,AMP(NT),THETA(NT),STD')
READ(5*) NT,NS,DELTA,(AMP(I),I=1,NT),(THETA(I),I=1,NT),STD
WRITE(6,190) NT,NS,DELTA,(AMP(I),I=1,NT),(THETA(I),I=1,NT),STD
190 FORMAT(1X,I3,I5,5F8.2)
DT=(NS-1)*DT/2.
NM=NT+1
NM2=NM+1
DO 200 I=1,NM
200 NM2=NM2+1
DO 210 I=1,NM2
XBAR(I)=0.
210 XVAR(I)=0.
NM1=NM+1
DO 220 I=1,NM1
220 ICOUNT(I)=0
EPS=1.E-5
LIMIT=50
ITRL=50
C SIMULATE NOISELESS OBSERVATIONS
C CALL SSSDOT(NT,AMP,THETA,NS,0,SSDOT)
WRITE(6,230)
230 FORMAT(2' ESTIMATION STARTS: '/)
DO 400 NL=1,NLOOP
WRITE(6,240) NL,IX
240 FORMAT(1X,'NLOOP= ',14,5X,'IX=I',I10)
C SIMULATE ARTIFICIAL NOISY DATA
C DO 250 I=1,NS
SMEAN = S(I)
CALL GAUS((IX,STD,SMEAN,Y(I)))
250 CONTINUE
C COMPUTE SSQ AND AIC FOR NT=0
C SSQ=0.
DO 260 I=1,NS
260 SSQ=SSQ-Y(I)**2
C VARIANCE OF THE NOISE IS UNKNOWN
C AIC=NS*ALOG(-SSQ/NS)
C VARIANCE OF THE NOISE IS KNOWN
C AIC=SSQ/STD**2
C WRITE(6,270) SSQ,AIC
270 FORMAT(1X,'SSQ= ',E15.6,'AIC= ',E15.6)
C APPLY THE AIC PROCEDURE
C NT=1
280 NT=NT+1
WRITE(6,290) NT
290 FORMAT(I2,'-TARGET MODEL: '/)
CALL IC(X:NT,SSQ,IX,ITRL)
WRITE(6,300)'(IX=I),I=1,N)
300 FORMAT(1X,'IC= ',E15.4)
USE D#2 FOR A SO THAT D IS NOT CONSTRANED TO BE POSITIVE

DO 310 I=1,NT
   X(I)=SORT(X(I))
   CALL ONEWT (X+N,SSQ, EPS,LIMT,ITER,IER)

310 RESTORE A=D#2

DO 320 I=1,NT
   X(I)=X(I)#2
   CALL ORDER(X,N)
   WRITE(6,330) (X(I),I=1,N)
330 FORMAT(2X,'ESTIMATED PARAMETERS!',8E12.4)

VARIANCE IS UNKNOWN
AICNT=NS#ALOG(-SSQ/NS) + ETA*(2#NT)

VARIANCE IS KNOWN
AICNT=-SSQ/STD#2 + ETA*(2#NT)

WRITE(6,340) SSQ,AICNT,ITER
340 FORMAT(2X,'SSQ=',E14.6,5X,'AIC=',E14.6,5X,'ITER=',I4)

IF(AICNT .GE. AIC) GOTO 360
   AIC=AICNT

DO 350 I=1,N
   DUM(I)=X(I)
   NT=NT+1
   IF(NT .LT. NM) GO TO 360
   GO TO 290

UPDATE THE COUNTER
ICOUNT(NT) = ICOUNT(NT) +1
IF(NT .EQ. 1) GO TO 400

KEEP DATA FOR COMPUTING SAMPLE MEAN AND VARIANCE OF THE
ESTIMATED PARAMETERS
K=0
DO 370 I=2,NT
   K=K+(I-2)#2
   N=(NT-1)#2
   DO 380 I=1,N
      XBAR(K+I)=XBAR(K+I)+DUM(I)
   380 XVAR(K+I)=XVAR(K+I)+DUM(I)#2

400 CONTINUE

COMPUTE AND OUTPUT SIMULATION STATISTICS
WRITE(6,410)
410 FORMAT(///,'STATISTICS')
   NT=0
   WRITE(6,420)
420 FORMAT(2X,'0-TARGET MODEL!')
   PCT=100.*ICOUNT(1)/FLOAT(NLOOP)
   WRITE(6,440) NT,PCT
   K=0
   DO 490 NT=1,NM
      N=NT#2
      WRITE(6,430) NT
430 FORMAT(/,X,12,'-TARGET MODEL!')

61
COUNT=FLOAT(ICON(NT+1))
COUNT=COUNT/NLOOP*100.
WRITE(6,440) NT,PCT
440 FORMAT('NT=',I6,' PCT=',F6.2)
IF(PCT.EQ.0.) GO TO 480
DO 450 I=1,N
I1=K+I
XBAR(I1)=XBAR(I1)/COUNT
XVAR(I1)=(XVAR(I1)-COUNT*XBAR(I1)**2)/COUNT
450 XVAR(I1)=SQRT(XVAR(I1))
WRITE(6,460) XBAR(K+I),I=1,N
460 FORMAT(3X,'MEAN:',8(E12.4))
WRITE(6,470) XVAR(K+I),I=1,N
470 FORMAT(3X,'STD:',6(E12.4))
480 K=K+N
490 CONTINUE
CALL EXIT
END

SUBROUTINE QNEW

SUBROUTINE QNEWT(X,N,XJ,eps,limit,icont,ier)

THIS SUBROUTINE IMPLEMENTS QUASI-NEWTON METHOD TO FIND A MAXIMUM OF
A FUNCTION.

X: INITIAL GUES INPUT LOCATION OF MAXIMUM OUTPUT
N: DIMENSION OF X
XJ: VALUE OF MAXIMUM OUTPUT
EPS: SMALLEST INCREMENT USED IN CONVERGENCE TEST
LIMIT: MAXIMUM NUMBER OF ITERATIONS
CONT: NUMBER OF ITERATIONS
IER: ERROR CODE
REQUIRES A SUBROUTINE JFCN(X,N,XJ,DJ) WHERE
X: ANY POINT IN THE PARAMETER SPACE
N: DIMENSION OF X AND DJ
XJ: VALUE OF THE FUNCTION AT X OUTPUT
DJ: GRADIENT EVALUATED AT X OUTPUT

DIMENSION X(1),DJ(1),X1(1),FJ(1)
DIMENSION B(1),S(1),DB(1),IOUT
COMMON/MAIN1/NDIM/INOUT/KIN,KOUT
COMMON/PARAM/S,DS,DSM,IOUT

MAX(N)=8, FJ(LIMIT+1), FX(LIMIT+1)
/MAIN1/ AND /INOUT/ ARE LINKED TO GMINV SUB.

NDIM = 8
KIN = 5
KOUT = 6

FOLLOWING CONSTANTS ARE USED TO CONTROL NUMERICAL PROCEDURES OF QNEW
B: INITIALIZATION PERTURBATION
DB: INCREMENT USED WHEN NOT NEAR A MAXIMUM
DBM: LARGEST INCREMENT ALLOWED
IOUT=PRINTOUT CONTROL

IOUT=1
B=.005
DB=.15
DBM=.5
IER = 0

62
DO 10 I=1,N
10 FX(I)=X(I)
   CALL JFCN(X,N,XJ0,DJ)
   FJ(I)=XJ0
   IF(IOUT.GE.1) WRITE(6,20) XJ0

20 FORMAT(1X,'SSDO0=',E13.6)

C INITIALIZE B
C
DO 40 I=1,N
DO 30 J=1,N
   XI(J) = X(J)
   X1(I) = X(I) + S
   CALL JFCN(X1,N,XJ1,DJ1)
DO 40 J=1,N
40 BI(J,I) = (DJ1(J)-DJ(J))/S
   CALL GMINV(N,N,BI,B+MR+1)
   IF(MR.LT.7) GO TO 60
   DO 50 I=1,N
   DO 50 J=1,N
   50 BI(I,J) = -BI(I,J)
   GO TO 100
C ALTERNATE INITIALIZATION
C
60 DJM = 0.
   DO 70 I=1,N
   DJM = DJM + DJ(I)**2
   DJM = S/SORT(DJM)
   DO 90 I=1,N
   DO 80 J=1,N
   80 BI(J,I) = 0.
   90 BI(I,I) = DJM
C
ITERATE TO SOLUTION
C
100 DO 130 ICONT=1,LIMIT
   CALL ITERAT(X,B,DJ,XJ,DXX,N,IR)
   ICONT=ICONT+1
   FJ(ICONT)=XJ
   DO 110 I=1,N
110 FX(ICONT,I)=X(I)
   IF(IOUT.GE.3) WRITE(6,120) ICONT,XJ,(X(I),I=1,N)
120 FORMAT(1X,I4,2E12.4,4E12.4/(29X,4E12.4))
   IF(IR.NE.0) GO TO 150
   IF(DXX.LT.EPS) GO TO 190
130 CONTINUE
   IF(IOUT.GE.1) WRITE(6,140) LIMIT
   IER = 2
140 FORMAT(2X,'NO CONVERGENCE IN','I4X,'ITERATIONS')
   GO TO 160
150 IER=3
160 XJ=FJ(1)
   KMAX=1
   DO 170 I=2,ICONT
   IF(FJ(I).LE.XJ) GO TO 170
   XJ=FJ(I)
   KMAX=I
170 CONTINUE
   DO 180 I=1,N
180 X(I)=FX(KMAX,I)
RETURN
SUBROUTINE ITERAT
C
C CALLED BY QNEW
C
SUBROUTINE ITERAT(X,B,F,XJ,DXN,N,IR)
DIMENSION X(1),B(8),F(1),DF(8),DX(8),DF(8),DXB(8)
COMMON/PARAM/DS,DSM,IOUT /SAMPLE/BT
IR=0
DXN = 0.
DO 20 I=1,N
DX(I) = 0.
DO 10 J=1,N
10 DX(I) = DX(I) + B(I,J)*F(J)
DO 20 DXN = DXN + DX(I)**2
DXN = SQRT(DXN)
IF(DXN.LE.DSM) GO TO 50
C
C INCREMENT TOO BIG
C
DO 30 I=1,N
30 DX(I) = DX(I)/DSM/DXN
DXN = DSM
IF(IOUT.GE.3) WRITE(6,40)
40 FORMAT(' INCREMENT TOO BIG')
C
C CHECK FOR CORRECT MOTION
C
DO 50 I=1,N
50 XDF = 0.
DO 60 I=1,N
60 XDF = XDF + DX(I)*F(I)
IF(XDF.GT.0.) GO TO 100
C
C MOTION IN WRONG DIRECTION; CHANGE TO THE DIRECTION OF GRADIENT
C
IF(IOUT.GE.3) WRITE(6,70)
70 FORMAT(' NEAR MINIMUM')
FM = 0.
DO 80 I=1,N
80 FM = FM + F(I)**2
DXN = DS
FM = DS/SQRT(FM)
DO 90 I=1,N
90 DX(I) = FM*F(I)
DO 110 I=1,N
100 X(I) = X(I) + DX(I)
C
C CONSTRAINT THE TARGET POSITIONS WITHIN THE RANGE (BT-BT)
C
MT1=N/2+1
IFLAG=0
DO 120 I=MT1,N
ABSX = ABS(X(I))
IF(ABSX.LE.-BT) GO TO 120
IFLAG=1
SIGNX=X(I)/ABSX
TEMP=-SIGNX*BT
DX(I)=DX(I)+TEMP*X(I)
X(I)=TEMP
120 CONTINUE
IF(IFLAG.EQ.0) GO TO 150
DXN=0.
DO 130 I=1,N
130 DXN=DXN+DX(I)**2
IF(DXN.NE.0.) GO TO 140
IR=1
XJ=1.E75
RETURN
140 DXN=SQRT(DXN)
150 CALL JFCN(XN,XJ,DF)
DO 160 I=1,N
160 F(I)=DF(I) + F(I)
DO 170 I=1,N
BF(I)=0.
DO 170 J=1,N
BF(I)=BF(I) + B(I,J)*DF(J)
170 DXB(I)=DXB(I) + DX(J)*B(J,I)
A1=0.
DO 180 I=1,N
180 A1=A1 + DX(I)*BF(I)
IF(A1.NE.0.) GO TO 190
IR=2
RETURN
190 A1=1./A1
DO 200 I=1,N
DO 200 J=1,N
200 B(I,J)=B(I,J) - A1*(BF(I)+DX(I))*DXB(J)
IF(IOUT.GE.4) WRITE(6,210) ((B(J,I),I=1,N),J=1,N)
210 FORMAT(2E20.4)
RETURN
END

C# SUBROUTINE GAUSS
C GENERATE GAUSSIAN DISTRIBUTION RANDOM NOISE
C
SUBROUTINE GAUSS(IX,SYM,AM,V)
A=0.
DO 10 I=1,12
CALL RANDU(IX,IY,Y)
IX=IY
10 A=A+Y
V=(A-6.0)*S+AM
RETURN
END
SUBROUTINE JFCN
COMPUTE J(X) AND ITS GRADIENTS

SUBROUTINE JFCN (X,P,SSQ,DJ)
DIMENSION AMP(4),THETA(4),X(1),DJ(1),S(64),SDOT(512)
COMMON /DATA/ Y(64),NS,STD /WORK/SDOT,B
NT=N/2
DO 10 K=1,NT
AM(K)=X(K)**2
10 THETA(K)=X(K+NT)
CALL SSDOT (NT,AMP,THETA,S,SDOT,NS,1)
SSQ=0.
DO 20 JL=1,NS
BML = Y(JL) -S(JL)
SSQ= SSQ-BML**2
20 CONTINUE
DO 40 IR=1,N
SUM=0.
DO 30 JL=1,NS
NN=NN*(JL-1) + IR
BML = Y(JL) -S(JL)
SUM=SUM+SDOT(NN,BML)
30 CONTINUE

C VARIANCE IS UNKNOWN
C DJ(IR)=-SUM/SSQ*NS
C
VARIANCE IS KNOWN
C DJ(IR)=SUM/STD**2
40 CONTINUE
RETURN
END

SUBROUTINE ORDER
ARRANGE THE ORDER OF THE TARGETS ACCORDING TO THEIR IN-SCAN POSITIONS

SUBROUTINE ORDER(X,N)
DIMENSION X(N)
NT=N/2
IF(NT.LE.1) RETURN
I1=NT+1
I2=NT-1
DO 10 I=I1,I2
J1=I+1
DO 10 J=1,NT
IF(X(J).LE.X(I)) GO TO 10
TEMP=X(I)
X(I)=X(J)
X(J)=TEMP
10 CONTINUE
20 CONTINUE
RETURN
END
SUBROUTINE IC

PROVIDE INITIAL GUESS FOR ONE

SUBROUTINE IC(X, NS, S, SDOT, NS)
DIMENSION X(1), S(64), SDOT(64), Y(64)
COMMON /SAMPLE/BT, /WORK/SDOT, /DATA/Y(64), NS
NT = N/2
AL = BT
AU = BT
IF(NT NE 1) GO TO 20
XJO = 1. E75
DO 10 I = 1, NS
10 Y1(I) = Y(I)
GO TO 50
20 XJO = SSQ
NT1 = NT - 1
CALL SSDOT(NT1, X(I), X(NT), S, SDOT, NS, 0)
DO 30 I = 1, NS
30 Y1(I) = Y(I) - S(I)
DO 40 I = 1, NT1
40 X(N - I) = X(N - I - 1)
IFLAG = 0
DO 90 K = 1, ITRL
CALL RANDU(IY, F)
DO 50 IX = 1, IY
50 THETA = AL + (AU - AL) * FY
CALL SSDOT(1, I, THETA, S, SDOT, NS, 0)
SUM1 = 0.
DO 60 I = 1, NS
60 SUM1 = SUM1 + S(I)**2
SUM2 = 0.
DO 70 I = 1, NS
70 SUM2 = SUM2 + S(I)*Y1(I)
AMP = SUM2/SUM1
IF(AMP LT 1. E-6) AMP = 1. E-6
XJO = 0.
DO 80 I = 1, NS
80 XJ = XJ + (Y1(I) - S(I))**2*AMP
IF(XJ GE XJO) GO TO 90
IFLAG = 1
CONTINUE
XJ = XJ
XNT = AMP
XNT*2 = THETA
XJO = XJ
GO TO 100
90 IF(XJ NE 0.1) GO TO 100
XNT = 1. E-6
XNT*2 = 0.
100 RETURN
END

SUBROUTINE SSDOT

COMPUTE S(T) AND ITS DERIVATIVE AT NS INSTANTS

SUBROUTINE SSDOT(NT, AMP, THETA, S, SDOT, NS, IFLAG)
REAL*4 S(1), SDOT(1), AMP(1), THETA(1)
COMMON /SAMPLE/BT, /DT
COMMON /SPLINE/TBEGIN, TDELTA, NPOINT, ARG(1024), VAL1(1024),
1 VAL2(1024), SPLIN1(1024), SPLIN2(1024)

67
C IF IFLAG=0, SDOT IS NOT COMPUTED
C
C IF(NT.EQ.0) GO TO 20
N = NT*2
DO 10 I=1,NS
S(I) = 0.
II = (I-1) * N
TI=RT+(I-1)*DT
DO 10 K=1,NT
T = TI - THETA(K)
KT = IFIX((T-TBEGIN)/TDELTA) + 1
D = T-ARG(KT)
S1=((SPLINI(KT,3) * D + SPLINI(KT,2))
1  * D + SPLINI(KT,1)) + D + VALI(KT)
S(I) = S(I) + S1*AMP(K)
IF(IFLAG .EQ. 0) GO TO 10
K1=II + K
C
C SDOT(K1) IS THE DERIVATIVE OF S W.R.T. B INSTEAD OF A, A=B**2
C
SDOT(K1) = S1 * 2.*SQR(D*(AMP(K))
K2 = K1+NT
S1 = ((SPLIN2(KT,3)*D + SPLIN2(KT,2))
1  * D + SPLIN2(KT,1)) + D + VAL2(KT)
SDOT(K2) =-AMP(K)*S2
10 CONTINUE
RETURN
20 NM=NT*2*NS
DO 30 I=1,NS
S(I) = 0.
DO 40 I=NM
SDOT(I)=
40 RETURN
END

SUBROUTINE GMINV
C
C MATRIX INVERSION ROUTINE
C
SUBROUTINE GMINV(NRNC,AU,NRMT)
C
MATRIX INVERSION ROUTINE.
C INPUT:
C NR,NC = ROW AND COLUMN DIMEN. OF A
C A = MATRIX TO BE INVERTED
C MT = PRINT CONTROL VARIABLE
C OUTPUT:
C U = GENERALIZED INVERSE OF A
C MR = RANK OF U
C
EXTERNAL DOT
DIMENSION A(1),U(1),S(30)
COMMON/MAIN1/NDIM
COMMON/INOUT/KIN*KOUT
NDIM1 = NDIM+1
TOL = 1.E-14
ADV = 1,E-24
MR = NC
$$\text{NRM1} = \text{NR-1}$$

$$\text{TOL1} = 0,$$

$$\text{JJ} = 1$$

DO 10 J = 1, NC

$$\text{S(J)} = \text{DOT(NR, A(JJ), A(JJ))}$$

IF $$\text{S(J)} > \text{TOL1}$$

10

$$\text{JJ} = \text{JJ} + \text{NDIM}$$

$$\text{TOL1} = \text{ADV} * \text{TOL1}$$

$$\text{ADV} = \text{TOL1}$$

$$\text{JJ} = 1$$

DO 100 J = 1, NC

$$\text{FAC} = \text{S(J)}$$

$$\text{JM1} = \text{J - 1}$$

$$\text{JRM} = \text{JJ + NRM1}$$

$$\text{JCM} = \text{JJ + JM1}$$

DO 20 I = JJ, JCM

20

$$\text{U(I)} = 0.$$

$$\text{U(JCM)} = 1.0$$

IF $$\text{J = EQ. 1}$$ GO TO 54

$$\text{KK} = 1$$

DO 30 K = 1, JM1

IF $$\text{S(K) = EQ. 1.0}$$ GO TO 30

$$\text{TEMP} = -\text{DOT(NR, A(JJ), A(KK))}$$

CALL VADD(K, TEMP, U(JJ), U(KK))

30

$$\text{KK} = \text{KK + NDIM}$$

DO 50 L = 1, 2

$$\text{KK} = 1$$

DO 50 K = 1, JM1

IF $$\text{S(K) = EQ. 0.}$$ GO TO 50

$$\text{TEMP} = -\text{DOT(NR, A(JJ), A(KK))}$$

CALL VADD(NR, TEMP, A(JJ), A(KK))

CALL VADD(K, TEMP, U(JJ), U(KK))

50

$$\text{KK} = \text{KK + NDIM}$$

$$\text{TOL1} = \text{TOL} * \text{FAC + ADV}$$

$$\text{FAC} = \text{DOT(NR, A(JJ), A(JJ))}$$

DO 55 I = JJ, JRM

55

$$\text{A(I)} = 0.$$

$$\text{S(J)} = 0.$$

$$\text{KK} = 1$$

DO 65 K = 1, JM1

IF $$\text{S(K) = EQ. 0.}$$ GO TO 65

$$\text{TEMP} = -\text{DOT(K, U(KK), U(JJ))}$$

CALL VADD(K, TEMP, U(JJ), U(KK))

65

$$\text{KK} = \text{KK + NDIM}$$

$$\text{FAC} = \text{DOT(J, U(JJ), U(JJ))}$$

$$\text{MR} = \text{MR - 1}$$

GO TO 75

70

$$\text{S(J)} = 1.0$$

$$\text{KK} = 1$$

DO 72 K = 1, JM1

IF $$\text{S(K) = EQ. 1.}$$ GO TO 72

$$\text{TEMP} = -\text{DOT(NR, A(JJ), A(KK))}$$

CALL VADD(K, TEMP, U(JJ), U(KK))

72

$$\text{KK} = \text{KK + NDIM}$$

75

$$\text{FAC} = 1./\text{SQRT(FAC)}$$

DO 80 I = JJ, JRM

80

$$\text{A(I)} = \text{A(I)} * \text{FAC}$$

DO 85 I = JJ, JCM

85

$$\text{U(I)} = \text{U(I)} * \text{FAC}$$

J = JJ + NDIM

DO 100 MR = EQ. NR, OR, MR = EQ. NC GO TO 120
```
IF (MT NE 0) WRITE(KOUT,110) NR, NC, MR
110 FORMAT(13,1X,12,8H M! RANK,12)
120 NEND = NC*NDIM
JJ=1
DO 135 J=1,NC
DO 125 I=1,NR
II=I-J
S(I)=0.
125 S(I)=S(I)+A(II+KK)*U(KK)
II=J
DO 130 I=1,NR
U(II)=S(I)
130 II=II+NDIM
135 JJ=JJ+NDIM1
RETURN
END

C
C SUBROUTINE VADD(N,C1,A,B)
C
C INPUT:
C N = ARRAY DIMENSION
C C1 = SCALAR
C A = NX1 VECTOR
C B = NX1 VECTOR
C OUTPUT:
C A = NX1 VECTOR SUM
DIMENSION A(C1),B(1)
DO 1 I=1,N
1 A(I) = A(I) + C1*B(I)
RETURN
END

FUNCTION DOT(NR,A,B)
C
C INPUT:
C NR = ARRAY DIMENSION
C A = NRX1 VECTOR
C B = NR X! VECTOR
DIMENSION A(1),B(1)
DOT = 0.
DO 1 I=1,NR
1 DOT = DOT + A(I)*B(I)
RETURN
END
```
ACKNOWLEDGMENTS

The critical comments and careful reviews of Drs. J. A. Tabaczynski, S. D. Weiner and K. P. Dunn are gratefully acknowledged. The author also wish to thank F. Chen for her assistance in computer programming. In addition, the skillful typing of the manuscript by C. A. Lanza is appreciated.
REFERENCES


**Title:** Simulation Study on Detection and Estimation of Closely Spaced Optical Targets

**Performing Organization Name and Address:**
Lincoln Laboratory, M.I.T.
P.O. Box 73
Lexington, MA 02173

**Prepared By:** Ming-Jyi Teel

**Contract or Grant Number:**
F19628-80-C-0004

**Report Date:** 18 March 1980

**Number of Pages:** 80

**Security Classification:** Unclassified

**Distribution Statement (of this Report):**
Approved for public release; distribution unlimited.

**Abstract:**
In this report the detection and estimation of closely spaced optical targets are studied using simulation. The observed signal which originates from two point targets may exhibit only one apparent peak when they are located within one detector width. The Alkai information criterion and a maximum likelihood estimator are used to detect and estimate such unresolved targets. For target separation between 3/4 and 1 detector width the detection rate at high, the estimator is unbiased and the estimation variance is close to the Cramer-Rao bound. The performance degrades greatly when the separation becomes smaller. This loss in performance is attributed to the increasing interference between the two targets and the difficulty in providing a "good" initial guess for the estimator.

**Key Words:**
- Ballistic Missile Defense (BMD)
- Closely Spaced Object (CBO)
- Cramer-Rao bound
- Detection
- Estimator
- Optical Targets
- Simulation