NPS-OR-95-011PR

NAVAL POSTGRADUATE SCHOOL

Monterey, California

MIXER: A TDA for Mixed Minefield Clearance

Alan R. Washburn

September 1995

Prepared for:
COMINEWARCOM (N02R)
Corpus Christi, TX 78419

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5000

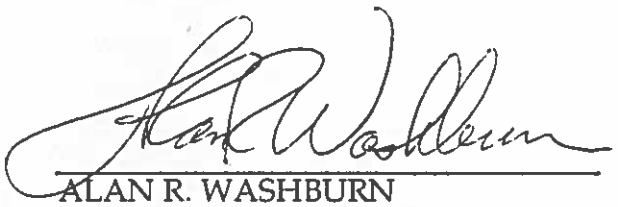
Rear Admiral M. J. Evans
Superintendent

Richard Elster
Provost

This report was prepared for and funded by COMINEWARCOM, Corpus Christi, Texas.

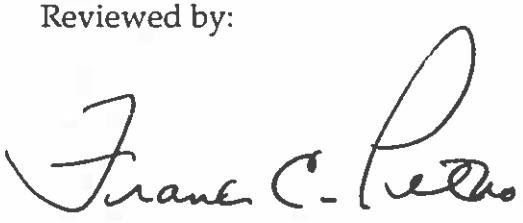
Further dissemination only as directed by COMINEWARCOM, September 1995, or higher DoD authority.

This report was prepared by:



ALAN R. WASHBURN
Professor of Operations Research

Reviewed by:



FRANK PETHO
Acting Chairman
Department of Operations Research

Released by:

PAUL J. MARTO
Dean of Research

MIXER: A TDA for Mixed Minefield Clearance

Alan R. Washburn

Department of Operations Research
Naval Postgraduate School
Monterey, CA 93943

1. Introduction.

MIXER is a Tactical Decision Aid (TDA) intended to help develop clearance plans for minefields where multiple mine types are present, and where minimizing minesweeper losses is one of the goals. MIXER is capable of *assessing* a given plan, and can also (within limits that will be described in section 6) derive an *optimal* plan. MIXER also has *rehearsal* and *reality* functions.

MIXER is currently represented by FORTRAN code that is included with this report and available in electronic form from COMINEWARCOM. This code should be regarded as a prototype, since it has no graphics and little in the way of the user-friendliness features that should be part of an operational TDA. The code makes the computational logic definite, and establishes that the assessment function can be done in real time by Monte Carlo simulation.

The main purpose of this report is to motivate and describe MIXER's logic. Reference will frequently be made to the FORTRAN code, in which case variables used in this report will be related to those used in the code. In particular, the FORTRAN code and this report both use I, J, K, and T as indexes for mine type, sweep type, resource type, and track number, respectively, with IMAX, JMAX, and KMAX being limits for I, J, and K.

Most of the rest of this report concerns the structure and assumptions on which MIXER is built. Possible further developments are discussed in section 8.

2. Katz distributions for number of mines present.

MIXER differs from current minesweeping TDAs in requiring the user to guess the number of mines of each type. The decision to require this guess was made reluctantly, since the user's knowledge of the number of mines is likely to be vague. The trouble is that it doesn't seem to be possible to have a quantitative discussion of losses to either minesweepers or to targets without such estimates, especially when the derivation of plans that are in some sense "optimal" is one of the goals.

In addition to guessing the number of mines of each type, the user of MIXER must also give a *standard deviation* for each guess as an indication of the amount that the guess might be off. MIXER interprets the guess itself as the *mean* μ of the random number X of mines actually present. Letting σ be the input standard deviation, the variance of X is taken to be σ^2. Each replication of the minefield in the simulation has a different value for X, but the values are all sampled from a distribution with mean μ and variance σ^2, except as noted below.

The random variable X, the number of mines of a particular type, is not completely described by the two inputs μ and σ; some distributional assumption must be made. MIXER assumes that X has a *Katz* distribution, a class indexed by two parameters α and β that are related to μ and σ by the equations

$$\mu = \alpha/(1 - \beta) \tag{1.1}$$

$$\sigma^2 = \alpha/(1 - \beta)^2 \tag{1.2}$$

The requirements on α and β are that $\alpha \geq 0$ and $\beta < 1$, plus an additional restriction if $\beta < 0$. Principal properties of Katz distributions are reported in [1]. The class consists of generalized negative binomial ($\beta > 0$), Poisson ($\beta = 0$), and binomial ($\beta < 0$) distributions.

Katz distributions are closed under the sample-and-subtract operation, which makes them natural for studying minesweeping. More precisely, suppose every mine is independently removed with probability p, and that the number of mines removed is observed to be y. Then the distribution of the number of mines remaining, given y, is still Katz, but with revised parameters

$$\alpha' = (1 - p)(\alpha + \beta y) \tag{1.3}$$

$$\beta' = (1 - p)\beta. \tag{1.4}$$

Equations (1.3) and (1.4) are derived in [2]. Katz distributions appear to be the largest class that is closed under the sample-and-subtract operation. The closed-ness of parts of the Katz class has been known and exploited for mine warfare purposes for some time [3].

The advantage of dealing with a closed class is that, if one conceives of minesweeping as a succession of sample-and-subtract stages, then the number of mines in each stage always has the same kind of distribution. One can therefore imagine running MIXER several times in the process of clearing a minefield, once for each stage. In its *reality* mode, MIXER uses the input sweeping plan to calculate p, asks the user for the actual number of mines swept, y, employs (1.3) – (1.4), and outputs α', β' for each mine type to a file NEWNUM.DAT that can serve as the numerical input to the next stage, if there is one. The *rehearsal* mode is similar, except that minesweeping results are determined randomly, rather than input by the user.

The user's μ and σ inputs, one for each mine type, are in the file NUMBERS.DAT, along with the resources available and the sweeping plan. In general, NUMBERS.DAT contains all of the data that are likely to change between stages, with "permanent data" in the other input file PARAMS.DAT. There are no restrictions on μ and σ other than being nonnegative, but there *are* some (μ, σ) pairs for which there is no Katz distribution. There is no Katz distribution if $\mu = 7.5$ and $\sigma = 0$, for example, since Katz distributions are

concentrated on nonnegative integers. Rather than stopping with an error in such cases, MIXER finds the closest Katz distribution by modifying σ. If $(\mu, \sigma) = (7.5, 0)$ is input, for example, MIXER will make $(\alpha, \beta) = (120, -15)$. This corresponds to a binomial distribution with 8 trials and 15/16 success probability, which has the correct mean (7.5), but a variance of 15/32, rather than 0. Such corrections are never necessary if $\sigma^2 \geq \mu$.

Operationally, the biggest defect of Katz distributions is that they place no special weight on the number 0, whereas X is *a priori* likely to be either 0 or "many". In the event that an unexpected mine type is encountered in some phase of minesweeping, the simplest recourse would be to augment the inputs for the next phase with a guess at the number of *remaining* mines of the new type.

3. Resources and sweep types.

MIXER deals with multiple resources, as well as multiple sweep types. Resources can be combined in various ways to make sweep types, with sweep type J requiring one unit of resource K if H(K,J) > 0 in input file PARAMS.DAT. For example, one ship and one EOD team might be combined to make one unit of the "hunting" sweep type, or the ship alone might make a unit of either the "magnetic" or "acoustic" sweep type.

The assumption in all parts of MIXER is that sweeping is done sequentially by sweep type in accordance with the input ordering in the SEQ array. All sweeping for the first sweep type must be completed before any sweeping by the second is carried out, etc. Each sweep type always sweeps at full strength; that is, as many sweeping units as can be made out of the surviving resources are assumed to sweep simultaneously.

Resources can be lost if H(K,J) > 1. Thus a lethal detonation might kill a towed sled (H(K,J) = 2) without killing the helicopter that is towing it (H(K,J) = 1).

4. Inputs in NUMBERS.DAT.

In general NUMBERS.DAT contains "tactical" inputs that depend on circumstances and time. Both this file and PARAMS.DAT use formatted FORTRAN read statements. Therefore, although either file can be edited, it is important to treat integers as being right justified, and not to change the location of the decimal point for real numbers. All real numbers in the sample files have a decimal point in the correct location. Specifically, NUMBERS.DAT includes the following:

a. The number of tracks NTRK(J) for each sweep type J, with $0 \leq \text{NTRK(J)} \leq 40$. This is the number of distinct entry points in the minefield intended for sweep type J. Sweep type JMAX corresponds to target traffic. Targets pass through the minefield once, on a single track, after all sweeping is done. Setting NTRK(JMAX) > 1 simply indicates uncertainty about where that single track will be. NTRK(JMAX) cannot be 0.

b. Track positions X(T,J) for each track. The coordinate system has the width of the minefield extending from 0 to WIDTH, so these positions will typically be in that interval. They are not *required* to be in that interval, but they are required to be integers.

c. Runs per track RUNS (T,J), a nonnegative integer. Each run on each track is made by as many sweeping units as can be assembled out of the remaining resources. No input is required for targets, each of which makes a single run.

d. The μ and σ numbers discussed in section 3. These are in the rows labeled "AVERAGE" and "STD_DEV".

e. The number of resources of type K in the last row labeled "RESOURC".

5. Inputs in PARAMS.DAT.

a. IMAX, JMAX, and KMAX.

b. The array IND(J). This basically distinguishes sweeping (IND = 1) from hunting (IND = 0).

c. The WIDTH and LENGTH of the minefield. The trapezoid adjustment factor TR softens the corners of actuation curves. The curves are rectangular when TR = 0, triangular when TR = 1, and trapezoidal when $0 < TR < 1$. The time HT required by a hunting sweep type to identify a mine is the last input on this line.

d. The navigation error SIG(J), a standard deviation in yards, nonnegative and real.

e. The resource value VAL(K). This is used only in the optimizing subroutine. The last value VAL(KMAX+1) multiplies the probability SIT that the first transitor is sunk; the effect of losses to subsequent transitors can only be accounted for by making this value artificially large.

f. The array H(K,J) that connects sweep types to resource types. One unit of K is required for a unit of J if $H(K,J) > 0$. If $H(K,J) > 1$, the resource unit will be killed by any lethal detonation.

g. The order of entry SEQ(J) for sweep type J.

h. TURN(J), VEL(J), AND DUTY(J). These are used to calculate the time required to sweep the LENGTH of the rectangular minefield, the equation being

$$HOUR(J) = DUTY(J) \times (TURN(J) + LENGTH/VEL(J)) \qquad (5.1)$$

DUTY(J) is a dimensionless factor used to convert time-on-track to the number of hours actually required. These three inputs and also LENGTH are used only to compute HOUR(J), and HOUR(J) is used only in optimization. All are real numbers.

i. The probability actuator setting ACT(I) for mines of type I. A mine counter could be roughly accounted for by letting ACT(I) be the reciprocal of the average number of mine counts required for actuation, the correspondence being exact only if the mine count distribution is geometric. From the user's perspective, being required to guess ACT(I) surely makes as little sense as being required to guess the number of mines. Treating ACT(I) in a Bayesian manner similar to mine numbers is a possibility for future development, but MIXER simply requires ACT(I) to be input. ACT(I) must be such that $0 \leq ACT(I) \leq 1$.

j. The width A(I,J) and height B(I,J) of the actuation curve for mines of type I versus sweep type J, with $A \geq 0$ and $0 \leq B \leq 1$. MIXER uses ACT(I) to modify B(I,J) for sweep types where IND(J) =1, and for nothing else. MIXER has no explicit input for mine sensitivity, but A and B depend on it.

k. The width AF(I,J) within which sweeping a mine may be lethal, and the probability of lethality BF(I,J) given actuation. Thus $AF \geq 0$ and $0 \leq BF \leq 1$, with no requirement for BF to be smaller than B. AF and BF depend on the depth of the detonation, of course.

6. Optimization methodology in subroutine OPT.

The optimization mode is intended to suggest "optimal" uses of resources in the sense that a particular Measure of Effectiveness (MOE) is minimized. MOE is a linear combination of the following quantities:

- total clearance time in hours (TIME)
- simple initial threat (SIT)
- resource losses (LOSS(K), K = 1, ..., KMAX).

Each of these quantities is calculated only approximately, since a fast optimization requires evaluation by analytic formula. The principal approximations are that

(1) Lost resources are assumed to be replaced immediately, so that the sweeping plan is carried out regardless of losses. It is therefore conceivable that LOSS(K) > NU(K), where NU(K) is the input number of units, an event that is logically impossible in the Monte Carlo simulation. When the risk to resources is small, the simulation and the optimization should calculate similar values for LOSS(K) and also SIT.

(2) Every run is assumed to be made on a randomly selected track in the optimization, regardless of the inputs in *X*(T,J). The relationship

$$\text{TOTRUN}(J) = \sum_{T=1}^{\text{NTRK}(J)} \text{RUNS}(T, J) \qquad (6.1)$$

is preserved, but each sweeping unit is assumed to make each run independently of the others. The simulation and the optimization would give strongly different results if all sweeps in the simulation were on the same track, but for reasonably uniform sweeping plans the results should be close.

The equation for MOE is

$$MOE = LAMBDA \times TIME + \sum_{K=1}^{KMAX} VAL(K) \times LOSS(K) + VAL(KMAX+1) \times SIT \quad (6.2)$$

LAMBDA is an interactive user input, whereas the VAL array is read from PARAMS.DAT. VAL(KMAX+1) will usually be relatively large to compensate for the fact that only the first transitor is included in the objective function. The principal tradeoff is between sweeping time and losses, whether the losses are to resources or transitors. By changing LAMBDA interactively, the user can make time small at the cost of large losses, or vice versa. The VAL array can also be changed, but not interactively.

The equation for TIME is

$$TIME = \sum_{J=1}^{JMAX} HOUR(J) \times TOTRUN(J) + HT \times HUNT, \quad (6.3)$$

where HOUR(J) is the number of clock hours required to complete one run of type J as described in equation (5.1), HT is the hunting identification time per mine, and HUNT is the number of mines found by hunting. The number of units sweeping simultaneously NP(J) does not influence the TIME computation because all sweep units are assumed to sweep in parallel, but NP(J) can be seen to influence the computation of losses in equation (6.4) below. If NP(J) is 0, then (6.4) will give no effectiveness to a run of type J and the optimal value of TOTRUN(J) will therefore be 0. This will have the desired effect in (6.3), but (6.3) is still biased high because every planned run takes time in (6.3), even though some planned runs will not be made in reality (or in the Monte Carlo simulation) on account of the loss of resources. Equation (6.3) is also biased high for other reasons (see

section 7.3) but it should still be reasonably accurate for the kind of low-loss sweeping plans that are anticipated to be attractive.

The computations of LOSS and SIT are made in subroutine F, which considers the sequential order in which the runs are made. For run type J, let Q(I,J) be the probability that a mine of type I has survived all previous run types, TH(I,J) the probability that one surviving mine of type I will damage a run of type J, SURV(I,J) the probability that a mine of type I survives a single randomly placed sweep of type J, and MEAN(I) the average number of mines of type I. Then the average number of lethal detonations for sweep type J is given by

$$\mathrm{KILL(J)} = \sum_{\mathrm{I=1}}^{\mathrm{IMAX}} \mathrm{MEAN(I)} \times \mathrm{TH(I,J)} \times \mathrm{Q(I,J)} \times \left(1 - \mathrm{SURV(I,J)}^{\mathrm{TOTRUN(J)} \times \mathrm{NP(J)}}\right), \quad (6.4)$$

the argument being that each mine will cause damage if it survives all previous sweeps, does *not* survive sweep J, and kills the sweeper when it detonates (TH(I,J) is a conditional probability). Kills of each sweep type are then associated with vulnerable assets to obtain LOSS(K), K = 1, ..., KMAX.

The computation of SIT is handled differently because the first transitor may not make it all the way through the minefield. TH(I,JMAX) is the (unconditional) probability that an unswept mine will damage the first transitor, so formula (11) of reference [2] is used with t = TH(I,JMAX) × Q(I,JMAX). This formula is employed once for each mine type.

The optimization method is to simply vary TOTRUN until MOE cannot be decreased by making a unit change in TOTRUN(J) for any J. The resulting local optima are not always global, although the author has so far not found any examples where the defect from global optimality seemed serious. The optimization problem is a nonlinear programming instance with integer variables, a difficult type. It would be comparatively easy if the integer constraint were relaxed, since the objective function is convex. A branch-and-bound approach to global optimization might be based on this observation, as

in reference [5]. A less ambitious approach would simply compare local optima from multiple starting points.

7. Comments on subroutines.

Each of the next subsections concerns one of the five files that constitute MIXER. FORTRAN77 code is reproduced in each subsection. Subroutines use the IOU system for listing arguments, where

- I is for inputs whose values are not changed,
- O is for outputs,
- U is for variables that are updated.

7.1 Main program MIXER, subroutines READIT, DISTRIB, and DUMPIT in file MIXER.FOR.

MIXER calls READIT to get the data in PARAMS.DAT and NUMBERS.DAT, outputs a preliminary summary, makes some preparatory calculations, and then enters a loop where the user can choose one of five options or quit. Specifically, MIXER does the following things (in order):

(1) SEQ(J) is changed to be the index of the J^{th} sweep to enter the minefield.

(2) KZA and KZB are changed from the input μ and σ values to α and β parameters (see section 2).

(3) B is adjusted to account for probability actuators, and the TH and SURV arrays are calculated for use in the theoretical evaluation subroutine F (see sections 6 and 7.2).

(4) NP(J), the number of parallel sweeps of type J that the resources will support, is calculated for use in subroutines MONTE and F.

(5) A and AF are divided by 2, and B and BF are multiplied by MAXINT, the largest integer returned by the random number generating subroutine RAND. This is the last manipulation of COMMON variables, which are essentially the data in PARAMS.DAT. The array P(I) is calculated for use by subroutine KATZ.

(6) The input plan is evaluated by subroutine F, a summary is output, and the option loop is entered.

(7) If OPT is called, it will return NTRK, but not RUNS or *X* (see sections 6 and 7.2). The DISTRIB subroutine distributes the tracks uniformly, using multiple runs per track only if required by array size constraints.

(8) If REHEARSE or REALITY are called, the revised mine and resource parameters are output to NEWNUM.DAT by subroutine DUMPIT, after which MIXER stops.

Subroutines READIT and DUMPIT are called only by MIXER. READIT will stop execution if array sizes are too small for the inputs. DUMPIT converts KZA and KZB from (α, β) to (μ, σ) before output (see section 2).

```
      PROGRAM MIXER
C THIS IS THE MASTER MIXER PROGRAM THAT READS DATA, DOES PREPARATORY
C CALCULATIONS FOR THE PLAN IN NUMBERS.DAT, REPORTS THE RESULTS,
C AND OFFERS THE MAIN MENU
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS, TARGETS
      INTEGER*4 O1,O2,O3,O4,O5,NTARG
      INCLUDE 'LIMITS.DAT'
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH
     +   ,BF(O1,O2),HOUR(O2),SIG(O2),P(O1),TIME,TH(O1,O2),UNDET
     +   ,ACT(O1),SIT,FAC,TMP,Q(O1),SURV(O1,O2),TR,A1,A2,A3,A4,LOSS(O4)
     +   ,COST,MEAN(O1),VAL(O4),HT,HUNT
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7,TEXT(5)*80
      INTEGER*4 IMAX,JMAX,KMAX,I,J,K,R,JM1,REPLY,SEED,FLAG
     +   ,RUNS(O5,O2),NTRK(O2),T,IND(O2),SEQ(O2)
     +   ,NN,H(O4,O2),NU(O4),NP(O2),X(O5,O2),MAXINT,TOTRUN(O2)
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +       ,SEQ,IMAX,JMAX,KMAX,IND,H
     +       , NM,NS,NR
      DATA MAXINT /2147483647/
      WRITE(6,*)' ***********WELCOME TO MIXER(1.5)********'
      WRITE(6,*)' READING THE SCENARIO IN PARAMS.DAT AND NUMBERS.DAT'
      CALL READIT(X,NTRK,RUNS,NU,KZA,KZB,TEXT)
C JM1 IS THE NUMBER OF SWEEP TYPES, NOT COUNTING TARGETS
      JM1=JMAX-1
C INVERT THE SEQUENCE FUNCTION FOR MONTE USING NP AS TEMPORARY STORE
      DO 4 J=1,JM1
4     NP(SEQ(J))=J
      DO 6 J=1,JM1
6     SEQ(J)=NP(J)
      SEED=4520783
      WRITE(6,11)IMAX,JM1,KMAX
11    FORMAT(I3,' MINE TYPES,',I3,' SWEEP TYPES,',I3,' RESOURCE TYPES')
      FLAG=0
      DO 40 I=1,IMAX
```

```
C TRANSFORM INPUTS TO KATZ PARAMETERS
      TMP=KZB(I)*KZB(I)
      IF(KZA(I).LE.0)THEN
         WRITE(6,*)' INCLUDING NO MINES OF TYPE ',I
         KZA(I)=0
         KZB(I)=0
      ELSEIF(KZA(I).LE.TMP)THEN
         FAC=KZA(I)/TMP
         KZA(I)=FAC*KZA(I)
         KZB(I)=1.-FAC
      ELSE
         FAC=KZA(I)*KZA(I)/(KZA(I)-TMP)
C R IS AN INTEGER, SO USER'S INPUTS MODIFIED SLIGHTLY IN BINOM CASE
         R=FAC+.5
         IF(KZA(I).GE.R)R=R+1
         KZB(I)=KZA(I)/(KZA(I)-R)
         KZA(I)=KZA(I)*(1.-KZB(I))
      ENDIF
      MEAN(I)=KZA(I)/(1.-KZB(I))
      DO 30 J=1,JMAX
         A1=A(I,J)
C ACTIVATION PROBABILITY ACCOUNTED FOR BY MODIFYING B
         IF(IND(J).EQ.1)B(I,J)=B(I,J)*ACT(I)
         TMP=A1*B(I,J)
C SURV IS THE PROBABILITY THAT A MINE SURVIVES A UNIT SWEEP
C SURV IS NEEDED FOR THEORETICAL CALCS, NOT FOR SIMULATION
         IF(TMP.GT.WIDTH)THEN
            FLAG=1
            SURV(I,J)=0
         ELSE
            SURV(I,J)=1.-TMP/WIDTH
         ENDIF
         A2=A1*(1+TR)
         A3=A1*(1-TR)
         A4=AF(I,J)
         IF(A4.GE.A2)THEN
            A4=A1
         ELSEIF(A4.GT.A3)THEN
            A4=A3+.5*(A4-A3)*(1.+(A2-A4)/(A2-A3))
         ENDIF
C AT THIS POINT A4 IS THE AREA UNDER THE ACTUATION CURVE WITHIN LETHAL
C RANGE, EXCEPT FOR THE B FACTOR, SO NOW CALCULATE CONDITIONAL PROB
C TH(I,J) THAT A MINE KILLS A SWEEP, GIVEN ACTUATION
         TMP=A4*BF(I,J)
         IF(A1.GT.0)THEN
            IF(TMP.LE.A1)THEN
               TH(I,J)=TMP/A1
            ELSE
               WRITE(6,*)' STOPPING WITH BAD SETUP IN MIXER'
               STOP
            ENDIF
         ELSE
            TH(I,J)=0.
         ENDIF
30    CONTINUE
C "THREAT" FOR VALUE TRAFFIC SHOULD BE UNCONDITIONAL IN SUBROUTINE F
      TH(I,JMAX)=TH(I,JMAX)*(1.-SURV(I,JMAX))
40    CONTINUE
      IF(FLAG.EQ.1)WRITE(6,*)' WARNING: TRUNCATION DUE TO SMALL WIDTH'
```

```
      TIME=0
      WRITE(6,*)' INPUT RUNS PER TRACK ARE...'
      DO 110 J=1,JM1
         NN=999
         DO 50 K=1,KMAX
            IF(H(K,J).NE.0)NN=MIN(NN,NU(K))
50       CONTINUE
         IF(NN.EQ.999)THEN
            WRITE(6,*)' NO RESOURCE LIMITS (!) FOR UNITS OF TYPE ',J
            WRITE(6,*)' FIX H(K,J) AND RE-RUN'
            STOP
         ENDIF
C NP(J) IS THE RESOURCE DEPENDENT NUMBER OF PARALLEL SWEEPS PER RUN
         WRITE(6,138)NS(J),(RUNS(T,J),T=1,NTRK(J))
         NP(J)=NN
         TOTRUN(J)=0
         DO 100 T=1,NTRK(J)
100      TOTRUN(J)=TOTRUN(J)+RUNS(T,J)
110   TIME=TIME+TOTRUN(J)*HOUR(J)
      DO 114 I=1,IMAX
         DO 113 J=1,JM1
C HALF SWEEP WIDTHS WANTED AFTER THIS, MAXINT IS LARGEST RANDOM NO.
            A(I,J)=A(I,J)*.5
            AF(I,J)=AF(I,J)*.5
            B(I,J)=B(I,J)*MAXINT
            BF(I,J)=BF(I,J)*MAXINT
113      CONTINUE
         A(I,JMAX)=A(I,JMAX)*.5
         AF(I,JMAX)=AF(I,JMAX)*.5
         B(I,JMAX)=B(I,JMAX)*MAXINT
         BF(I,JMAX)=BF(I,JMAX)*MAXINT
C P(I) IS FOR THE KATZ SUBROUTINE
         IF(ABS(KZB(I)).LT..001)THEN
            P(I)=MAXINT*EXP(-KZA(I))
         ELSE
            P(I)=MAXINT*(1.-KZB(I))**(KZA(I)/KZB(I))
         ENDIF
114   CONTINUE
C EVALUATE THE INPUT PLAN -- ALL COMMON VARIABLES FIXED FROM NOW ON
120   CALL F(TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,LOSS,COST,SIT,Q,HUNT)
      WRITE(6,*)' THEORETICAL CLEARANCE LEVELS:'
      WRITE(6,115)(NM(I),I=1,IMAX)
115   FORMAT(2X,9A8)
      WRITE(6,116)((1-Q(I)),I=1,IMAX)
116   FORMAT(10F8.4)
      TIME=TIME+HT*HUNT
      WRITE(6,117)TIME,SIT
117   FORMAT(' THEORETICAL RESULTS ARE TIME(HRS):',F8.2,' SIT:',F6.4,' A
     +VERAGE LOSSES ARE...')
      WRITE(6,118)(NR(K),LOSS(K),K=1,KMAX)
118   FORMAT(5(1X,A7,':',F6.2,1X))
121   WRITE(6,*)' CHOOSE: QUIT(0), OPT(1), INPUT(2), MONTE(3), REHEARSE(
     +4), REALITY(5)'
      READ(5,*)REPLY
      IF(REPLY.EQ.1)THEN
         CALL OPT(KZA,KZB,TH,SURV,NP,MEAN,SIT,LOSS,TIME,HUNT,TOTRUN)
         CALL DISTRIB(TOTRUN,WIDTH,JM1,NTRK,RUNS,X)
         WRITE(6,131)
131      FORMAT(' CURRENTLY USING RUNS PER EVENLY SPACED TRACK AS FOLLOW
```

```
     +S')
          DO 135 J=1,JM1
135       WRITE(6,138)NS(J),(RUNS(T,J),T=1,NTRK(J))
138       FORMAT(A8,1X,70I1)
          GO TO 121
         ELSEIF(REPLY.EQ.2)THEN
          WRITE(6,*)' note:EACH RUN IS MADE BY ALL AVAILABLE SWEEP UNITS'
139       WRITE(6,145)'INDEX:',(J,J=1,JM1)
          WRITE(6,143)'SWEEP:',(NS(J), J=1,JM1)
          WRITE(6,145)'TOTRUN:',(TOTRUN(J),J=1,JM1)
          WRITE(6,*)' INPUT THE INDEX OF THE SWEEP TYPE YOU WANT TO CHANG
     +E (0 IF NONE)'
          READ(5,*)REPLY
          IF(REPLY.GT.0.AND.REPLY.LE.JM1)THEN
             WRITE(6,*)' INPUT THE TOTAL NUMBER OF RUNS YOU WANT'
             READ(5,*)TOTRUN(REPLY)
             TOTRUN(REPLY)=MAX(TOTRUN(REPLY),0)
             GO TO 139
          ENDIF
          CALL DISTRIB(TOTRUN,WIDTH,JM1,NTRK,RUNS,X)
          WRITE(6,131)
          DO 140 J=1,JM1
140       WRITE(6,138)NS(J),(RUNS(T,J),T=1,NTRK(J))
          GO TO 120
143       FORMAT(10A8)
145       FORMAT(A8,10I8)
         ELSEIF(REPLY.EQ.3)THEN
          CALL MONTE(X,RUNS,NTRK,P,NU,NP,KZA,KZB,SEED)
          GO TO 121
         ELSEIF(REPLY.EQ.4)THEN
          CALL REHEARSE(X,RUNS,NTRK,P,SURV,NU,KZA,KZB,SEED)
         ELSEIF(REPLY.EQ.5)THEN
          CALL REALITY(NU,KZA,KZB)
         ENDIF
C CONVERT KATZ FROM ALPHA-BETA TO MU-SIGMA BEFORE STOPPING
         DO 455 I=1,IMAX
         IF(KZB(I).GE.1.)THEN
          WRITE(6,*)' LOGICAL FAULT IN MIXER, NULLING MINE TYPE ',I
          KZA(I)=0.
          KZB(I)=0.
         ELSEIF(KZA(I).LE.0)THEN
          WRITE(6,*)' NO MINES OF TYPE ',I,' REMAIN'
          KZA(I)=0.
          KZB(I)=0.
         ELSE
          KZA(I)=KZA(I)/(1.-KZB(I))
          KZB(I)=SQRT(KZA(I)/(1.-KZB(I)))
         ENDIF
455      CONTINUE
         CALL DUMPIT(TEXT,NTRK,X,RUNS,KZA,KZB,NU,IMAX,JMAX,KMAX,JM1)
         WRITE(6,*)'***************THANK YOU FOR USING MIXER*************'
         END

         SUBROUTINE DISTRIB(
        I                    TOTRUN,WIDTH,JM1
        O                    ,NTRK,RUNS,X)
         IMPLICIT NONE
C THIS SUBROUTINE DISTRIBUTES A TOTAL NUMBER OF RUNS OVER EVENLY
C SPACED TRACKS, BUT NO MORE THAN 05 TRACKS
```

```
      INTEGER*4 O1,O2,O3,O4,O5,NTARG
      INCLUDE 'LIMITS.DAT'
      REAL*4 WIDTH,FAC,TMP
      INTEGER*4 I,J,K,R,JM1
     +   ,RUNS(O5,O2),NTRK(O2),T,X(O5,O2),TOTRUN(O2)
         DO 140 J=1,JM1
            IF(TOTRUN(J).EQ.0)THEN
               NTRK(J)=0
            ELSEIF(TOTRUN(J).LE.O5)THEN
               NTRK(J)=TOTRUN(J)
               FAC=WIDTH/NTRK(J)
               TMP=FAC*.5+.5
               DO 135 T=1,NTRK(J)
                  X(T,J)=TMP
                  RUNS(T,J)=1
                  TMP=TMP+FAC
135            CONTINUE
            ELSE
               K=TOTRUN(J)/O5
               I=TOTRUN(J)-K*O5
               NTRK(J)=O5
               FAC=WIDTH/NTRK(J)
               TMP=FAC*.5+.5
               DO 136 T=1,NTRK(J)
                  X(T,J)=TMP
                  RUNS(T,J)=K
                  TMP=TMP+FAC
136            CONTINUE
               IF(I.GT.0)THEN
                  K=O5/I
                  R=(O5-K*I+K+1)/2
                  DO 137 T=1,I
                     RUNS(R,J)=RUNS(R,J)+1
                     R=R+K
137               CONTINUE
               ENDIF
            ENDIF
140      CONTINUE
      END

      SUBROUTINE READIT(
     O                  X,NTRK,RUNS,NU,KZA,KZB,TEXT)
      IMPLICIT NONE
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),TURN(O2)
     +       ,BF(O1,O2),HOUR(O2),WIDTH,TR,SIG(O2),ACT(O1),VEL(O2)
     +       ,LENGTH,DUTY(O2),VAL(O4),HT
      INTEGER*4 IMAX,JMAX,KMAX,I,J,K,T,JM1,IND(O2),H(O4,O2),NU(O4)
     +         ,SEQ(O2),X(O5,O2),NTRK(O2),RUNS(O5,O2)
      CHARACTER NAME*7,TEXT(5)*80
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +       ,SEQ,IMAX,JMAX,KMAX,IND,H
     +       , NM,NS,NR
      OPEN(UNIT =8, FILE='PARAMS.DAT')
      OPEN(UNIT =9, FILE='NUMBERS.DAT')
      READ(8,*)
      READ(8,*)
```

```
      READ(8,10) IMAX,JMAX,KMAX
      IF(IMAX.GT.01.OR.JMAX.GT.02.OR.KMAX.GE.04)THEN
         WRITE(6,*)' READIT SAYS RECOMPILE WITH BIGGER ARRAYS...'
         STOP
      ENDIF
      JM1=JMAX-1
      READ(8,*)
      READ(8,10)(IND(J),J=1,JM1)
      READ(8,*)
      READ(8,20)NAME,WIDTH,LENGTH,TR,HT
10    FORMAT(19I4)
20    FORMAT(A7,9F8.2)
25    FORMAT(A7,9I8)
30    FORMAT(A7,9A8)
      READ(8,*)
      READ(8,20)NAME,(SIG(J),J=1,JMAX)
      READ(8,*)
      READ(8,*)
      READ(8,*)
      DO 36 K=1,KMAX
36    READ(8,25)NR(K),(H(K,J),J=1,JM1)
      READ(8,*)
      READ(8,20)NAME,(VAL(K),K=1,KMAX+1)
      READ(8,*)
      READ(8,25)NAME,(SEQ(J),J=1,JM1)
      SEQ(JMAX)=JMAX
      READ(8,*)
      READ(8,20)NAME,(TURN(J),J=1,JM1)
      READ(8,*)
      READ(8,20)NAME,(VEL(J),J=1,JM1)
      READ(8,*)
      READ(8,20)NAME,(DUTY(J),J=1,JM1)
      DO 38 J=1,JM1
      IF(VEL(J).GT.0)THEN
         HOUR(J)=DUTY(J)*(TURN(J)+LENGTH/VEL(J))
      ELSE
         WRITE(6,37)J
37       FORMAT(' VELICITY FOR SWEEP TYPE ',I3,' IS NOT POSITIVE!')
         STOP
      ENDIF
38    CONTINUE
      READ(8,*)
      READ(8,20)NAME,(ACT(I),I=1,IMAX)
      READ(8,*)
      READ(8,30)NAME,(NS(J),J=1,JMAX)
      DO 40 I=1,IMAX
40    READ(8,20)NM(I),(A(I,J),J=1,JMAX)
      READ(8,*)
      READ(8,*)
      DO 50 I=1,IMAX
50    READ(8,20)NAME,(B(I,J),J=1,JMAX)
      READ(8,*)
      READ(8,*)
      DO 60 I=1,IMAX
60    READ(8,20)NAME,(AF(I,J),J=1,JMAX)
      READ(8,*)
      READ(8,*)
      DO 70 I=1,IMAX
70    READ(8,20)NAME,(BF(I,J),J=1,JMAX)
```

```
      READ(9,110)TEXT(1)
      READ(9,25)NAME,(NTRK(J),J=1,JMAX)
      DO 80 J=1,JMAX
      IF(NTRK(J).GT.O5)THEN
         WRITE(6,*)' TOO MANY TRACKS OF TYPE ',J
         STOP
      ENDIF
80    CONTINUE
      READ(9,110)TEXT(2)
      DO 90 J=1,JMAX
      READ(9,100)(X(T,J),T=1,NTRK(J))
90    CONTINUE
      READ(9,110)TEXT(3)
      DO 95 J=1,JM1
95    READ(9,100)(RUNS(T,J),T=1,NTRK(J))
      DO 97 T=1,NTRK(JMAX)
97    RUNS(T,JMAX)=1
      READ(9,110)TEXT(4)
      READ(9,20)NAME,(KZA(I),I=1,IMAX)
      READ(9,20)NAME,(KZB(I),I=1,IMAX)
      READ(9,110)TEXT(5)
      READ(9,25)NAME,(NU(K),K=1,KMAX)
100   FORMAT(13I6)
110   FORMAT(A80)
      CLOSE(8)
      CLOSE(9)
      END

      SUBROUTINE DUMPIT(
     I          TEXT,NTRK,X,RUNS,KZA,KZB,NU,IMAX,JMAX,KMAX,JM1)
      IMPLICIT NONE
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 KZA(O1),KZB(O1)
      INTEGER*4 X(O5,O2),RUNS(O5,O2),I,J,K,IMAX,JMAX,KMAX,JM1,T,NU(O4)
     +          ,NTRK(O2)
      CHARACTER TEXT(5)*80
      OPEN(UNIT=10,FILE='NEWNUM.DAT')
      WRITE(10,670)TEXT(1)
      WRITE(10,660)'NTRK(J)',(NTRK(J),J=1,JMAX)
      WRITE(10,670)TEXT(2)
      DO 620 J=1,JMAX
      WRITE(10,680)(X(T,J),T=1,NTRK(J))
620   CONTINUE
      WRITE(10,670)TEXT(3)
      DO 630 J=1,JM1
630   WRITE(10,680)(RUNS(T,J),T=1,NTRK(J))
      WRITE(10,670)TEXT(4)
      WRITE(10,650)'AVERAGE',(KZA(I),I=1,IMAX)
      WRITE(10,650)'STD_DEV',(KZB(I),I=1,IMAX)
      WRITE(10,670)TEXT(5)
      WRITE(10,660)'RESOURC',(NU(K),K=1,KMAX)
      CLOSE(10)
      WRITE(6,*)' OUTPUT HAS BEEN WRITTEN TO NEWNUM.DAT.  TO CONTINUE,'
      WRITE(6,*)' EDIT NEWNUM.DAT, COPY IT TO NUMBERS.DAT, AND RUN MIXER
     + AGAIN.'
650   FORMAT(A7,9F8.2)
660   FORMAT(A7,9I8)
670   FORMAT(A80)
```

```
680   FORMAT(13I6)
      END
```

7.2 Subroutines OPT and F in file OPT.FOR.

Subroutine OPT optimizes array TOTRUN by dithering the array until no unit change results in an improvement in MOE (see section 6). Evaluation of the terms in MOE is through subroutine F.

Subroutine F implements equation (6.4) and also equation (11) of reference [2] to calculate LOSS(K), COST, SIT, and the clearance levels Q. COST is simply the total cost of all losses, weighted using the unit values in VAL(K). All inputs except for TOTRUN(J) are calculated in MIXER and passed to F via OPT. The array Q(I,J) referred to in section 6 actually does not require the J argument in subroutine F.

```
      SUBROUTINE OPT(
     I                KZA,KZB,TH,SURV,NP,MEAN
     O                ,SIT,LOSS,TIME,HUNT
     U                ,TOTRUN)
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O4,O5
      PARAMETER(O1=10,O2=9,O4=8,O5=40)
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR,HUNT
     +      ,BF(O1,O2),HOUR(O2),SURV(O1,O2),SIG(O2),COST,SIT,TIME,HT
     +      ,ACT(O1),TMP,TH(O1,O2),LAMBDA,LOSS(O4),MEAN(O1),Q(O2),VAL(O4)
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,JM1,NP(O2)
     +   ,IND(O2),SEQ(O2),H(O4,O2),J,K,TOTRUN(O2),FLAG
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +        ,SEQ,IMAX,JMAX,KMAX,IND,H
     +        ,NM,NS,NR
      JM1=JMAX-1
      CALL F(TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,LOSS,COST,SIT,Q,HUNT)
10    WRITE(6,*)'INPUT PRICE OF TIME, ABOUT 100, OR 0 FOR MAIN MENU:'
      READ(5,*)LAMBDA
      IF(LAMBDA.LE.0)RETURN
      LAMBDA=LAMBDA*1E-6
      COST=COST+LAMBDA*HT*HUNT
35    FLAG=0
      DO 70 J=1,JM1
      IF(NP(J).LE.0)GO TO 70
      TMP=COST
      TOTRUN(J)=TOTRUN(J)+1
      CALL F(TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,LOSS,COST,SIT,Q,HUNT)
      COST=COST+LAMBDA*HUNT*HT
      IF(COST+LAMBDA*HOUR(J).LT.TMP)THEN
         FLAG=1
      ELSEIF(TOTRUN(J).GT.1)THEN
```

```
            TOTRUN(J)=TOTRUN(J)-2
            CALL F(TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,LOSS,COST,SIT,Q,HUNT)
            COST=COST+LAMBDA*HUNT*HT
            IF(COST-LAMBDA*HOUR(J).LT.TMP)THEN
               FLAG=1
            ELSE
               TOTRUN(J)=TOTRUN(J)+1
               COST=TMP
            ENDIF
         ELSE
            TOTRUN(J)=0
            COST=TMP
         ENDIF
70       CONTINUE
         IF(FLAG.EQ.1)THEN
            GO TO 35
         ELSE
            CALL F(TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1,LOSS,COST,SIT,Q,HUNT)
         ENDIF
         TIME=0
         DO 80 J=1,JM1
         TIME=TIME+TOTRUN(J)*HOUR(J)
80       CONTINUE
         TIME=TIME+HUNT*HT
         WRITE(6,117)TIME,SIT
117      FORMAT(' THEORETICAL RESULTS ARE TIME(HRS):',F8.2,' SIT:',F6.4,' A
        +VERAGE LOSSES NEXT:')
         WRITE(6,118)(NR(K),LOSS(K),K=1,KMAX)
118      FORMAT(5(A7,':',F6.2,2X))
         GO TO 10
         END

         SUBROUTINE F(
        I              TOTRUN,NP,TH,SURV,KZA,KZB,MEAN,JM1
        O              ,LOSS,COST,SIT,Q,HUNT)
         IMPLICIT NONE
         INTEGER*4 O1,O2,O4
         PARAMETER(O1=10,O2=9,O4=8)
         REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR,HUNT
        +      ,BF(O1,O2),HOUR(O2),SURV(O1,O2),SIG(O2),FAC,TMP,VAL(O4)
        +      ,ACT(O1),TH(O1,O2),HT
         REAL*4 LOSS(O4),COST,SIT,KILL(O2),Q(O2),MEAN(O1)
         CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
         INTEGER*4 IMAX,JMAX,KMAX,J,JM1,TOTRUN(O2)
        +   ,IND(O2),SEQ(O2),H(O4,O2),I,K,JP,R,NP(O2)
         COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
        +        ,SEQ,IMAX,JMAX,KMAX,IND,H
        +        ,NM,NS,NR
         COST=0.
         HUNT=0.
         DO 5 K=1,KMAX
            LOSS(K)=0.
5        CONTINUE
         DO 8 J=1,JM1
8        KILL(J)=0
         DO 10 I=1,IMAX
10       Q(I)=1.
         DO 40 JP=1,JM1
         J=SEQ(JP)
```

```
C CALCULATE LETHAL HITS ON TYPE J SWEEPS
      R=TOTRUN(J)*NP(J)
      DO 20 I=1,IMAX
         FAC=SURV(I,J)**R
         TMP=MEAN(I)*Q(I)*(1.-FAC)
         IF(IND(J).EQ.0)HUNT=HUNT+TMP
         KILL(J)=KILL(J)+TH(I,J)*TMP
         Q(I)=Q(I)*FAC
20    CONTINUE
C TRANSLATE SWEEP KILLS TO RESOURCE KILLS
      DO 30 K=1,KMAX
      IF(H(K,J).EQ.2)THEN
         LOSS(K)=LOSS(K)+KILL(J)
         COST=COST+VAL(K)*KILL(J)
      ENDIF
30    CONTINUE
40    CONTINUE
C CALCULATE SIT USING KATZ GENERATING FUNCTION
      SIT=1.
      DO 60 I=1,IMAX
         TMP=TH(I,JMAX)*Q(I)
         FAC=KZB(I)
         IF(ABS(FAC).GT..001)THEN
            SIT=SIT*(1+FAC*TMP/(1.-FAC))**(-KZA(I)/FAC)
         ELSE
            SIT=SIT*EXP(-KZA(I)*TMP)
         ENDIF
60    CONTINUE
      SIT=1.-SIT
      COST=COST+VAL(KMAX+1)*SIT
      END
```

7.3 Subroutines MONTE, RAND, NORMAL, and KATZ in file MONTE.FOR.

MONTE is the slowest of MIXER's options because Monte Carlo simulation requires many replications to produce accurate results. The user inputs the desired number of replications (in 100's), and must then wait on the order of a second per hundred replications, depending on the scenario. Probability estimates are accurate to within about $1/\sqrt{n}$ if there are n replications, so measuring probabilities to within .01 requires 10,000 replications or about 100 seconds. For exploratory work a few hundred iterations may suffice.

The benefit of Monte Carlo simulation is that calculations closely imitate what happens in reality. If the sampling error is minimized by using a large number of iterations, results can therefore be quite accurate.

The following operations take place within MONTE:

(1) For each mine type, generate the number of mines using the KATZ subroutine, assign a cross-channel location XM(M) to each one, and let MINES be the total number.

(2) Intermix the mines in the along-channel direction, letting MINTYP(M) be the type of the M^{th} mine that the sweepers will encounter in the forward direction. Initially STATUS(M) = 0 for all M, indicating that the mine has not yet been swept.

(3) Considering the sweep types in the order stored in SEQ(J), make all runs forward, then backward, etc., until all runs and sweep types have been considered. When an unswept mine is encountered, test whether it actuates, and, if so, whether the detonation is lethal to a resource unit. When a sweep resource of type K is lost, reduce the number NUN(K) of remaining resources, and, if necessary, the number NN of units still sweeping simultaneously.

(4) When all sweeping is done, pass target traffic through the minefield using a similar logic.

(5) Convert the collected totals to averages and output.

(6) Offer the user the chance to repeat the calculations or return to MIXER.

MONTE does not calculate the amount of time required for sweeping. If it did, then the average sweeping time would be smaller than equation (6.3) because (6.3) does not account for the possibility that the number of units sweeping in parallel might fall to 0, in which case no additional time is required to complete the sweep time. However, it is hard to imagine being attracted to a sweeping plan whose average time is significantly smaller than (6.3), since such plans would presumably be rejected on account of high resource losses. This being the case, there seems to be little point in keeping track of time in MONTE.

The assumptions behind equation (6.3) are suspect in other ways. In reality there may be opportunities for sweep types involving different resources to proceed in parallel, and

consideration must be given to logistics, day versus night, the weather, etc. in scheduling sweeps. All of these things could be accounted for in a Monte Carlo simulation, but only at the cost of slowing it down. Given current computer speeds, it appears that a simulation fast enough to be usable tactically should confine itself to assessing losses.

Subroutine RAND is essentially Schrage's implementation of what Park and Miller [4] call the "minimal standard", except that the division required to put the output in the unit interval is omitted for the sake of efficiency.

Subroutine NORMAL gets a unit normal by summing 12 uniforms. NORMAL is used to assign navigation errors to sweepers (all simultaneous sweepers get the same error) and to transitors (each transitor gets an error independent of the others).

Subroutine KATZ generates a Katz (α, β) sample X by exploiting the fact that the ratio $\text{Prob}(X = j+1)/\text{Prob}(X = j)$ is $(\alpha + \beta j)/(j+1)$. The quantity P(I) calculated in MIXER and passed to KATZ is the initial probability $\text{Prob}(X = 0)$.

```
      SUBROUTINE MONTE(
     I                  X,RUNS,NTRK,P,NU,NP,KZA,KZB
     U                  ,SEED)
C THIS IS THE SUBROUTINE THAT SIMULATES THE SWEEPING OF THE MINEFIELD
C AND THE SUBSEQUENT PASSAGE OF TRAFFIC
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS, TARGETS
      INTEGER*4 O1,O2,O3,O4,O5,NTARG
      INCLUDE 'LIMITS.DAT'
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR
     +   ,BF(O1,O2),HOUR(O2),XM(O3),THRT(NTARG),SIG(O2),VAL(O4)
     +   ,DIST,POS,CAS(O1,O4),SWPT(O1,O2),A1,A2,B1,CASTOT(O4)
     +   ,ACT(O1),TMP,TOTRUN(O2),P(O1),NAVERR(NTARG),HT
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,I,J,JP,K,R,JM1,NUM(O1),MINES,MINTYP(O3)
     +   ,RUNS(O5,O2),NTRK(O2),T,LO,M,STATUS(O3),IND(O2),SEQ(O2)
     +   ,INCR,MM,REP,REPMAX,NN,NNN,NNNN,H(O4,O2),NU(O4),NUN(O4),IR,O3M1
     +   ,X(O5,O2),U(O3),MAXINT,SEED,INTREP,LIVE(NTARG),NP(O2)
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +       ,SEQ,IMAX,JMAX,KMAX,IND,H
     +       ,NM,NS,NR
      DATA MAXINT /2147483647/
      O3M1=O3-1
      JM1=JMAX-1
      WRITE(6,*)' ENTER NUMBER OF REPLICATIONS IN HUNDREDS, OR 0 FOR MAI
     +N MENU:'
      READ(5,*)REPMAX
      IF(REPMAX.LE.0)RETURN
```

```
C INITIALIZE COUNTERS FOR:
C NUMBER OF I MINES SWEPT BY J SWEEPS IN SWPT
C NUMBER OF K RESOURCES KILLED BY I MINES IN CAS (TOTAL FOR K IN CASTOT)
C NUMBER OF COMPLETED J RUNS, INCLUDING PARALLEL FACTOR, IN TOTRUN
C NUMBER OF TIMES NNNN TARGET IS KILLED IN THRT
10     DO 130 I=1,IMAX
       DO 124 J=1,JMAX
124    SWPT(I,J)=0
       DO 125 K=1,KMAX
125    CAS(I,K)=0
130    CONTINUE
       DO 132 K=1,KMAX
132    CASTOT(K)=0
       DO 135 J=1,JM1
135    TOTRUN(J)=0
       DO 140 NNNN=1,NTARG
140    THRT(NNNN)=0
       IF(REPMAX.GT.3)WRITE(6,'(\A)')'WAIT:'

C START THE MAIN LOOP
       DO 539 REP=1,REPMAX
       IF(REPMAX.GT.3)WRITE(6,'(\A)')'*'
C COMMENT PREVIOUS 2 IF STATEMENTS AND UNCOMMENT NEXT FOR MICROSOFT
C      IF(REPMAX.GT.3)WRITE(6,145)REP
145    FORMAT('+REPLICATION ',I6)
       DO 539 INTREP=1,100
C INITIALIZE THE NUMBER OF K UNITS IN NUN, WHICH MAY DECREASE
       DO 150 K=1,KMAX
150    NUN(K)=NU(K)
       MINES=0
C GENERATE THE MINE NUMBERS AND INITIALIZE CLEARANCE
       CALL RAND(IMAX,U,SEED)
       DO 170 I=1,IMAX
       CALL KATZ(U(I),KZA(I),KZB(I),P(I),O3,NUM(I))
       MINES=MINES+NUM(I)
       SWPT(I,JMAX)=SWPT(I,JMAX)+NUM(I)
       IF(MINES.GT.O3)THEN
          WRITE(6,*)' TRUNCATING TOTAL MINES TO ',O3
          MINES=O3
       ENDIF
170    CONTINUE
C GENERATE A CROSS-CHANNEL LOCATION FOR EACH MINE
       CALL RAND(MINES,U,SEED)
       TMP=WIDTH/MAXINT
       DO 180 M=1,MINES
180    XM(M)=TMP*U(M)
C PUT THE MINES IN RANDOM ORDER
       CALL RAND(MINES,U,SEED)
       LO=MINES
182    IF(LO.EQ.0)GO TO 187
       J=1+MOD(U(LO),LO)
       K=0
       DO 185 I=1,IMAX
          K=K+NUM(I)
          IF(J.LE.K)THEN
             MINTYP(LO)=I
             STATUS(LO)=0
             LO=LO-1
             NUM(I)=NUM(I)-1
```

```
            GO TO 182
          ENDIF
185     CONTINUE
187     CONTINUE
        CALL RAND(O3,U,SEED)
C IR USED TO DECIDE WHEN TO GET ANOTHER BATCH OF O3 RANDOM NUMBERS
        IR=0
C THE 450 LOOP GOES THROUGH THE SWEEP TYPES IN ORDER OF ENTRY
C MINES ENCOUNTERED IN REVERSE ORDER WHEN INCR = -1 EVERY OTHER RUN
        DO 450 JP=1,JM1
           J=SEQ(JP)
           INCR=1
           MM=1
           NN=999
           DO 200 K=1,KMAX
              IF(H(K,J).NE.0)NN=MIN(NN,NUN(K))
200        CONTINUE
           IF(NN.EQ.0)GO TO 450
           DO 420 T=1,NTRK(J)
           CALL NORMAL(1,TMP,SEED)
C ALL PARALLEL SWEEPS HAVE THE SAME NAV ERROR
           POS=X(T,J)+TMP*SIG(J)
           DO 418 R=1,RUNS(T,J)
              DO 400 M=1,MINES
C STATUS SHOWS WHO SWEPT THE MINE, IF ANYONE
              IF(STATUS(MM).NE.0)GO TO 400
              I=MINTYP(MM)
              A2=A(I,J)
              IF(A2.EQ.0)GO TO 400
C IMPLEMENT THE TRAPEZOIDAL CORRECTION
              A1=A2*(1+TR)
              A2=A2*(1-TR)
              DIST=ABS(XM(MM)-POS)
              IF(DIST.LT.A1)THEN
                 IF(DIST.LT.A2)THEN
                    B1=B(I,J)
                 ELSE
                    B1=B(I,J)*(DIST-A2)/(A1-A2)
                 ENDIF
                 NNN=NN
C THE MINE IS TESTED AGAINST ALL SURVIVING PARALLEL SWEEPERS
                 DO 220 NNNN=1,NNN
                 IF(STATUS(MM).NE.0)GO TO 400
                 IR=IR+1
                 IF(IR.GE.O3M1)THEN
                    CALL RAND(O3,U,SEED)
                    IR=1
                 ENDIF
                 IF(U(IR).LT.B1)THEN
                    STATUS(MM)=J
                    SWPT(I,J)=SWPT(I,J)+1
                    IF(DIST.LT.AF(I,J))THEN
                       IR=IR+1
                       IF(U(IR).LT.BF(I,J))THEN
                          DO 210 K=1,KMAX
                             IF(H(K,J).EQ.2)THEN
                                CAS(I,K)=CAS(I,K)+1
                                IF(NN.EQ.NUN(K))NN=NN-1
                                NUN(K)=NUN(K)-1
```

```
                              ENDIF
210                         CONTINUE
                         ENDIF
                      ENDIF
                   ENDIF
220                CONTINUE
C                  IF NO UNITS OF SWEEP LEFT, GO ON TO NEXT SWEEP TYPE
                   IF(NN.EQ.0)GO TO 450
                ENDIF
400          MM=MM+INCR
C NOTE FRACTIONAL RUNS ARE NOT COUNTED IN TOTRUN
          TOTRUN(J)=TOTRUN(J)+NN
          INCR=-INCR
418       MM=MM+INCR
420       CONTINUE
450    CONTINUE
C DONE SWEEPING, NOW DO TARGET TRAFFIC
       J=JMAX
       DO 535 T=1,NTRK(J)
       CALL NORMAL(NTARG,NAVERR,SEED)
C NOTE EVERY TARGET HAS ITS OWN NAV ERROR
       DO 500 NNNN=1,NTARG
       NAVERR(NNNN)=X(T,J)+NAVERR(NNNN)*SIG(J)
500    LIVE(NNNN)=1
       DO 530 M=1,MINES
          IF(STATUS(M).NE.0)GO TO 530
          I=MINTYP(M)
          A2=A(I,J)
          IF(A2.EQ.0)GO TO 530
          A1=A2*(1+TR)
          A2=A2*(1-TR)
          DO 525 NNNN=1,NTARG
             IF(LIVE(NNNN).EQ.0)GO TO 525
             DIST=ABS(XM(M)-NAVERR(NNNN))
             IF(DIST.LT.A1)THEN
                IF(DIST.LT.A2)THEN
                   B1=B(I,J)
                ELSE
                   B1=B(I,J)*(DIST-A2)/(A1-A2)
                ENDIF
                IR=IR+1
                IF(IR.GE.O3M1)THEN
                   CALL RAND(O3,U,SEED)
                   IR=1
                ENDIF
                IF(U(IR).LT.B1)THEN
                   SWPT(I,J)=SWPT(I,J)+1
                   IF(DIST.LT.AF(I,J))THEN
                      IR=IR+1
                      IF(U(IR).LT.BF(I,J))THEN
                         THRT(NNNN)=THRT(NNNN)+1
                         LIVE(NNNN)=0
                      ENDIF
                   ENDIF
                   GO TO 530
                ENDIF
             ENDIF
525       CONTINUE
530    CONTINUE
```

```
535   CONTINUE
539   CONTINUE
      IF(REPMAX.GT.3)WRITE(6,*)
      REPMAX=REPMAX*100
      DO 550 I=1,IMAX
      DO 540 J=1,JM1
      SWPT(I,JMAX)=SWPT(I,JMAX)-SWPT(I,J)
540   SWPT(I,J)=SWPT(I,J)/REPMAX
      SWPT(I,JMAX)=SWPT(I,JMAX)/REPMAX
      DO 545 K=1,KMAX
         CAS(I,K)=CAS(I,K)/REPMAX
         CASTOT(K)=CASTOT(K)+CAS(I,K)
545   CONTINUE
550   CONTINUE
      DO 553 J=1,JM1
      R=0
      DO 552 T=1,NTRK(J)
552   R=R+RUNS(T,J)
      R=R*NP(J)
      IF(R.GT.0)THEN
         TOTRUN(J)=100.*TOTRUN(J)/(R*REPMAX)
      ELSE
         TOTRUN(J)=9999999.
      ENDIF
553   CONTINUE
      WRITE(6,*)'UNIT RUNS BY SWEEP TYPE, RELATIVE TO NUMBER PLANNED (**
     +**** MEANS N/A):'
      WRITE(6,560)(NS(J),J=1,JM1)
      WRITE(6,590)'PERCENT',(TOTRUN(J),J=1,JM1)
      DO 555 NNNN=1,NTARG
555   THRT(NNNN)=THRT(NNNN)/(REPMAX*NTRK(JMAX))
      WRITE(6,*)'AVERAGE NUMBER OF MINES SWEPT BY SWEEP TYPE'
      WRITE(6,560)(NS(J),J=1,JM1),' UNSWEPT'
560   FORMAT(8X,9A8)
      DO 570 J=1,JMAX
570   TOTRUN(J)=0
      DO 580 I=1,IMAX
      DO 575 J=1,JMAX
575   TOTRUN(J)=TOTRUN(J)+SWPT(I,J)
      WRITE(6,590)NM(I),(SWPT(I,J),J=1,JMAX)
580   CONTINUE
      WRITE(6,590)'TOTAL ',(TOTRUN(J),J=1,JMAX)
      WRITE(6,*)'RESOURCE LOSSES TO MINE TYPE'
      WRITE(6,560)(NR(K),K=1,KMAX)
      DO 585 I=1,IMAX
      WRITE(6,590)NM(I),(CAS(I,K),K=1,KMAX)
585   CONTINUE
      WRITE(6,590)'TOTAL ',(CASTOT(K),K=1,KMAX)
      WRITE(6,600)(THRT(NNNN),NNNN=1,NTARG)
590   FORMAT(1X,A7,9F8.2)
600   FORMAT(' SEQUENTIAL THREAT:',10F5.3)
      TMP=0.
      DO 610 NNNN=1,NTARG
610   TMP=TMP+THRT(NNNN)
      WRITE(6,620)NTARG,TMP
620   FORMAT(' AVERAGE LOSSES OUT OF',I3,' TRANSITORS:',F7.4)
      WRITE(6,*)' ENTER NUMBER OF NEW REPLICATIONS IN HUNDREDS, OR 0 FOR
     + MAIN MENU'
      READ(5,*)REPMAX
```

```
      IF(REPMAX.GT.0)GO TO 10
      END

      SUBROUTINE KATZ(
     I                U,ALPHA,BETA,Q,MAX
     O                ,M)
C GENERATES A KATZ SAMPLE BASED ON THE RANDOM NUMBER U
      IMPLICIT NONE
      REAL*4 ALPHA,BETA,P,NUM,SUM,Q
      INTEGER M,MAX,U
      P=Q
      SUM=P
      M=0
      NUM=ALPHA
10    IF(SUM.LT.U.AND.M.LT.MAX.AND.NUM.GT.0)THEN
         M=M+1
         P=P*NUM/M
         SUM=SUM+P
         NUM=NUM+BETA
         GO TO 10
      ENDIF
      END

      SUBROUTINE RAND(
     I                N
     O                ,U
     U                ,SEED)
C SCHRAGE'S METHOD, EXCEPT U IS BETWEEN 1 AND MAXINT
      INTEGER*4 A,MAXINT,Q,R,SEED,HI,LO,TEST,I,N
      INTEGER*4 U(N)
      DATA A,MAXINT,Q,R /16807,2147483647,127773,2836/
      DO 20 I=1,N
      HI=SEED/Q
      LO=SEED-HI*Q
      TEST=A*LO-R*HI
      IF(TEST.GT.0)THEN
         SEED=TEST
      ELSE
         SEED=TEST+MAXINT
      ENDIF
      U(I)=SEED
20    CONTINUE
      END

      SUBROUTINE NORMAL(
     I                  N
     O                  ,X
     U                  ,SEED)
C SUMS 12 UNIFORMS TO GET A NORMAL
      IMPLICIT NONE
      INTEGER*4 I,J,N,SEED,MAXINT
      INTEGER*4 U(12)
      REAL*4 X(N)
      DATA MAXINT /2147483647/
      DO 10 I=1,N
      CALL RAND(12,U,SEED)
      X(I)=0
      DO 5 J=1,12
5     X(I)=X(I)+U(J)
```

```
      X(I)=X(I)/MAXINT-6.
10    CONTINUE
      END
```

7.4 Subroutine REHEARSE in file REHEARSE.FOR.

This subroutine is essentially a single replication of the simulation, except that the updated Katz parameters and surviving resources are calculated and output to NEWNUM.DAT. After renaming NEWNUM.DAT to NUMBERS.DAT and possibly changing the minesweeping plan, the user might then consider the next simulated phase of minesweeping.

Since there is only one replication of the simulation, a user-friendly version of this subroutine should give a graphical representation of the minesweeping.

```
      SUBROUTINE REHEARSE(
     I                    X,RUNS,NTRK,P,SURV
     U                    ,NU,KZA,KZB,SEED)
C THIS SUBROUTINE DOES A SINGLE REPLICATON AS IN MONTE, BUT ADDS
C UPDATES OF THE KATZ PARAMATERS AND NU SO DUMPIT CAN WRITE THEM
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O3,O4,O5,NTARG,REPLY
      INCLUDE 'LIMITS.DAT'
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR
     +   ,BF(O1,O2),HOUR(O2),XM(O3),SIG(O2),VAL(O4),SURV(O1,O2)
     +   ,DIST,POS,CAS(O1,O2),SWPT(O1,O2),A1,A2,B1,P(O1),HT
     +   ,ACT(O1),TMP,Q(O1),CLR(O1),TOTRUN(O2),NAVERR(NTARG)
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,I,J,JP,K,R,JM1,NUM(O1),MINES,MINTYP(O3)
     +   ,RUNS(O5,O2),NTRK(O2),T,LO,M,STATUS(O3),IND(O2),SEQ(O2)
     +   ,INCR,MM,NN,NNN,NNNN,H(O4,O2),NU(O4),NUN(O4),O3M1
     +   ,X(O5,O2),MAXINT,U(O3),SEED,DEAD(NTARG)
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +      ,SEQ,IMAX,JMAX,KMAX,IND,H
     +      ,NM,NS,NR
      DATA MAXINT /2147483647/
      JM1=JMAX-1
      O3M1=O3-1
10    DO 130 I=1,IMAX
      DO 124 J=1,JM1
124   SWPT(I,J)=0
      DO 125 K=1,KMAX
125   CAS(I,K)=0
130   CONTINUE
      DO 135 J=1,JM1
135   TOTRUN(J)=0
      DO 140 K=1,KMAX
140   NUN(K)=NU(K)
      CALL RAND(IMAX,U,SEED)
```

```
      MINES=0
      DO 170 I=1,IMAX
      CALL KATZ(U(I),KZA(I),KZB(I),P(I),O3,NUM(I))
      MINES=MINES+NUM(I)
      IF(MINES.GT.O3)THEN
         WRITE(6,*)' O3 NOT BIG ENOUGH',I,NUM(I)
         MINES=O3
      ENDIF
      CLR(I)=0
170   CONTINUE
      CALL RAND(MINES,U,SEED)
      TMP=WIDTH/MAXINT
      DO 180 M=1,MINES
180   XM(M)=TMP*U(M)
C PUT THE MINES IN RANDOM ORDER
      CALL RAND(MINES,U,SEED)
      LO=MINES
182   IF(LO.EQ.0)GO TO 187
      J=1+MOD(U(LO),LO)
      K=0
      DO 185 I=1,IMAX
         K=K+NUM(I)
         IF(J.LE.K)THEN
            MINTYP(LO)=I
            STATUS(LO)=0
            LO=LO-1
            NUM(I)=NUM(I)-1
            GO TO 182
         ENDIF
185   CONTINUE
187   CONTINUE
      DO 450 JP=1,JM1
         J=SEQ(JP)
         INCR=1
         MM=1
            NN=999
            DO 190 K=1,KMAX
            IF(H(K,J).NE.0)NN=MIN(NN,NUN(K))
190         CONTINUE
            IF(NN.EQ.0)GO TO 450
         DO 420 T=1,NTRK(J)
         CALL NORMAL(1,TMP,SEED)
         POS=X(T,J)+TMP*SIG(J)
         DO 418 R=1,RUNS(T,J)
            DO 400 M=1,MINES
               IF(STATUS(MM).NE.0)GO TO 400
               I=MINTYP(MM)
               A2=A(I,J)
               IF(A2.EQ.0)GO TO 400
               A1=A2*(1+TR)
               A2=A2*(1-TR)
               DIST=ABS(XM(MM)-POS)
               IF(DIST.LT.A1)THEN
                  IF(DIST.LT.A2)THEN
                     B1=B(I,J)
                  ELSE
                     B1=B(I,J)*(DIST-A2)/(A1-A2)
                  ENDIF
                  NNN=NN
```

```
                    DO 220 NNNN=1,NNN
                    IF(STATUS(MM).NE.0)GO TO 400
                    CALL RAND(1,U,SEED)
                    IF(U(1).LT.B1)THEN
                       STATUS(MM)=J
                       CLR(I)=CLR(I)+1
                       SWPT(I,J)=SWPT(I,J)+1
                       IF(DIST.LT.AF(I,J))THEN
                          CALL RAND(1,U,SEED)
                          IF(U(1).LT.BF(I,J))THEN
                                DO 210 K=1,KMAX
                                   IF(H(K,J).EQ.2)THEN
                                       CAS(I,K)=CAS(I,K)+1
                                       IF(NN.EQ.NUN(K))NN=NN-1
                                       NUN(K)=NUN(K)-1
                                   ENDIF
210                             CONTINUE
                          ENDIF
                       ENDIF
                    ENDIF
220                 CONTINUE
C                   IF NO UNITS OF SWEEP LEFT, GO ON TO NEXT SWEEP TYPE
                    IF(NN.EQ.0)GO TO 450
                 ENDIF
400           MM=MM+INCR
           TOTRUN(J)=TOTRUN(J)+NN
           INCR=-INCR
418        MM=MM+INCR
420        CONTINUE
450     CONTINUE
C CALCULATE CLEARANCE LEVELS Q(I) BASED ON TOTRUN(J)
        DO 470 I=1,IMAX
           Q(I)=1.
           DO 460 J=1,JM1
460        IF(SURV(I,J).LT.1.)Q(I)=Q(I)*SURV(I,J)**TOTRUN(J)
470     CONTINUE
C DONE SWEEPING, NOW DO TARGET TRAFFIC
        J=JMAX
        CALL RAND(1,U,SEED)
        TMP=(U(1)-1)/MAXINT
C T IS A RANDOMLY SELECTED TARGET TRACK
        T=1+TMP*NTRK(J)
        CALL NORMAL(NTARG,NAVERR,SEED)
        DO 500 NNNN=1,NTARG
        NAVERR(NNNN)=X(T,J)+NAVERR(NNNN)*SIG(J)
500     DEAD(NNNN)=0
        DO 530 M=1,MINES
           IF(STATUS(M).NE.0)GO TO 530
           I=MINTYP(M)
           A2=A(I,J)
           IF(A2.EQ.0)GO TO 530
           A1=A2*(1+TR)
           A2=A2*(1-TR)
           DO 525 NNNN=1,NTARG
              IF(DEAD(NNNN).EQ.1)GO TO 525
              DIST=ABS(XM(M)-NAVERR(NNNN))
              IF(DIST.LT.A1)THEN
                 IF(DIST.LT.A2)THEN
                    B1=B(I,J)
```

```
                  ELSE
                     B1=B(I,J)*(DIST-A2)/(A1-A2)
                  ENDIF
                  CALL RAND(1,U,SEED)
                  IF(U(1).LT.B1)THEN
                     IF(DIST.LT.AF(I,J))THEN
                        CALL RAND(1,U,SEED)
                        IF(U(1).LT.BF(I,J))THEN
                           DEAD(NNNN)=1
                        ENDIF
                     ENDIF
                     GO TO 530
                  ENDIF
               ENDIF
525         CONTINUE
530      CONTINUE
         WRITE(6,*)' COMPLETED UNIT RUNS BY SWEEP TYPE'
         WRITE(6,560)(NS(J),J=1,JM1)
         WRITE(6,590)'TOTRUN',(TOTRUN(J),J=1,JM1)
         WRITE(6,*)' CLEARANCE LEVELS BASED ON COMPLETED RUNS...'
         WRITE(6,540)(NM(I),I=1,IMAX)
540      FORMAT(2X,9A8)
         WRITE(6,550)((1.-Q(I)),I=1,IMAX)
550      FORMAT(10F8.4)
         WRITE(6,*)' NUMBER OF MINES SWEPT BY SWEEP TYPE'
         WRITE(6,560)(NS(J),J=1,JM1),' TOTAL'
560      FORMAT(7X,9A8)
         DO 561 J=1,JMAX
561      TOTRUN(J)=0
         DO 563 I=1,IMAX
         SWPT(I,JMAX)=0
         DO 562 J=1,JM1
         TOTRUN(J)=TOTRUN(J)+SWPT(I,J)
562      SWPT(I,JMAX)=SWPT(I,JMAX)+SWPT(I,J)
563      TOTRUN(JMAX)=TOTRUN(JMAX)+SWPT(I,JMAX)
         DO 565 I=1,IMAX
         WRITE(6,590)NM(I),(SWPT(I,J),J=1,JMAX)
565      CONTINUE
         WRITE(6,590)'TOTAL ',(TOTRUN(J),J=1,JMAX)
         WRITE(6,*)'RESOURCE LOSSES TO MINE TYPE'
         WRITE(6,560)(NR(K),K=1,KMAX)
         DO 570 I=1,IMAX
         WRITE(6,590)NM(I),(CAS(I,K),K=1,KMAX)
570      CONTINUE
         WRITE(6,600)(DEAD(NNNN),NNNN=1,NTARG)
590      FORMAT(1X,A6,9F8.2)
600      FORMAT(' SEQUENTIAL THREAT: ',10I4)
         WRITE(6,*)' ENTER 0 TO STOP, OR 1 TO REPEAT WITH NEW SEED:'
         READ(5,*)REPLY
         IF(REPLY.GT.0)GO TO 10
C  UPDATE KZA, KZB, NU, AND RETURN
         DO 608 I=1,IMAX
            KZB(I)=KZB(I)*Q(I)
            KZA(I)=MAX(0.,Q(I)*KZA(I)+KZB(I)*CLR(I))
608      CONTINUE
         DO 620 K=1,KMAX
620      NU(K)=NUN(K)
         END
```

7.5 Subroutine REALITY in file REALITY.FOR.

This subroutine is similar to REHEARSE except that results are input by the user, rather than simulated. No random numbers are employed. An operational version would keep complete records, communicate status to other units, etc.

```
      SUBROUTINE REALITY(
     O                   NU
     U                   ,KZA,KZB)
      IMPLICIT NONE
C LIMITS ON MINE TYPES, SWEEP TYPES, MINES, RESOURCES, TRACKS
      INTEGER*4 O1,O2,O3,O4,O5,NTARG
      INCLUDE 'LIMITS.DAT'
      REAL*4 KZA(O1),KZB(O1),A(O1,O2),B(O1,O2),AF(O1,O2),WIDTH,TR
     +   ,BF(O1,O2),HOUR(O2),SIG(O2),VAL(O4),HT
     +   ,ACT(O1),TMP,Q(O1),CLR(O1),TOTRUN(O2)
      CHARACTER NM(O1)*7,NS(O2)*8,NR(O4)*7
      INTEGER*4 IMAX,JMAX,KMAX,I,J,K,JM1
     +   ,IND(O2),SEQ(O2)
     +   ,H(O4,O2),NU(O4),MAXINT
      COMMON A,B,AF,BF,HOUR,ACT,WIDTH,TR,SIG,VAL,HT
     +       ,SEQ,IMAX,JMAX,KMAX,IND,H
     +       ,NM,NS,NR
      DATA MAXINT /2147483647/
      JM1=JMAX-1
      WRITE(6,*)' FOR EACH SWEEP TYPE, ENTER THE TOTAL NUMBER OF RUNS TH
     +IS PERIOD'
      WRITE(6,*)' EVERY PASS OF A SWEEPING UNIT THROUGH THE MINEFIELD IS
     + A RUN'
      WRITE(6,*)' IF 2 RUNS PLUS 60% OF A THIRD WERE COMPLETED, ENTER 2.
     +6'
      DO 135 J=1,JM1
      WRITE(6,10)J,NS(J)
10    FORMAT(' TYPE ',I3,' (',A8,'):')
      READ(5,*)TOTRUN(J)
135   CONTINUE
      WRITE(6,*)' FOR EACH MINE TYPE, INPUT THE NUMBER SWEPT THIS PERIOD
     + (AN INTEGER)'
      DO 140 I=1,IMAX
      WRITE(6,137)I,NM(I)
137   FORMAT(' TYPE',I3,' (',A7,'):')
      READ(5,*)CLR(I)
140   CONTINUE
      WRITE(6,*)' FOR EACH RESOURCE TYPE, GIVE THE NUMBER AVAILABLE FOR
     + THE NEXT PERIOD'
      DO 220 K=1,KMAX
      WRITE(6,210)K,NR(K)
210   FORMAT(' TYPE',I3,' (',A8,'):')
      READ(5,*)NU(K)
220   CONTINUE
C UPDATE KATZ PARAMETERS
      DO 455 I=1,IMAX
      TMP=0
      DO 452 J=1,JM1
452   TMP=TMP+TOTRUN(J)*A(I,J)*B(I,J)
```

```
      TMP=TMP/(WIDTH*MAXINT*.5)
C NONCLEARANCE PROBABILITY FOR TYPE I
      IF(TMP.LT.20)THEN
         Q(I)=EXP(-TMP)
         KZB(I)=KZB(I)*Q(I)
         KZA(I)=Q(I)*KZA(I)+KZB(I)*CLR(I)
      ELSE
         KZA(I)=0.
      ENDIF
455   CONTINUE
      END
```

7.6 The file LIMITS.DAT.

This file is included in other programs using the FORTRAN INCLUDE compiler directive. The purpose is simply to avoid having to make multiple changes of array size limitations.

```
      PARAMETER(O1=10,O2=9,O3=999,O4=8,O5=40,NTARG=10)
```

The prototype will accept at most 10 mine types, 9 sweep types (including targets), and 8 resource types. The number of mines in the simulation is limited to 999, and the number of tracks for any sweep type is limited to 40. The number of targets is 10.

8. Further development.

MIXER is a usable TDA in its current form, but only in the hands of an expert. Section 8.1 discusses improvements that would make MIXER easier to use, and section 8.2 discusses extensions.

8.1 Useability improvements.

Any TDA for mixed minefield countermeasures by multiple resources should function as part of a C4I system with links to a database, since there are bound to be lots of parameters required. Ideally, a user would specify the minefield's location, state the name of the responsible country and see a list of the mine types and inventories possessed by that country. He would then select appropriate sweep types from a list, thereby determining all of the A/B data. This is most of the data that is currently in

PARAMS.DAT. Mine inventory information would seem to be essential if the user is to give a distribution of the number of mines of each type that are present, as MIXER or any other TDA with MIXER's functions must require. The simplest default would be to make the distribution of each type Poisson ($\beta = 0$), with the mean being some fraction of the inventory. At present, even an "expert" attempting to use MIXER would be handicapped by the current lack of easily accessible inventory information. The first priority for mixed minefield analysis should be to aid or automate the construction of the input files that MIXER requires. The prototype MIXER establishes that something useful can be done with those inputs, but does nothing to ease their acquisition.

MIXER could benefit from better prompts, better error checking, and some graphics. The most important need for graphics is in REHEARSE, where the location and fate of every mine and sweeping unit could be represented on a chart. Such a representation could go a long way toward making intuitive some of the concepts that are otherwise necessarily abstract. Another good graphic would be a display of the Katz probability distribution, with slide bars for μ and σ that the user could adjust. If MIXER is ultimately made part of a C4I system, then it should have a Graphical User Interface (GUI) that respects the system's conventions.

While MIXER can certainly be made more user-friendly, the mine countermeasures problem cannot be made simple. Any TDA such as MIXER with a Bayesian view of decision making must require certain "inputs" that are actually decisions made in secret by the enemy. Probability actuator settings are an explicit example in MIXER, but mine sensitivity settings are also required because some judgment about sensitivity is required to determine the A/B data. The user of MIXER is essentially trying to guess what was in the mind of the enemy when the minefield was laid, information that will not be found in any database.

8.2 Enhancements and revisions.

8.2.1 Non-uniform optimization.

MIXER can make a Monte Carlo assessment of any input clearance plan, uniform or not, but the optimization feature will always produce a uniform plan. If traffic is expected to concentrate in one part of the minefield, it might be worthwhile to permit nonuniform plans in the optimization. This would be a harder optimization problem, but it is still well defined and no new data are required. There would be an impact on the Katz assumption, since nonuniform clearance is not a simple sample-and-subtract operation.

8.2.2 Improved optimization.

In the prototype, a user who wants to consume exactly a fixed number of hours must manipulate the cost of time until clearance requires that time. Such a user would prefer to state the time limit as a constraint and avoid the manipulation. There might also be constraints on SIT or on various kinds of resource loss. The resulting constrained optimization problem has a nonlinear objective function similar to the one in COGNIT [6]. However, the problem is more difficult than the one solved by COGNIT on account of the need to deal with multiple resources, so solution by exhaustion will not be possible except on small problems. Even so, improvements as discussed in section 6 should be possible.

8.2.3 Inclusion of mine counters.

MIXER includes only the geometric distribution, in the form of an actuation probability. It would not be hard to include a general distribution in the Monte Carlo simulation — all that is required is to randomly assign a count to each mine as it is generated, decrementing the count with each actuation until detonation occurs. Solving the optimization problem would be more difficult, however, and the idea of clearance as a succession of similar stages would be more difficult to implement.

8.2.4 Multi-segment minefields.

MIXER's assumption that sweep types must enter a minefield sequentially has some strong implications. Clearance times are surprisingly large, since different sweep types cannot operate simultaneously. Also, there is a tendency for ships to have nothing to do until helicopters are done, and for helicopters to have nothing to do once the ships start. If the minefield were big enough to be segmented so that different sweep types could safely function in parallel as long as they stay in different segments, then clearance could be considerably faster. An optimal plan might have helicopters merely clear mines that are dangerous to ships in one segment, afterwards spending all of their time in a segment that ships never enter. The optimization problem could still be formatted abstractly and solved approximately.

8.2.5 Game formulation.

Instead of requiring inputs for mine counts and sensitivity, one might formulate the problem as a game where the miner controls those quantities and has the objective of maximizing the same total loss that the clearance force attempts to minimize. The attractive feature is that the mine count and sensitivity inputs would no longer be required. However, game formulations have their own problems, including difficult mathematics and the possibility of optimized plans that involve randomization. The current consensus seems to be that two-sided formulations such as Breakthrough [7], while useful in systems studies, are difficult to turn into TDAs.

8.2.6 Better damage model.

The AF(I,J) and BF(I,J) arrays should actually have an additional index K, or possibly replace J by K. In the prototype, a detonation is either lethal or not, and a lethal detonation is lethal for all resources K with $H(K,J) > 1$. In reality, different resources might simply have different damage distances or probabilities.

REFERENCES

[1] Johnson, N., and S. Kotz. 1969. *Discrete Distributions*, Wiley, New York, pp. 37-43.

[2] Washburn, A. 1994. "Mine Warfare Models," Naval Postgraduate School course notes.

[3] Sutter, F., and J. Lancaster. 1982. "Application of Statistical Decision Theory to the Mine Countermeasure Exploratory Mission," Proceedings of the 49th MORS, pp. 75-86.

[4] Park, S., and K. Miller. 1988. "Random Number Generators: Good Ones and Hard to Find," *Comm. ACM 31*, **10**, pp. 1192-1201.

[5] Washburn, A. 1995. "Branch and Bound Methods for Search Problems," Naval Postgraduate School Technical Report, NPSOR-95-003, Monterey, CA 93943.

[6] CINCPAC. 1988. "A Cognitive Planning Aid for Naval Minesweeping Operations," Strategic Planning and Policy Directorate, Research and Analysis Division, Camp Smith, Hawaii.

[7] Sutter, F. 1983. "Mine Warfare Trainer – Mathematical Model Report for the Breakthrough MCM Objective (revision A)," Naval Coastal Systems Center TM216-77.

DISTRIBUTION LIST

1. Research Office (Code 08) 1
   Naval Postgraduate School
   Monterey, CA 93943-5000

2. Dudley Knox Library (Code 52) 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Department of Operations Research 1
   Editorial Assistant (Code OR/Bi)
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Prof. Alan R. Washburn (Code OR/Ws) 5
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. NAVTACSUPPACT 1
   Washington Navy Yard
   Building 200
   901 M St. SE
   Washington, DC 20374

6. COMINEWARCOM (N02R) 2
   325 Fifth St. SE
   Corpus Christi, TX 78419

7. Coastal Systems Station 3
   6703 W. Highway 98
   Panama City, FL 32407-7001

8. Center for Naval Analyses 1
   4401 Ford Avenue
   P.O. Box 16268
   Alexandria, VA 22302-0268

9. Office of Naval Research (322TE) 1
   800 North Quincy Street
   Arlington, VA 22217-5660

10. Commander in Chief 1
    U.S. Pacific Command
    Attn: J53 (Mr. McCurdy)
    Camp H.M. Smith, HI 96861-4015