

ARMY RESEARCH LABORATORY



Predicting Fragmentation Propagation Probabilities for Ammunition Stacks

John Starkenberg
Kelly J. Benjamin
Robert B. Frey

ARL-TR-949

January 1996

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

19960215 011

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project(0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE January 1996		3. REPORT TYPE AND DATES COVERED Final, February - December 1994
4. TITLE AND SUBTITLE Predicting Fragmentation Propagation Probabilities for Ammunition Stacks			5. FUNDING NUMBERS 4G031-412-T5	
6. AUTHOR(S) John Starkenberg, Kelly J. Benjamin, and Robert B. Frey				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory ATTN: AMSRL-WT-TB Aberdeen Proving Ground, MD 21005-5066			8. PERFORMING ORGANIZATION REPORT NUMBER ARL-TR-949	
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) By combining several existing models, we have developed a tool for estimating the probabilities associated with the propagation of detonation or burning between ammunition stacks. The models include the FRAGHAZ program for the Monte Carlo treatment of fragment trajectories and the accumulation of hit probabilities, the Jacobs-Roslund criterion for initiation of detonation and the ballistic limit condition for initiation of burning. We have applied this tool to artillery ammunition and missile stacks. Since the appropriate fragmentation input data was not always available, notably in the case of missiles, we developed methods of estimating this data. Single artillery round donors were shown to require a near-direct hit in order to initiate detonation in either artillery ammunition or missile acceptor stacks. Artillery ammunition donor stacks were shown to be much more lethal than missile donor stacks, and missile acceptor stacks were shown to be more vulnerable to the propagation of burning than artillery ammunition acceptor stacks.				
14. SUBJECT TERMS fragments, ammunition fragments, en masse detonation, quantity-distance			15. NUMBER OF PAGES 79	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

		<u>Page</u>
	LIST OF FIGURES	v
	LIST OF TABLES	vii
1.	BACKGROUND	1
2.	APPROACH	1
3.	FRAGHAZ	2
4.	FRAGPROP	4
5.	DONOR MODELS	6
5.1	Stack Description	6
5.2	Arena Test Fragmentation Data	6
5.3	Adapting Arena Test Fragmentation Data	7
5.4	Estimating Fragmentation Data	8
6.	ACCEPTOR MODELS	10
6.1	Stack Description	10
6.2	Vulnerable Areas	10
6.3	Detonation Vulnerability	12
6.4	Burning Vulnerability	13
6.5	Comparison of Detonation and Burning Thresholds	14
6.6	Vulnerable Area Reduction	15
6.7	Mechanical Damage Vulnerability	16
7.	CONTAINER PENETRATION	18
8.	MINIMUM RANGE	18
9.	COMPUTATIONAL CONFIGURATIONS AND RESULTS	18
9.1	Theater of Operations Considerations	18
9.2	Stack Arrangements	19
9.3	Propagation Probabilities and Distances	20
10.	SUMMARY AND CONCLUSIONS	27
11.	REFERENCES	29

	<u>Page</u>
APPENDIX A: PROPAGATION ACCIDENTS IN AMMUNITION HOLDING AREAS	31
APPENDIX B: FRAGPROP LISTING	35
APPENDIX C: LETHALITY AND VULNERABILITY DATA	77
APPENDIX D: DONOR FRAGMENTATION GENERATOR LISTING	81
DISTRIBUTION LIST	89

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. FRAGHAZ representation of a donor stack showing the stack axis and hazard volume (from NSWC TR 87-59)	3
2. FRAGHAZ representation of target (acceptor) vulnerability and algorithm for determining hazard probability (from NSWC TR 87-59)	4
3. FRAGHAZ/FRAGPROP flowchart	5
4. Comparison of critical velocities for detonation and burning	15
5. Angular region of vulnerability on a cylindrical charge	16
6. Relationship between the angular region of vulnerability and the maximum obliquity for horizontal storage	17
7. Relationship between the angular region of vulnerability, the maximum obliquity, and the fragment elevation angle for vertical storage	17
8. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for an M107 donor stack against a TOW-2A acceptor stack	21
9. Probabilities of burning, mechanical damage, and hit as functions of range for single M107 donor projectiles against M107 acceptor stacks	23
10. Probabilities of burning, mechanical damage, and hit as functions of range for single M107 donor projectiles against TOW-2A acceptor stacks	23
11. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for M107 donor stacks against M107 acceptor stacks	24
12. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for M107 donor stacks against TOW-2A acceptor stacks	24
13. Probabilities of mechanical damage and hit as functions of range for TOW-2A donor stacks against M107 acceptor stacks	25
14. Probabilities of burning, mechanical damage, and hit as functions of range for TOW-2A donor stacks against TOW-2A acceptor stacks	25

INTENTIONALLY LEFT BLANK.

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Lethality and Vulnerability Parameters for M107 Stacks	20
2. Lethality and Vulnerability Parameters for TOW-2A Stacks	20
3. FRAGPROP Computation Input Conditions	22
4. 1% Propagation Probability Distances (ft)	27
A-1. Summary of Representative Propagation Accident Reports	34
C-1. Weapon Dimensions and Materials	79
C-2. Energetic Material Performance Parameters	80
C-3. Jacobs-Roslund Constants	80
C-4. THOR Velocity Equation Constants	80
C-5. THOR Mass Equation Constants	80

INTENTIONALLY LEFT BLANK.

1. BACKGROUND

In conjunction with a study of the benefits of insensitive munition (IM) technology, a need arose to develop a methodology for prediction of the probabilities of propagation of reaction (detonation or burning) and mechanical damage between a detonating stack of ammunition and its neighbors as functions of the distance between them. Such a capability could provide input to further analyses to predict losses in a variety of combat scenarios.

An informal survey of propagation accident reports (summarized in Appendix A) indicates that the most common mechanism of reaction propagation involves ignition of fires (often in combustible packaging) by fragments, debris, or firebrands from the source explosion and subsequent violent reaction of munitions in those fires. The resulting chain of events may take hours or even days to unfold. This scenario appears to be too complex and variable to model at the present time.

In some other cases, fragments from the source (donor) explosion can promptly damage, ignite mild to violent burning in, or detonate the energetic components (not including any combustible packaging) of munitions in nearby (acceptor) stacks. When detonation results, reaction can propagate further by the same mechanism, rapidly consuming large quantities of ammunition. When only burning or mechanical damage results, the acceptor stack may be totally or partially destroyed but no further propagation by the fragment mechanism ensues. This scenario is amenable to modeling.

In order to span the range of munition vulnerability, we wanted to obtain predictions applicable to typical (thick-walled) artillery projectiles and (thin-walled) missiles stacked on pallets. We also wanted to obtain predictions for a single artillery projectile donor representing an attacking munition which might be required to start the donor-acceptor chain in an analysis. We have chosen palletized and single M107 155-mm projectiles and palletized TOW-2A missiles as representative items.

2. APPROACH

We determined that the following elements are required to successfully model propagation among stacks of these items:

- descriptions of the stack storage arrangements

- arena test data for donor stack munitions describing the initial fragment mass, velocity and shape distributions, or estimates of such data
- a treatment of fragment trajectories to determine hazard probabilities
- descriptions of the vulnerable components of the acceptor stack munitions
- criteria for the initiation of detonation and burning as well as for mechanical damage.

The pallet arrangements commonly used for the weapons of interest have been determined. These form the building blocks for larger stacks.

Arena test data is generally available only for warheads whose performance is measured by fragmentation statistics. That is, data is available for the M107 projectile but not for the shaped-charge warhead and rocket motors found in the TOW-2A. We had to develop a method to estimate the data for the latter items.

We decided to modify an existing computer program called FRAGHAZ (McClesky 1988) in order to compute fragment trajectories and the desired probabilities. FRAGHAZ was developed to predict the hazard to a human target due to fragmentation from an exploding ammunition stack. Fragmentation data for a number of munitions is provided with the program.

Descriptions of the vulnerable components of these weapons were sometimes difficult to come by. They were obtained from a variety of sources. In some cases, best guesses were used. In all cases, the geometries were simplified to simple cylindrical metal shells filled with energetic materials.

An initiation criterion for detonation is well known and a simple criterion for burning may be developed from research data pertinent to this phenomenon.

3. FRAGHAZ

FRAGHAZ uses a fourth order Runge-Kutta scheme to compute trajectories for each fragment specified in the arena test input data. The trajectories include effects of ricochet and wind (if desired).

All the fragments are assumed to emanate from a single vertical line extending from the base to the top of munitions in the face of the stack as shown in Figure 1. This defines the stack axis. The initial fragment height is selected at random within this range. The initial fragment velocity and elevation angle are selected at random in a range near the values given in the input data. Parameters used to determine the fragment drag coefficient are determined as functions of a randomly selected value. The program replicates these trajectory computations many times with randomly selected environmental conditions. While FRAGHAZ provides "Monte Carlo" and "Full Factorial" options, the random selections referred to in the foregoing description apply only to the Monte Carlo option.

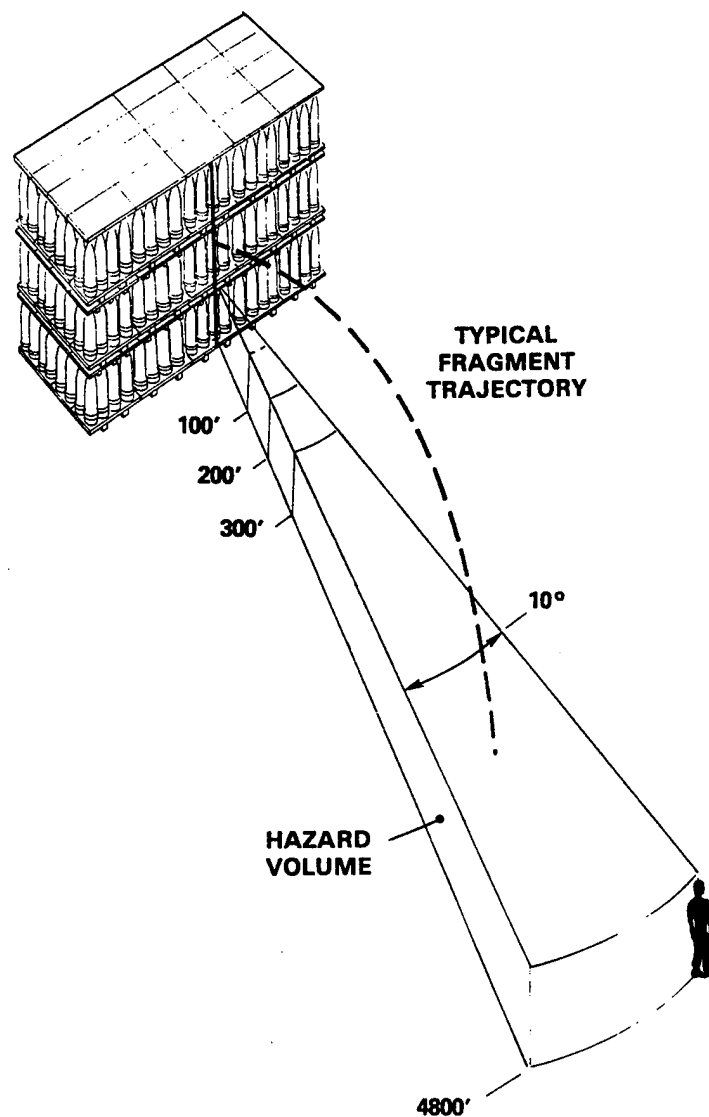


Figure 1. FRAGHAZ representation of a donor stack showing the stack axis and hazard volume (from NSWG TR 87-59).

A downrange "hazard volume," also shown in Figure 1, is defined. This consists of a cylindrical sector originating at the stack axis having a specified angular width (equivalent to the azimuthal sector associated with arena test data collection) and having the height of the target (nominally a 5.72-ft tall standing man). It is divided into a number of 100-ft annular segments. As each fragment passes through a hazard-volume segment, various hit probabilities and fragment densities are accumulated and represented as functions of the downrange distance associated with the midpoint of the segment. A determination of whether or not the fragment is hazardous based on its kinetic energy is made. The hazard hit probabilities depend on the ratio of the presented area of the target to the presented area of the hazard-volume segment with respect to hazardous fragments as shown in Figure 2.

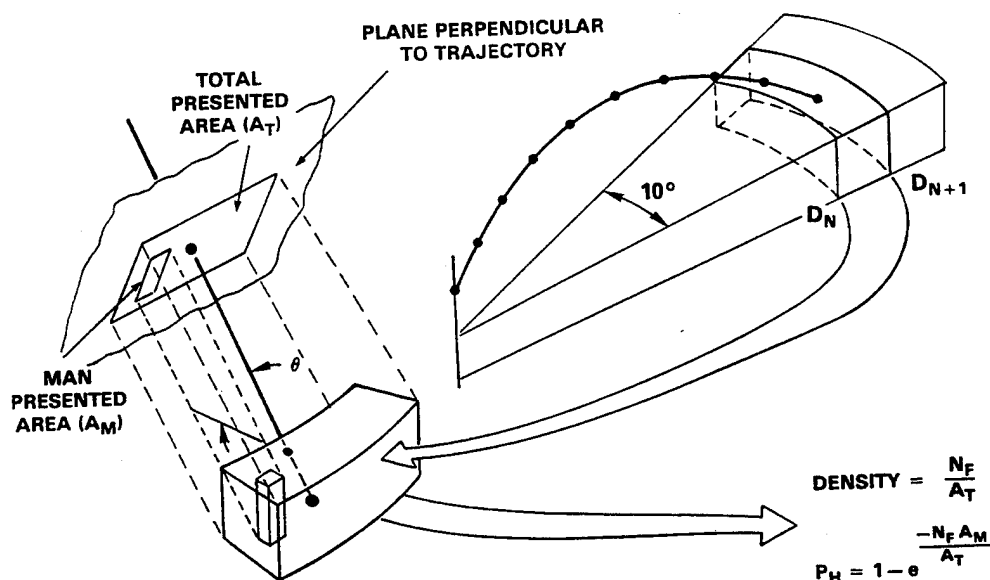


Figure 2. FRAGHAZ representation of target (acceptor) vulnerability and algorithm for determining hazard probability (from NSWC TR 87-59).

4. FRAGPROP

While its basic features were retained, the modifications to FRAGHAZ were extensive. They proceeded in two phases: the first to streamline the existing code and the second to implement the required new models. The resulting code was renamed FRAGPROP. A listing of the source code is provided in Appendix B. A flowchart applicable to both FRAGHAZ and FRAGPROP is shown in Figure 3.

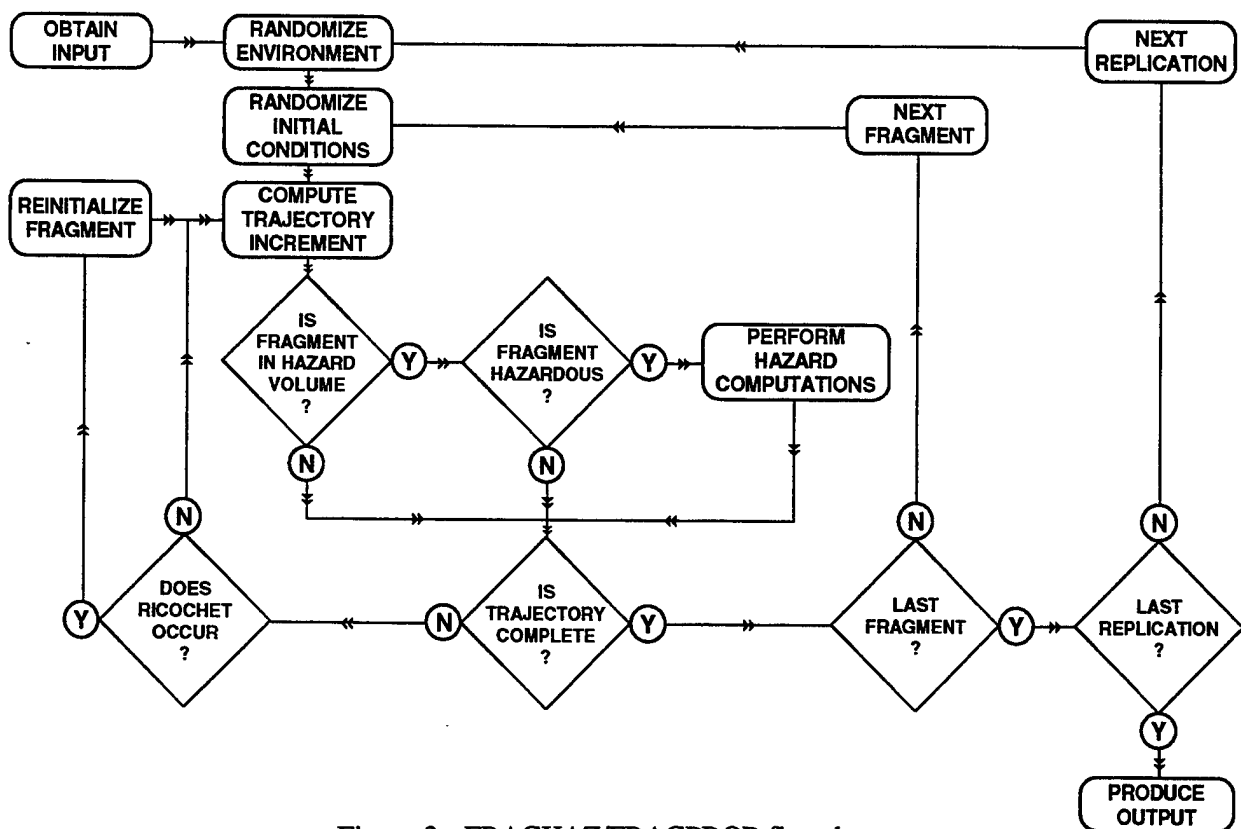


Figure 3. FRAGHAZ/FRAGPROP flowchart.

The original program made minimal use of modularization and was unwieldy to work with. We broke the code down into subroutines. As a result, the overall logic is easily followed in the main program. We changed the variable names to more descriptive forms which conform to FORTRAN naming conventions. In conjunction with this, we eliminated most of the type declarations. Many of the variables and computations were unnecessary for our purposes and we eliminated them. Some expressions, although correct, were recast in order to eliminate mixed-type operations. Elseif constructions were eliminated for clarity and less convoluted logic was substituted where possible. Execution speed was increased by moving expressions which had been repeatedly evaluated within loops to positions outside those loops and by eliminating redundant computations. For simplicity, we also eliminated the Full Factorial option.

New features and algorithms were also incorporated. Input was reorganized to read separate files for the run parameters, the donor description, and the acceptor description. Fragmentation data input was rewritten to reject fragments not meeting user-specified criteria for minimum mass and initial elevation angle. The downrange segment size (which had been fixed at 100 ft) was generalized to accept user specification. Computation of vulnerable areas for the acceptor elements and algorithms for determination of the detonation, burn, and mechanical damage lethality of each fragment including the effects of

container penetration were added. Finally, a determination and report of the minimum range at which the analysis is applicable was implemented. These algorithms are described in detail in the following sections.

5. DONOR MODELS

5.1 Stack Description. The donor stack is described as in the original FRAGHAZ code. The height of the bottom of the stack and the height of munitions in the stack are required to provide bounds for the initial height of each fragment. The size of the stack contributing to downrange fragmentation is described by specifying the number of "interaction areas" present on the downrange face of the stack. An interaction area is formed by each side-by-side pair of munitions. Thus, an arrangement of three projectiles in a row forms two interaction areas. If two such arrangements are stacked one on top of the other, four interaction areas result. However, if two such arrangements are set side by side (six rounds in a row), five interaction areas are produced. Interaction areas are considered instead of individual munitions because the effect of focusing along the plane of symmetry between the rounds augments the fragment velocities. Interaction areas are specified along with arena test data for multiple munitions. However, where data for single munitions (including estimated data) must be used, the donor unit is the single munition and the number of munitions on the downrange face specifies the stack size.

5.2 Arena Test Fragmentation Data. Arena test data forms the basis for describing the fragmentation characteristics of a particular munition. Data for interacting M107 projectiles including parameters for 215 fragments is provided with FRAGHAZ. Each fragment recovered in the test is characterized by five parameters: the polar zone of its origin, its mass, velocity, and area to mass ratio as well as the ratio of its maximum to average presented area (to account for the effects of tumbling on aerodynamic drag).

Fragment multipliers (McClesky 1988) are used to normalize the data for scaling to any stack size (i.e., any number of interaction areas or munitions) and any hazard volume azimuthal sector width. One fragment multiplier must be supplied for each polar zone represented in the data. The formula for computing the fragment multiplier for the i th polar zone, ϕ_i , is

$$M_F(\phi_i) = \frac{1}{N_I \psi_s(\phi_i)} ,$$

where N_I is the number of interaction areas (or single munitions, if appropriate) in the face of the stack from which the data was collected and $\psi_S(\phi_i)$ is the azimuthal sector width over which fragments were collected in the i th polar zone.

5.3 Adapting Arena Test Fragmentation Data. Arena test data for single munitions tabulated in the Joint Munitions Effectiveness Manual of Fragmentation Data (FM 101-62-3) is not in the FRAGHAZ format and does not provide all of the parameters required. Data for individual fragments is not provided. Rather, the mean fragment weight and integrated number of fragments for 50-grain weight increments are tabulated along with the initial velocity for each polar zone. A shape factor, K ($=0.5126 \text{ cm}^2/\text{g}^{2/3}$ for the M107), for all fragments, is given and can be used in the expression,

$$\frac{a_{f_{\text{avg}}}}{m_f} = \frac{K}{m_f^{1/3}},$$

to estimate the area to mass ratio of a fragment.

We expanded this data to generate a table of 870 individual fragments (as required by FRAGPROP) by rounding the integrated number of fragments for each polar zone and fragment mass increment to an integral value and generating table entries for that number of individual fragments. These fragments were each assigned the mean weight of the associated weight increment and the mean velocity of the associated polar zone. In doing this, it was possible to eliminate small fragments. Since the ratio of the fragments' maximum to average presented area is not given, we used the average value taken from the FRAGHAZ input data for M107 stacks.

$$\frac{a_{f_{\text{max}}}}{a_{f_{\text{avg}}}} = 1.479.$$

For this arena test, there is only one unit in the stack and the integrated number of fragments is adjusted to apply to the entire 360° azimuthal sector. Thus, the fragment multipliers are independent of polar zone ($M_F = 1/360 = 2.7778 \times 10^{-3}$).

A capability to invert the polar distribution of the donor fragmentation data was added to FRAGPROP in order to allow representation of a nose-down attacking projectile as the initial event in a propagation chain.

5.4 Estimating Fragmentation Data. The TOW-2A missile is not a fragmenting weapon and fragmentation data is not available. We needed to estimate data having a representative distribution of values of the five parameters required by the FRAGPROP input. Several analytical techniques applicable to single cylindrical munitions are available for this purpose. The fragmentation unit in this case is a single munition rather than an interaction area. Thus, focusing effects are ignored and the number of rounds is specified in lieu of the number of interaction areas. Geometric data required to represent the TOW-2A warhead and rocket motor as cylindrical components and energetic material performance data as needed for use with the following analyses are summarized in Appendix C.

The Mott equation (see Victor 1994) describes the distribution of fragment mass. The fraction, f , of fragments having a mass greater than m_f is given by

$$f(m_f) = \sqrt{\exp(-2m_f/\overline{m}_f)} ,$$

where $\overline{m}_f$ is the average fragment mass. This equation may be solved for m_f .

$$m_f(f) = -\frac{1}{2} \overline{m}_f \ln(f^2) .$$

By selecting random values of f which are evenly distributed between 0 and 1, this equation may be used to generate sets of fragment masses having the Mott distribution. The total number of fragments is given by

$$n_f = \frac{m_c}{\overline{m}_f} ,$$

where m_c , the total mass of all fragments, may be identified with the casing mass. Victor also gives the average fragment mass as

$$\overline{m}_f = \frac{A}{p_d} \sqrt{1+\mu/2} \left[\frac{h_c}{d_x} (d_x + h_c)^{3/2} \right],$$

where A is a constant, p_d is the detonation pressure, μ is the ratio of the mass of the casing to the mass of the charge, h_c is the thickness of the casing, and d_x is the diameter of the explosive or propellant charge (equal to the inside diameter of the casing). $A = 676.2 \text{ g-kbar/in}^{3/2}$ for mass in g, pressure in kbar and distance in inches. The Kamlet-Jacobs formula may be used to estimate the detonation pressure (in GPa) if it is not otherwise known.

$$p_d = 1.558 \rho_o^2 N \sqrt{MQ},$$

where ρ_o is the unreacted density in g/cm^3 , N is the number of moles of detonation product per gram of unreacted material, M is the average molecular weight of the detonation product gases, and Q is the chemical energy of the detonation reaction in cal/g .

No method exists for predicting a distribution of fragment velocities. However, the Gurney equation (see Dehn 1984) may be used to estimate a single velocity for all fragments.

$$v_f = \frac{\sqrt{2E}}{\sqrt{\mu+1/2}},$$

where $\sqrt{2E}$ is a property of the energetic material (having the units of velocity) known as the Gurney constant. Its value may be found in the literature or estimated (in milliseconds) using the equation of Kamlet and Finger.

$$\sqrt{2E} = 233 \frac{\sqrt{p_d}}{\rho_o^{0.6}},$$

for density in g/cm^3 and detonation pressure in kbar. Although this approach gives only one value of the fragment velocity, randomization over a narrow range takes place within FRAGPROP.

For estimating the area to mass ratio of a fragment, Victor (1994) gives the shape factor as $K = 0.5199 \text{ cm}^2/\text{g}^{2/3}$ for randomly shaped fragments. We are not aware of any method to estimate the ratio of a fragment's maximum to average presented area, so we assumed a fixed value of 1.5.

The TOW-2A storage orientation is horizontal in contrast to that of the M107. Thus, it is necessary to interpret the (rather narrow) polar distribution of fragments with respect to the munition axis as an azimuthal distribution with respect to the stack axis. This is assumed to produce an azimuthal sector width of only 20° . Because the missile components are treated as horizontal cylinders, the polar distribution with respect to the stack axis is assumed uniform. Since the azimuthal sector width is independent of polar zone, the fragment multiplier is a constant ($M_F = 1/20 = 0.05$).

We wrote a short program (incorporating the FRAGHAZ random number generator) to produce FRAGPROP input data for a general warhead and rocket motor combination in accordance with the foregoing algorithms and used it to generate data for 365 fragments from the TOW-2A warhead and flight motor. These components produce both aluminum and steel fragments. We neglected fragments from the launch motor, which has a small diameter. The program is listed in Appendix D.

6. ACCEPTOR MODELS

6.1 Stack Description. The basic acceptor stack description is given by providing dimensions of height (H_a), width (W_a), and depth (D_a). The storage orientation may be vertical, as with the M107, or horizontal, as with the TOW-2A. The vulnerable components of an acceptor may include a warhead, rocket motor, or both.

6.2 Vulnerable Areas. Vulnerable areas at the front and top of the stack are required. Side and back areas are excluded as the acceptor stack is assumed to face the donor. These depend on the weapon dimensions and the storage orientation (horizontal or vertical).

For the purpose of determining hit probabilities the total front and top areas of the stack are considered vulnerable. The front vulnerable area is

$$A_{fh} = H_a W_a ,$$

and the top vulnerable area is

$$A_{th} = D_a W_a .$$

For mechanical damage, the vulnerable areas are the areas presented to the front and top faces of the stack of the entire weapon (represented as a cylinder of diameter D_w and length L_w) multiplied by the number of weapons in that face of the stack (N_f or N_t). The vulnerable areas depend on the storage orientation. For vertical storage, the front vulnerable area is

$$A_{fm} = N_f D_w L_w ,$$

and the top vulnerable area is

$$A_{tm} = N_t \pi (D_w/2)^2 .$$

For horizontal storage, the front vulnerable area is unchanged.

$$A_{fm} = N_f D_w L_w ,$$

while the top vulnerable area is

$$A_{tm} = N_t D_w L_w .$$

Similarly, for detonation and burning, the *maximum* vulnerable areas are those of the energetic materials in the warhead and/or rocket motor (cylinders of diameter D_x and length L_x) presented to the front and top faces of the stack. For vertical storage, the front maximum vulnerable area is

$$A_{fx_{max}} = N_f D_x L_x ,$$

and the top maximum vulnerable area is

$$A_{tx_{max}} = N_t \pi (D_x/2)^2 .$$

For horizontal storage, the front maximum vulnerable area is

$$A_{fx_{\max}} = N_f D_x L_x ,$$

and the top maximum vulnerable area is

$$A_{tx_{\max}} = N_t D_x L_x .$$

Here, the subscript, x, represents either the warhead explosive or rocket motor propellant.

With respect to a particular fragment, the vulnerable areas for detonation and burning may be further limited depending on the velocity and diameter of that fragment. Below some critical threshold, none of the presented area of an energetic component is vulnerable. Above the threshold, part or all of the areas is vulnerable. Partial vulnerability is generally a function of the maximum obliquity, $\theta_{\max}$, from the normal to the surface of the weapon component at which an impacting fragment can produce reaction. Expressions for $\theta_{\max}$ for detonation and burning are developed in the following sections.

6.3 Detonation Vulnerability. The vulnerability of a weapon component to initiation of detonation by fragment impact is described by the Jacobs-Roslund formula (Liddiard and Roslund 1993) for critical impact velocity. This formula applies to cylindrical projectiles having diameters greater than the failure diameter of the energetic material.

$$v_{jr} = \frac{a_{jr} (1 + b_{jr}) \left(1 + c_{jr} \frac{h_c}{d_f} \right)}{\cos \theta \sqrt{d_f}} ,$$

where a_{jr} , b_{jr} , and c_{jr} are characteristic constants which have been determined for a number of explosives. Values of the Jacobs-Roslund constants used in conjunction with our analysis are given in Appendix C. It is assumed that the fragment always strikes the vulnerable component with an average value of presented area. Thus, the fragment diameter, d_f , may be determined from the average fragment presented areas assuming a circular cross-section,

$$d_f = 2 \sqrt{\frac{a_{f_{avg}}}{\pi}} .$$

The angle, θ , is the obliquity of the fragment with respect to the normal to the surface of the casing. The obliquity dependence of the critical velocity may be expressed as

$$v_{jr}(\theta) = \frac{v_{jr}(0)}{\cos \theta} .$$

We define the maximum obliquity that can produce detonation for a fragment with velocity, v_f , such that

$$v_{jr}(\theta_{max}) = v_f .$$

Thus,

$$\cos \theta_{max} = \frac{v_{jr}(0)}{v_f} ,$$

and θ_{max} is defined for $v_f \geq v_{jr}(0)$. This represents the critical condition. This model has been calibrated for steel fragments. However, we also used this calibration for the aluminum fragments produced by the TOW-2A warhead casing. As a worst case, we assumed that the fragments were flat-ended cylinders ($b_{jr} = 0$ with an appropriate value for c_{jr}) and that the critical diameter of the energetic material was always smaller than the diameter of the fragment. Since actual fragments rarely exhibit this morphology, a more realistic simulation might result from a random selection of the critical velocity between limits for flat- and round-ended cylindrical fragments.

6.4 Burning Vulnerability. The vulnerability of a weapon component to initiation of burning may be associated with perforation of the casing by the fragment. This is particularly true in the case of thin-walled munitions (Gilman). Some critical residual velocity, v_{rc} , may be required to produce reaction. The THOR velocity equation relates the residual velocity, v_r , to the initial velocity, v_f , for a penetrating fragment.

$$v_r = v_f - 10^{a_v} (h_c a_{f_{avg}})^{b_v} m_f^{c_v} v_f^{d_v} / (\cos \theta)^{e_v} .$$

Again, it is assumed that the fragment strikes the vulnerable component with an average value of presented area. With $v_r = v_{rc}$ at $\theta = \theta_{max}$,

$$\cos \theta_{max} = \left[\frac{10^{a_v} (h_c a_{f_{avg}})^{b_v} m_f^{c_v} v_f^{d_v}}{v_f - v_{rc}} \right]^{1/e_v} ,$$

and θ_{max} is defined for

$$v_f - v_{rc} \geq 10^{a_v} (h_c a_{f_{avg}})^{b_v} m_f^{c_v} v_f^{d_v} .$$

This expression represents the critical condition. Values of the THOR constants used in conjunction with our analysis are given in Appendix C. Since appropriate values of the critical residual velocity are not known, we used $v_{rc} = 0$ as a worst case. This model has also been calibrated for steel fragments but, again, we used the same calibration for the aluminum fragments produced by the TOW-2A warhead casing.

6.5 Comparison of Detonation and Burning Thresholds. It is tempting to presume that the less violent burning response is easier to produce than detonation. However, comparison of the critical velocities predicted by these models shows that this is not always the case. Critical velocities for detonation and burning of a typical fragment are plotted as functions of casing thickness in Figure 4. For thin, easily perforated casings (e.g., ~1 mm for the TOW-2A), burning is produced by fragments having much lower velocities than those required to produce detonation. (This result would still hold if a nonzero critical residual velocity of expected magnitude were used.) For thicker casings (e.g., ~15 mm for the M107), casing perforation is extremely difficult to achieve but detonations may be produced at lower velocities due to the transmitted shock.

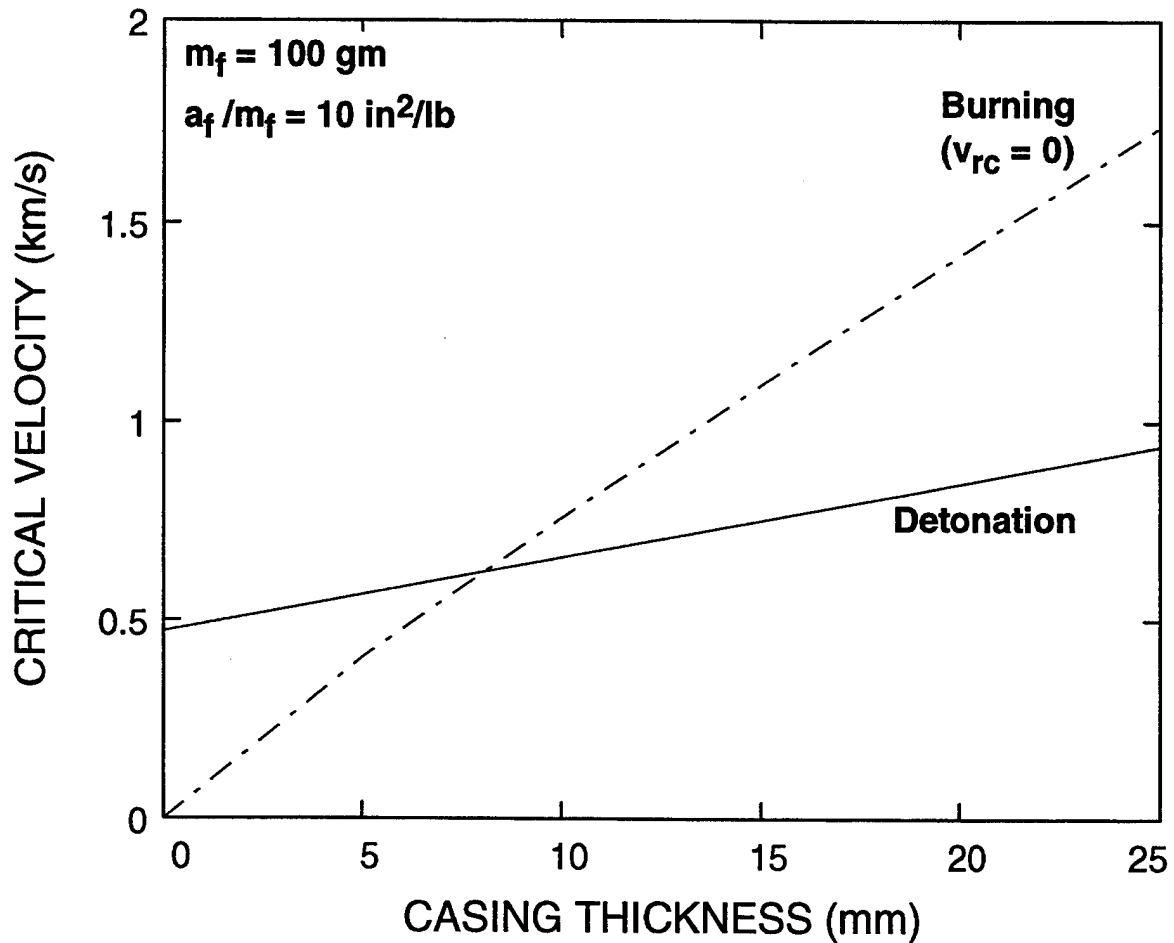


Figure 4. Comparison of critical velocities for detonation and burning.

The point of intersection of the two threshold curves varies with the shape of the fragment (represented by the area to mass ratio). For ratios lower than the $10 \text{ in}^2/\text{lb}$ value used in Figure 4 the point of intersection is shifted toward high values of casing thickness. Thus, results are sensitive to assumptions regarding fragment orientation (presented area) at impact.

6.6 Vulnerable Area Reduction. If the angle subtended at the center of the cylinder representing a vulnerable component by the region on the surface over which an impacting fragment is lethal with respect to detonation or burning ($\theta \leq \theta_{\max}$) is $2\beta_{\max}$ as shown in Figure 5, then the reduction in the maximum vulnerable area of that component is given by

$$A = \sin \beta_{\max} A_{\max} .$$

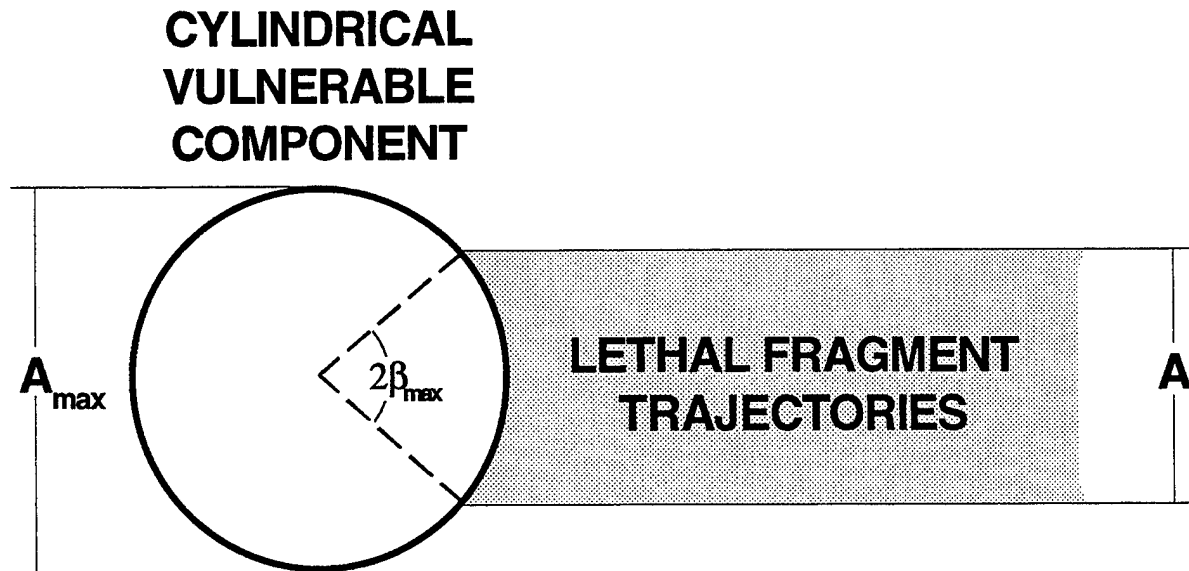


Figure 5. Angular region of vulnerability on a cylindrical charge.

The relationship between $\theta_{\max}$ and $\beta_{\max}$ depends on the storage orientation. For horizontal storage,

$$\beta_{\max} = \theta_{\max}$$

as shown in Figure 6. For vertical storage, the elevation angle of the fragment trajectory, α_f , must be accounted for, as shown in Figure 7. Thus,

$$\cos \beta_{\max} = \frac{\cos \theta_{\max}}{\cos \alpha_f} .$$

6.7 Mechanical Damage Vulnerability. A single mechanical damage criterion based on kinetic energy applies to all weapon components. The stack is considered vulnerable to damage when the kinetic energy of a fragment exceeds a specified value, regardless of the obliquity. Accurate threshold values are not known, so we arbitrarily chose 400 and 50 ft-lb as the critical kinetic energy for the M107 and TOW-2A, respectively. The smaller value for the TOW-2A reflects its much thinner skin.

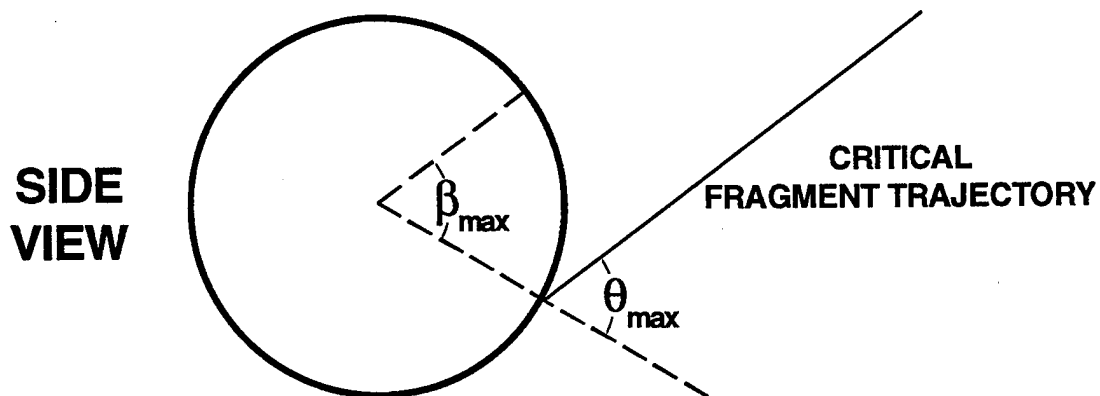


Figure 6. Relationship between the angular region of vulnerability and the maximum obliquity for horizontal storage.

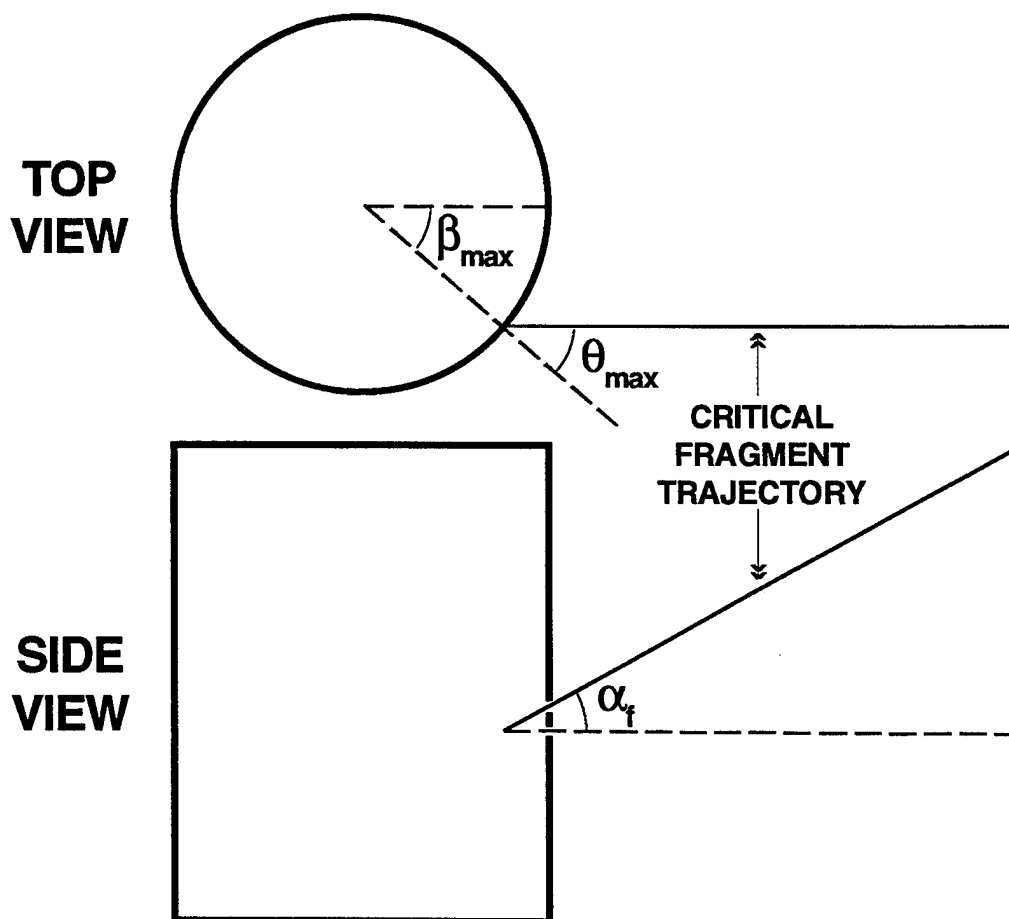


Figure 7. Relationship between the angular region of vulnerability, the maximum obliquity, and the fragment elevation angle for vertical storage.

7. CONTAINER PENETRATION

The TOW-2A is packaged in a steel container, and provisions were made in FRAGPROP for estimating the reduction in fragment mass and velocity associated with penetration of the container. This occurs both when fragments are launched from a donor stack and when they impact an acceptor stack. In addition to the THOR velocity equation, the THOR mass equation is required.

$$m_r = m_f - 10^{a_m} (h_c a_{f_{avg}})^{b_m} m_f^{c_m} v_f^{d_m} / (\cos \theta)^{e_m} .$$

Use of the THOR equations with $\theta = 0$ yields the maximum residual velocity and mass, representing the worst-case scenario. The residual diameter is determined from the residual mass assuming that the area to mass ratio remains unchanged.

8. MINIMUM RANGE

The FRAGHAZ analysis is applicable as long as the angle subtended by the acceptor stack does not exceed the width, ψ , of the azimuthal sector associated with the donor fragmentation data. The minimum range is thus given by

$$r_{min} = \frac{W_a}{2 \tan(\psi/2)} .$$

Note that r_{min} is zero for sectors wider than 180° .

9. COMPUTATIONAL CONFIGURATIONS AND RESULTS

9.1 Theater of Operations Considerations. Storage regulations applicable to basic load ammunition holding areas in theaters of operations limit the explosive quantity in any stack to 4,000 kg (8,818 lb). Such stacks must be separated by at least 77 m (253 ft). (See Army Regulation 385-64.) We considered M107 and TOW-2A stacks containing approximately this maximum weight.

Each M107 projectile contains 15.4 lb of Composition B. Thus, a pallet of eight projectiles contains 123.2 lb and a stack of 72 pallets contains 8,870.4 lb.

A TOW-2A missile contains 6.7 lb of LX-14 plus 0.2 lb of other high explosives in the warhead, 1.2 lb of M7 propellant in the launch motor and 7.0 lb of GCV propellant in the flight motor for a total of 15.1 lb. A pallet of 12 missiles contains 181.2 lb, and a stack of 48 pallets contains 8,697.6 lb.

9.2 Stack Arrangements. The arrangement of pallets affects the lethality and vulnerability of a stack. Consideration of the least and most lethal arrangements as well as the least and most vulnerable arrangements provides scope to a determination of propagation probabilities. Pertinent stack parameters may be estimated from pallet dimensions given in Appendix C.

Each M107 pallet contains eight vertical projectiles in an arrangement that is one projectile high, four projectiles wide, and two projectiles deep ($1 \times 4 \times 2$). The 72 pallets may be arranged in any permutation of $3 \times 4 \times 6$, where the first dimension is the height of the stack in pallets, the second dimension is the width of the stack in pallets, and the third dimension is the depth of the stack in pallets. Individual pallets retain their $1 \times 4 \times 2$ orientations for all permutations. Table 1 gives the number of interaction areas, stack dimensions, and total front and top vulnerable area associated with each permutation.

The $4 \times 3 \times 6$ pallet arrangement is both least lethal (having the fewest interaction areas) and least vulnerable (having the smallest vulnerable area, while the $4 \times 6 \times 3$ arrangement is both most lethal and most vulnerable.

Each TOW-2A pallet contains 12 horizontal missiles in an arrangement that is 3 missiles high, 1 missile wide, and 4 missiles deep ($3 \times 1 \times 4$). The 48 pallets may be arranged in any permutation of $3 \times 4 \times 4$. Table 2 gives the number of missiles on the front face, stack dimensions, and total front and top vulnerable areas associated with each permutation.

Based on the number of missiles on the front face, the $4 \times 4 \times 3$ pallet arrangement is most lethal. The other two arrangements are nominally equal in lethality. The $3 \times 4 \times 4$ arrangement is most vulnerable, while the $4 \times 3 \times 4$ arrangement is least vulnerable. The missile stacks are considerably larger and more vulnerable to being hit than the artillery projectile stacks.

Table 1. Lethality and Vulnerability Parameters for M107 Stacks

Pallet Arrangement ($n_h \times n_w \times n_d$)	Number of Interaction Areas on Face of Stack	Stack Dimensions			Vulnerable Areas (front + top) (ft ²)
		height (ft)	width (ft)	depth (ft)	
3 × 4 × 6	45	8.00	9.63	7.20	146.3
3 × 6 × 4	69	8.00	14.50	4.77	185.2
4 × 3 × 6	44	10.67	7.20	7.20	128.6
4 × 6 × 3	92	10.67	14.50	3.55	206.2
6 × 3 × 4	66	16.00	7.20	4.77	149.5
6 × 4 × 3	90	16.00	9.63	3.55	188.3

Table 2. Lethality and Vulnerability Parameters for TOW-2A Stacks

Pallet Arrangement ($n_h \times n_w \times n_d$)	Number of Missiles on Face of Stack	Stack Dimensions			Vulnerable Areas (front + top) (ft ²)
		height (ft)	width (ft)	depth (ft)	
3 × 4 × 4	36	10.06	20.25	14.64	500.0
4 × 3 × 4	36	13.41	15.17	14.64	425.3
4 × 4 × 3	48	13.41	20.25	10.96	493.4

The single M107 projectile donor may be erect or inverted and may detonate at any specified height above the ground. The arrangements producing the least and greatest lethality are not apparent. However, preliminary computations indicated that the inverted round at the surface may be considered least lethal and the erect round at a 50-ft elevation most lethal.

9.3 Propagation Probabilities and Distances. The graphics program developed in conjunction with FRAGPROP plots the probabilities of exceeding the criteria for propagation of detonation (labeled D), burning (B), and mechanical damage (M), as well as the overall (lethal and nonlethal) hit probability (H) on a logarithmic scale as a function of range. A typical example is shown in Figure 8. The dashed vertical line near the probability axis indicates the minimum range for which the analysis is accurate.

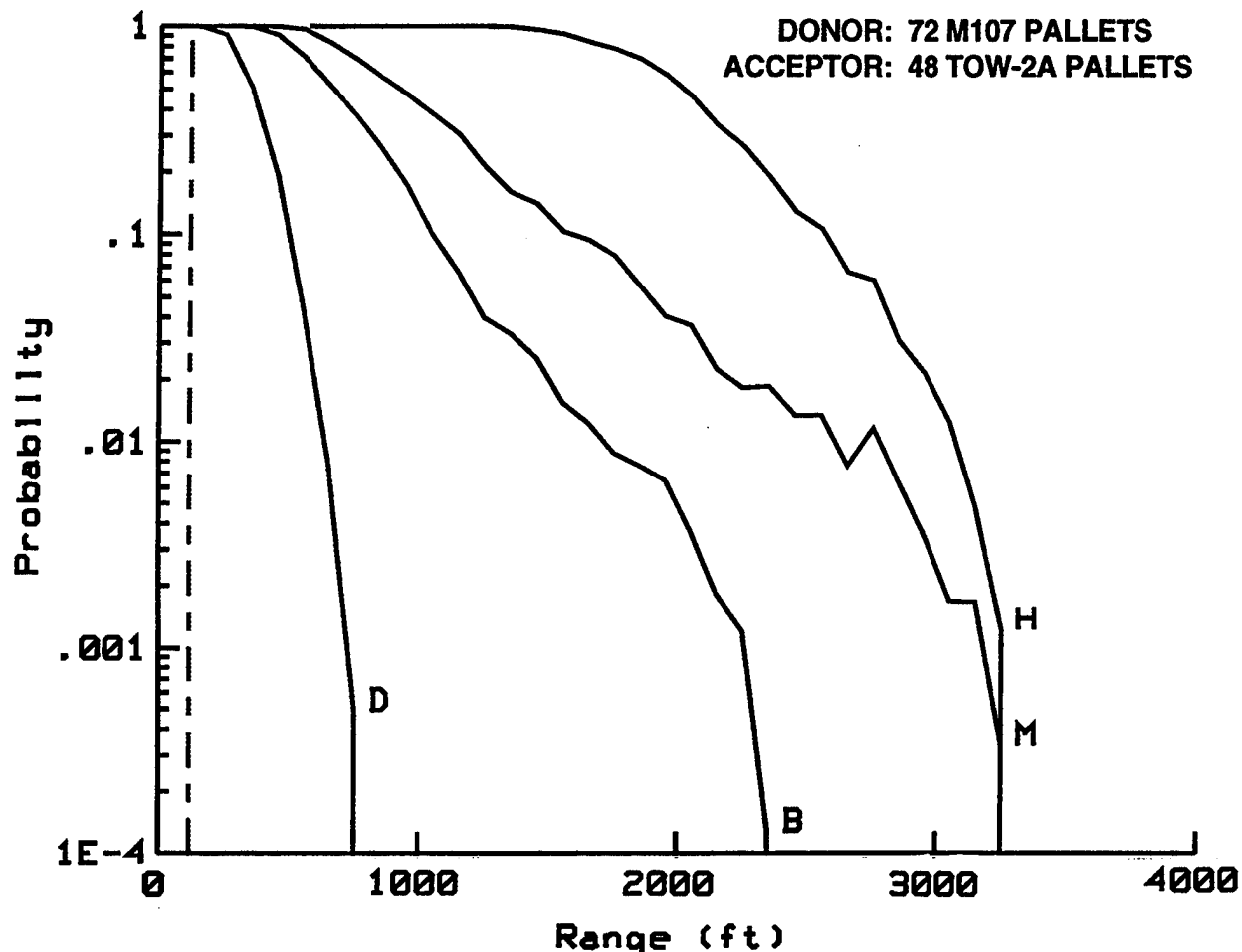


Figure 8. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for an M107 donor stack against a TOW-2A acceptor stack.

Generally, the probabilities decrease with increasing range and, theoretically, they can be expected to vanish entirely beyond some maximum range. For comparison purposes, it is often useful to consider the range at which a probability drops to some small value (e.g., 1%). The small probabilities predicted at long range may become nonmonotonic prior to vanishing. The ability to predict nonzero probabilities at long range is limited by the number of fragments in the donor data and the number of replications. Generally, the results are not accurate near vanishing probability. The number of replications required to produce monotonically decreasing probabilities varies inversely with the number of fragments describing the donor.

We made FRAGPROP computations using zero-wind conditions and soil constants varying between 0.5 and 4.0 for each of the three donors (single M107 projectiles, stacked M107 pallets, and stacked TOW-2A pallets) vs. both of the acceptors (stacked M107 and TOW-2A pallets). Preliminary computations were made to determine the maximum range required in each problem. Input conditions for the final computations are summarized in Table 3. These values depend on the donor and apply to both acceptors.

Table 3. FRAGPROP Computation Input Conditions

Donor	Number of Fragments	Number of Replications	Segment Size (ft)	Maximum Range (ft)
Single M107	870	50	50	2,400
M107 Stack	215	200	100	3,600
TOW-2A Stack	365	125	20	720

In order to determine the scope of probability values, two computations were made for each donor-acceptor pair: one for the least lethal donor stack arrangement against the least vulnerable acceptor stack arrangement and one for the most lethal donor stack arrangement against the most vulnerable acceptor stack arrangement. The graphics capability allows the four probabilities to be plotted as functions of range for two problems at a time as shown in Figures 9–14. Associated pairs of probability curves are joined by shading lines.

Results for single M107 donors against palletized M107 acceptors are shown in Figure 9. The probabilities decrease gradually with range. The distance at which the detonation propagation probability drops to 1% (on the upper curve of the pair) is 125 ft, while the 1% distance for burning is only 113 ft. Thus, burning propagation appears unlikely in this configuration. The probability of mechanical damage remains above 1% to a range of 687 ft, while the hit probability drops to 1% at 1,369 ft.

Donor: INVERTED M107 @ 0 ft
 Acceptor: 4x3x6 M107 PALLETS

Donor: ERECT M107 @ 50 ft
 Acceptor: 4x6x3 M107 PALLETS

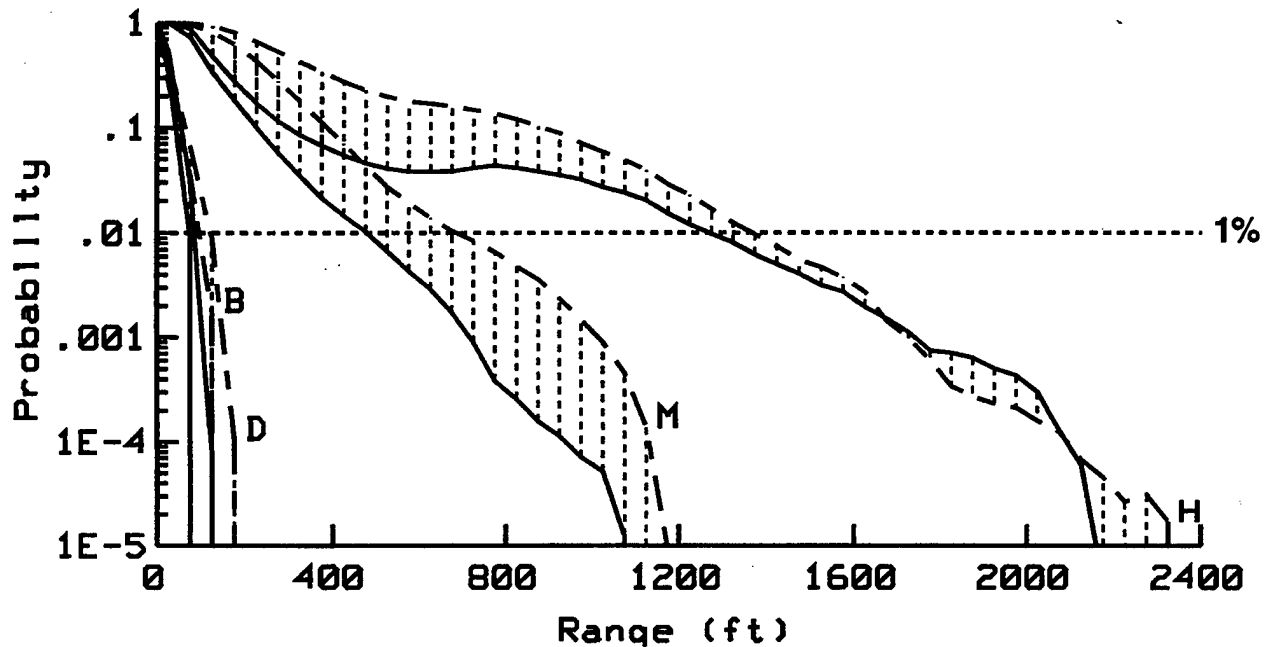


Figure 9. Probabilities of burning, mechanical damage, and hit as functions of range for single M107 donor projectiles against M107 acceptor stacks.

Donor: INVERTED M107 @ 0 ft
 Acceptor: 4x3x4 TOW-2A PALLETS

Donor: ERECT M107 @ 50 ft
 Acceptor: 3x4x4 TOW-2A PALLETS

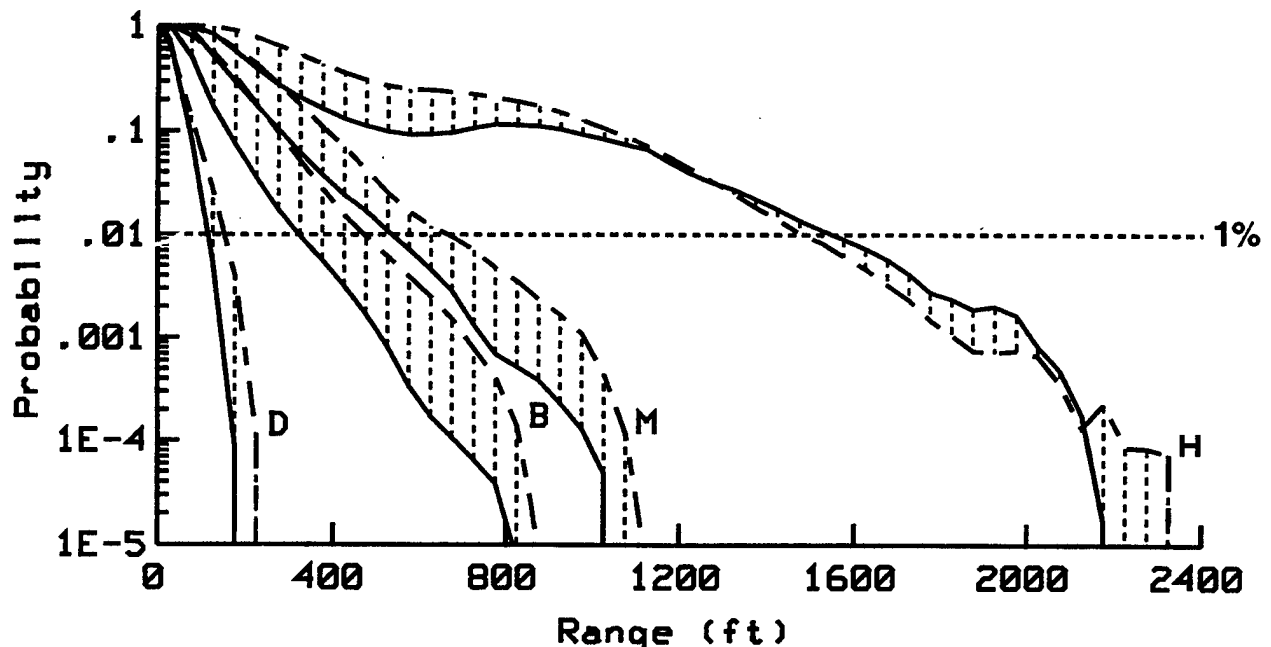


Figure 10. Probabilities of burning, mechanical damage, and hit as functions of range for single M107 donor projectiles against TOW-2A acceptor stacks.

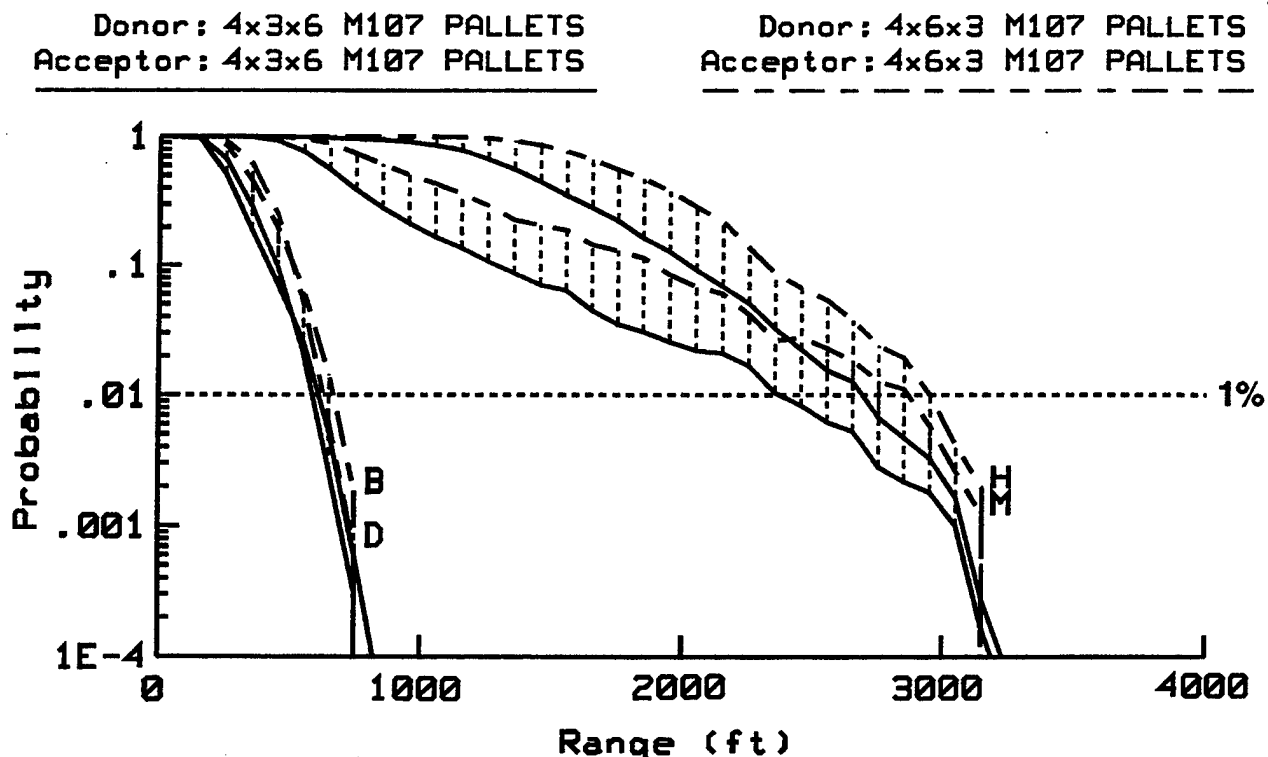


Figure 11. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for M107 donor stacks against M107 acceptor stacks.

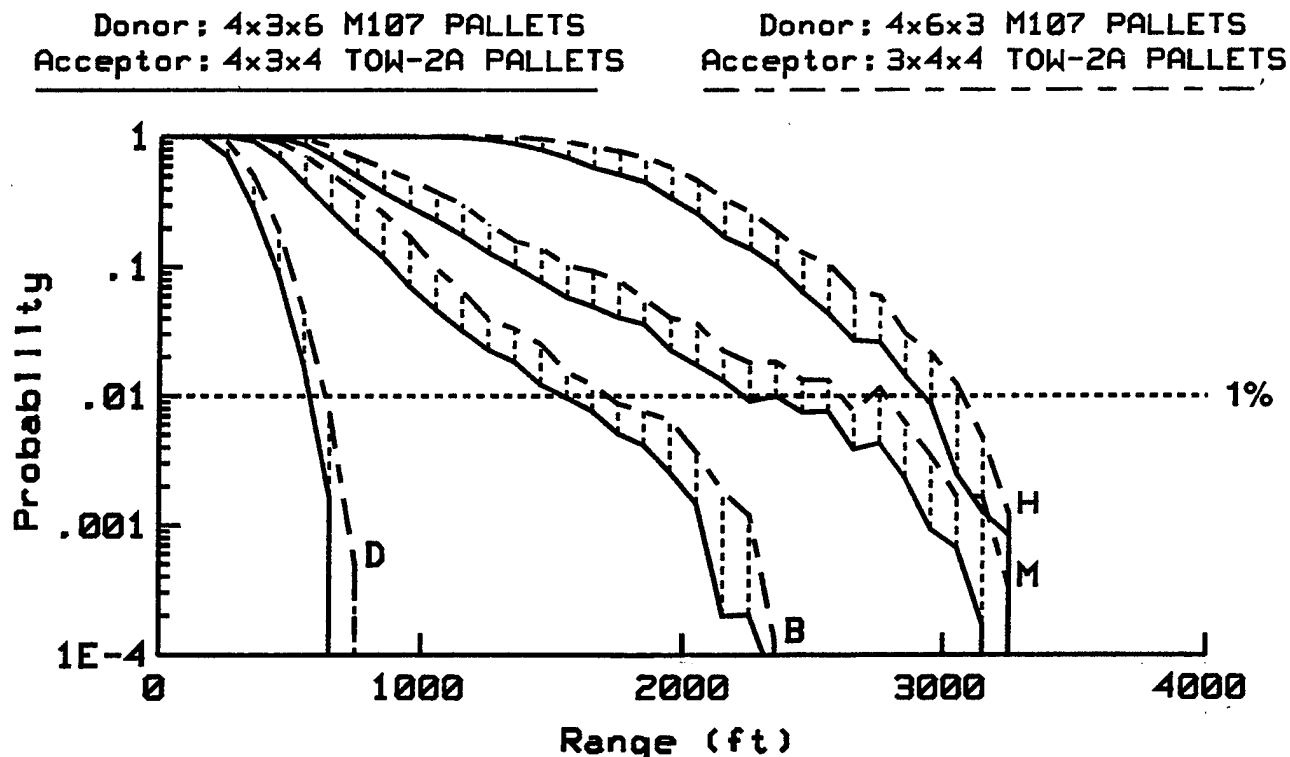


Figure 12. Probabilities of detonation, burning, mechanical damage, and hit as functions of range for M107 donor stacks against TOW-2A acceptor stacks.

Donor: 3x4x4 TOW-2A PALLETS
 Acceptor: 4x3x6 M107 PALLETS

Donor: 4x4x3 TOW-2A PALLETS
 Acceptor: 4x6x3 M107 PALLETS

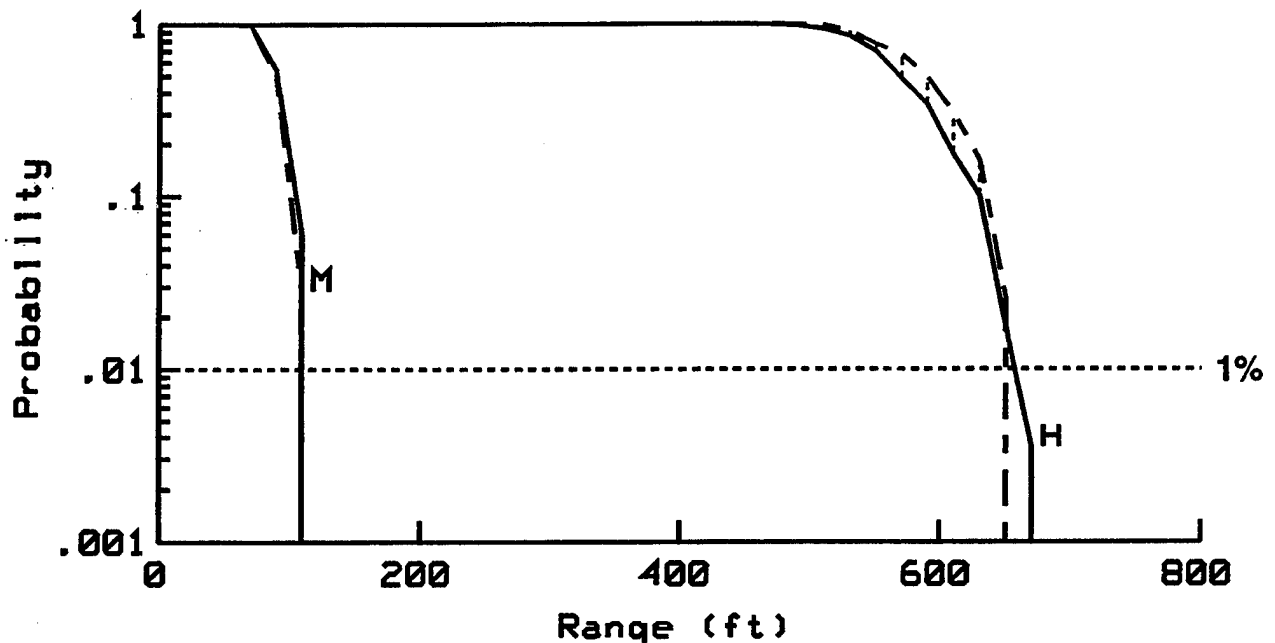


Figure 13. Probabilities of mechanical damage and hit as functions of range for TOW-2A donor stacks against M107 acceptor stacks.

Donor: 3x4x4 TOW-2A PALLETS
 Acceptor: 4x3x4 TOW-2A PALLETS

Donor: 4x4x3 TOW-2A PALLETS
 Acceptor: 3x4x4 TOW-2A PALLETS

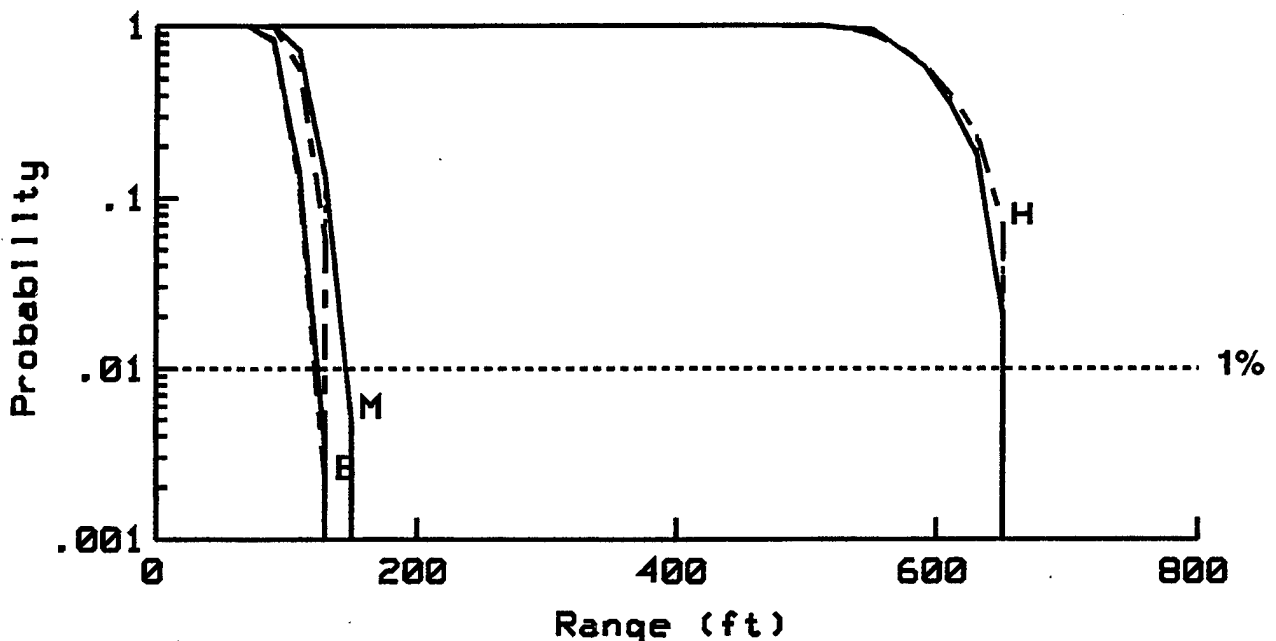


Figure 14. Probabilities of burning, mechanical damage, and hit as functions of range for TOW-2A donor stacks against TOW-2A acceptor stacks.

Similar results for single M107 donors against palletized TOW-2A acceptors are shown in Figure 10. The 1% distance for detonation is 162 ft, while a burn probability greater than 1% persists to a distance of 471 ft. The probability of mechanical damage drops to 1% at 666 ft. The range over which the overall hit probability remains above 1% is 1,551 ft.

Results for palletized M107 donors against palletized M107 acceptors are shown in Figure 11. Probabilities near unity persist to a greater range. This arrangement produces a 1% probability of detonation at 642 ft, while burning reactions propagate with a probability of 1% or greater over distances to 685 ft. The probability of mechanical damage remains above 1% to 2,871 ft. The 1% distance for the overall hit probability is 2,950 ft.

Similar results for palletized M107 donors against palletized TOW-2A acceptors are shown in Figure 12. This arrangement produces a 1% probability of detonation at 643 ft, while burning reactions propagate with a probability of 1% or greater over distances to 1,715 ft. A probability of mechanical damage greater than 1% persists to 2,779 ft. The 1% distance for the overall hit probability is 3,080 ft.

Results for palletized TOW-2A donors against palletized M107 acceptors are shown in Figure 13. Neither detonation nor burning propagation is predicted. The other probabilities remain high and then drop rapidly to zero with increasing range. Probabilities of mechanical damage exceeding 1% persist to 127 ft. The 1% distance for hit probability is 662 ft.

Similar results for palletized TOW-2A donors against palletized TOW-2A acceptors are shown in Figure 14. No detonation propagation is predicted, but burning propagation occurs at the 1% probability level out to a range of 129 ft. Probabilities of mechanical damage of 1% or greater persist to 149 ft. The 1% distance for hit probability is 667 ft. The fact that the burn and mechanical damage curves nearly coincide indicates that the arbitrary kinetic energy threshold for mechanical damage is too high.

The 1% propagation probability ranges discussed in the foregoing paragraphs are summarized in Table 4.

The lethality of the three donors varies substantially. Significant differences between the single and palletized M107 donors occur simply because of the number of munitions involved. The probabilities for

Table 4. 1% Propagation Probability Distances (ft)

Donor	Acceptor	Detonation	Burning	Mechanical Damage	Hit
Single M107	M107 Stack TOW-2A Stack	124	113	687	1,369
		162	471	666	1,551
M107 Stack	M107 Stack TOW-2A Stack	642	685	2,871	2,950
		643	1,715	2,779	3,080
TOW-2A Stack	M107 Stack TOW-2A Stack	NP	NP	127	662
		NP	129	149	667

NP = no propagation

these donors drop slowly and the maximum ranges are considerably greater than the 1% ranges. TOW-2A donors are much less lethal due to the small fragments they produce. The character of the probability curves for these donors is different. Probabilities remain high until approaching the maximum range where they drop sharply.

The relative vulnerability of the two acceptors depends on the type of response. The TOW-2A is considerably more susceptible to burning than the M107. For the other responses, the 1% propagation distances are generally very similar. Although the (rather arbitrary) kinetic energy threshold is lower for TOW-2A acceptors, their probability of mechanical damage is reduced due to the presence of the container.

It is notable that this analysis (which includes a number of worst-case assumptions) does not preclude propagation of detonation from M107 stacks stored at the required 253-ft distance from either acceptor. At 250 ft, the probability of detonation propagation to an M107 stack is between 68% and 94% (depending on the stack configurations) and the probability of detonation propagation to a TOW-2A stack is between 71% and 91%.

10. SUMMARY AND CONCLUSIONS

By combining several existing models, we have developed a tool (FRAGPROP) for estimating the probabilities associated with the propagation of reaction between user-described ammunition stacks. The

acceptor responses considered include detonation of energetic materials, burning of energetic materials (but not of combustible packaging), and mechanical damage. The models include the FRAGHAZ program for the Monte Carlo treatment of fragment trajectories and the accumulation of hit probabilities, the Jacobs-Roslund criterion for initiation of detonation, and the ballistic limit condition for initiation of burning.

A number of difficulties arose in applying this tool to munitions of interest. Since the appropriate fragmentation input data was not always available, notably in the case of missiles, we developed methods of estimating this data. It was necessary to represent the vulnerability of explosives and propellants for which Jacobs-Roslund parameters are not available using estimates based on values for similar compositions. It was also necessary to assess the lethality of aluminum fragments with models intended for use with steel fragments.

Although only two weapons have been specifically analyzed, the responses of the thick-walled M107 and the thin-walled TOW-2A are expected to encompass the range of responses of a wide variety of munitions. The analysis was used to predict distances below which the propagation probabilities for each response exceed 1% (see Table 4). Based on these predictions, we conclude that the artillery ammunition donor stacks are much more lethal than the missile donor stacks (which cannot produce detonation) and the missile acceptor stacks are more vulnerable to the propagation of burning (but not of detonation) than the artillery ammunition acceptor stacks. However, different assumptions regarding the orientation of fragments at impact time might enhance sensitivity to burning.

11. REFERENCES

- Dehn, J. T. "Models of Explosively Driven Metal." BRL-TR-2626, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, December 1984.
- Department of the Army. "Ammunition and Explosives Safety Standards." AR 385-64, 22 May 1987.
- Department of the Army. "Joint Munitions Effectiveness Manual of Fragmentation Data." FM 101-62-3.
- Gilman, R. D. "Cased Granular Propellant Ignition." Date and source unknown.
- Liddiard, T. P., and L. A. Roslund. "Projectile/Fragment Impact Sensitivity of Explosives." NSWC TR 89-184, Naval Surface Warfare Center, Silver Spring, MD, June 1993.
- McClesky, F. "Quantity-Distance Fragment Hazard Computer Program (FRAGHAZ)." NSWC TR 87-59, Naval Surface Warfare Center, Silver Spring, MD, February 1988.
- Victor, A. C. "Prediction/Analysis of Munition Reactions for Insensitive Munitions Threat Hazard Assessment." Proceedings of the Insensitive Munition Technology Symposium, Williamsburg, VA, June 1994.

INTENTIONALLY LEFT BLANK.

APPENDIX A:
PROPAGATION ACCIDENTS IN AMMUNITION HOLDING AREAS

INTENTIONALLY LEFT BLANK.

We considered several sources of information which are pertinent to propagation accidents in ammunition holding areas. These include reports of actual events found in the files of the Department of Defense Explosive Safety Board as well as results of hazard classification, packaging and MILVAN tests.

At least 57 accidents involving open storage of ammunition have been identified. We have reviewed reports of many of these. There are a number of difficulties associated with the data given in these reports. Much of it is old, dating to the World War II era. The observations were made at different times by different people with different ideas about what is important. The amount of information reported varies widely, and details of storage configurations were not usually included.

We drew a number of conclusions from our review. En masse detonation of large quantities of ammunition is rare, usually involving bombs or large projectiles. More commonly, propagation accidents follow a sequence of events involving fire in an ammunition stack and cookoff of munitions in that stack. This leads to propagation of fire to neighboring stacks via hot fragments and burning debris. The process is then repeated in events which may take days to unfold. Propulsive reactions in which items such as rockets and mortar rounds are launched toward neighboring or distant stacks often contribute to propagation. White phosphorus rounds are also major contributors to this type of propagation. Three illustrative accident reports are summarized in Table A-1.

The most relevant hazard classification test is the bonfire test. While this provides the maximum fragment radius, it doesn't always identify the number of fragments which travel shorter distances. Distances over which firebrands are spread are usually not reported. Another relevant series of tests were conducted in MILVAN containers.¹ Although the quantities of explosive in each MILVAN were modest, items were thrown large distances.

The importance of packaging was demonstrated by Teitell and Reeves² who conducted tests to determine the vulnerability of several munitions in wood packaging to several threats. The general results

¹ Lawrence, W. "Fragment Hazards From Munitions in Containers." BRL-TR-3203, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1991.

² Teitell, L., and H. J. Reeves. "Fire Retardant Packaging for Artillery Ammunition." BRL-MR-2490, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1975.

Table A-1. Summary of Representative Propagation Accident Reports

DDESB FILE	466	480	1271
LOCATION	PUEBLO, CO	IE SHIMA, JAPAN	DA NANG, VIETNAM
AMMUNITION	3 "PILES" OF TNT-LOADED 3" M42 AMMUNITION SEPARATED BY 120' AND 285'	2162 TONS OF BOMBS, ROCKETS, SMALL ARMS, 20mm IN OPEN STORAGE WITH ABOUT 200' BETWEEN STACKS	LARGE MARINE CORPS AMMUNITION SUPPLY POINT
CAUSE	LIGHTNING		FIRE, POSSIBLY CAUSED BY ENEMY ACTION
DURATION	HOURS	2 DAYS	> 24 HOURS
RESULT	• MULTIPLE EVENTS • FIRE SPREAD BY HOT FRAGMENTS AND BURNING DEBRIS • VIRTUALLY ALL AMMO LOST • FRAGS TO 3700' • THROWN ROUND DETONATED ON IMPACT AT 200' • BURNING WOOD AND FIBERBOARD TRAVELED 400'.	ESSENTIALLY EVERYTHING LOST	• MULTIPLE EXPLOSIONS • ROCKETS COMMUNICATED REACTION TO AN AIR FORCE ASP • LARGE LOSS OF AMMO • "NO CELL APPEARED TO COMMUNICATE DIRECTLY TO ITS NEIGHBOR."

of these tests were that the threats did not cause detonations but started fires in propellants and wood packaging. Propellant fires rapidly cooked off warheads and wood fires cooked off warheads causing both detonations and less violent explosions. Detonations sometimes scattered stacks and stopped reaction. Fire-resistant packaging prevented the spread of reaction in some cases.

We conclude that the dominant mode of propagation in ammunition accidents is fire leading to cookoff leading to more fire and more cookoff. Combustible packaging and propulsive reactions are major contributors to propagation. Insensitive munition technology will reduce the probability of propagation, especially where stacks can be separated by more than 50 ft. However, quantitative analysis of propagation probability for this mechanism is probably not possible at this time.

APPENDIX B:
FRAGPROP LISTING

INTENTIONALLY LEFT BLANK.

```

C***** F000010
C
C      program frgpr F000020
C
C      F R A G P R O P F000030
C      F000040
C      F000050
C      common blocks F000060
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum F000070
C      common/repls/ nreps,fnreps F000080
C      common/frags/ nfrags,tfrags F000090
C      common/range/ rngmax, rngseg, rngsgh, rngscl, rngsix, rngmin F000100
C      common/rkvar/ rk(7) F000110
C      F000120
C      F000130
C      type declarations F000140
C      character*8 uniela,unirem F000150
C      double precision rndinp,rndnxt,rndsav F000160
C      F000170
C      report the program title F000180
C      write(*,'(//,a)') ' FRAGPROP' F000190
C      F000200
C      set up the run F000210
C      call setup(rndinp,rndsav,rndnxt, ihbgn,tsbgn) F000220
C      F000230
C      begin the replication loop F000240
C      do 300 irep=1,nreps F000250
C      F000260
C      ifrcm=(irep-1)*nfrags F000270
C      F000280
C      initialize the replication parameters F000290
C      call rpini(rndnxt, slcnst,asldeg,alttud,wspeed,wdrdeg,wdrad, F000300
+      jxrng,jrkmax) F000310
C      F000320
C      begin the fragment loop F000330
C      do 200 ifrag=1,nfrags F000340
C      F000350
C      estimate the remaining time F000360
C      frcmp=max(float(ifrcm+ifrag-1)/tfrags,smlnum) F000370
C      call ertim(ihbgn,tsbgn,frcmp,timela,uniela,timrem,unirem) F000380
C      F000390
C      report the current replication and F000400
C      fragment and the remaining time F000410
C      write(*,'(a,i3,a,i3,a,i3,a,i3,a,f5.2,a)') F000420
+      '+Replication: ',irep,'/',nreps,' Fragment: ',ifrag,'/', F000430
+      nfrags,' Remaining Time: ',timrem,unirem F000440
C      F000450
C      initialize the fragment parameters F000460
C      call frini(irep,ifrag,rndnxt, irico,lrange,htfrft,mfrht1, F000470
+      vfrag,seldeg,selrad,aeldeg,aelrad,cd1,cd2,cd3, F000480
+      deldst) F000490
C      F000500
C      begin the trajectory computation loop F000510
C      itraj=0 F000520
100 itraj=itraj+1 F000530
C      F000540
C      determine the time step F000550
C      call tstep(deldst,irico,vfrag,aelrad, deltim) F000560
C      F000570
C      increment the time F000580
C      rk(1)=rk(1)+deltim F000590
C      F000600
C      perform the Runge-Kutta integration F000610
C      call rkint(wspeed,wdrdeg,wdrad,jxrng,jrkmax,cd1,cd2,cd3, F000620
+      deltim,alttud,ifrag) F000630
C      F000640
C      determine the fragment variable values F000650
C      at the end of the integration step F000660
C      call frvar(vfrag,selrad,seldeg,aelrad,aeldeg,range,irange) F000670
C      F000680

```

c	perform hazard computations as required	F000690
c	call hazch(ifrag,irange,mfrht1,mfrht2,lrange,aelrad,vfrag)	F000700
c		F000710
c	check the range for trajectory termination	F000720
c	if(range.gt.rngmax) go to 200	F000730
c		F000740
c	check the fragment velocity for trajectory termination	F000750
c	if(vfrag.lt.20.0) go to 200	F000760
c		F000770
c	check for ricochet	F000780
c	if(rkv(5).le.0.0) then	F000790
c		F000800
c	check the ricochet count	F000810
c	if(irico.gt.6) go to 200	F000820
c		F000830
c	check the elevation angle for trajectory termination	F000840
c	if(aeldeg.ge.asldeg) go to 200	F000850
c		F000860
c	determine the ricochet velocity and angle	F000870
c	call ricoc(slcnst,vfrag,aeldeg,selrad,vrico,ricrad)	F000880
c		F000890
c	check the ricochet velocity for trajectory termination	F000900
c	if(vrico.lt.20.0) go to 200	F000910
c		F000920
c	reinitialize the velocity integration variables	F000930
c	call reini(irico,vrico,ricrad,wspeed,wdrad, vfrag,	F000940
+	selrad,seldeg,aelrad,aeldeg,deldst,mfrht1)	F000950
c		F000960
c	endif	F000970
c		F000980
c	end of the trajectory computation loop	F000990
c	go to 100	F001000
c		F001010
c	end of the fragment loop	F001020
200	continue	F001030
c		F001040
c	accumulate hit probabilities	F001050
c	call hpacc	F001060
c		F001070
c	end of the replication loop	F001080
300	continue	F001090
c		F001100
c	determine the elapsed time	F001110
c	call ertim(ihbgn,tsbgn,1.0,timela,uniela,timrem,unirem)	F001120
c		F001130
c	write the output files	F001140
c	call outpt	F001150
c		F001160
c	report completion	F001170
c	write(*,'(a,f5.2,a/)')	F001180
+	' +Computation Complete	F001190
+	+ timela,uniela	F001200
c		F001210
c	stop ' '	F001220
c		F001230
c	end	F001240
c		F001250
c	*****	F001260

```

C***** S000010
C S000020
C      subroutine setup(rndinp,rndsav,rndnxt, ihbgn,tsbgn) S000030
C S000040
C      sets up the run S000050
C S000060
C      common blocks S000070
C      common/error/ meror,irerr,iferr,merout S000080
C      common/rangs/ nrsegs,nrsegs S000090
C      common/denst/ fd(97),fdd(97),fdb(97),fdm(97) S000100
C      common/phitt/ ph(97),phd(97),phb(97),phm(97) S000110
C S000120
C      type declarations S000130
C      character dfile*25,afile*25 S000140
C      double precision rndinp,rndnxt,rndsav S000150
C S000160
C      assign constants S000170
C      call const S000180
C S000190
C      obtain input S000200
C      call input(rndinp,rndsav,rndnxt,dfile,minvrt,afile) S000210
C S000220
C      compute parameters requiring both donor and acceptor input S000230
C      call doacc S000240
C S000250
C      output the run conditions S000260
C      call outin(rndsav,dfile,minvrt,afile) S000270
C S000280
C      compute the maximum vulnerable areas of the acceptor S000290
C      call vulna S000300
C S000310
C      initialize the hit probabilities S000320
C      call arini(nrsegs,ph,phd,phb,phm,0.0) S000330
C S000340
C      initialize the fragment densities S000350
C      call arini(nrsegs,fd,fdd,fdb,fdm,0.0) S000360
C S000370
C      obtain the starting time S000380
C      call gettim(ihbgn,imbgn,isbgn,ilbgn) S000390
C      tsbgn=3600.0*float(ihbgn)+60.0*float(imbgn) S000400
C      +float(isbgn)+float(ilbgn)/100.0 S000410
C S000420
C      report return S000430
C      if(meror.gt.0) write(2,'(1x,a)') 'returning from setup' S000440
C S000450
C      return S000460
C S000470
C      end S000480
C S000490
C***** S000500

```

```

C***** C000010
C      subroutine const C000020
C      assigns constants C000030
C      C000040
C      C000050
C      common blocks C000060
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum C000070
C      common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2, C000080
C      + hcam3 C000090
C      common/cnstu/ fgrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmftt,fi2tc2 C000100
C      C000110
C      mathematical constants C000120
C      qpi=atan(1.0) C000130
C      hpi=2.0*qpi C000140
C      pi=4.0*qpi C000150
C      tpi=2.0*pi C000160
C      raddeg=360.0/tpi C000170
C      degrad=tpi/360.0 C000180
C      smlrad=1.0e-3*degrad C000190
C      bignum=1.0e30 C000200
C      smlnum=1.0e-30 C000210
C      C000220
C      physical constants C000230
C      gg=32.174 C000240
C      tglbgr=1.4e4*gg C000250
C      hrair0=3.8239e-2 C000260
C      racnst=3.39e4 C000270
C      vsound=1.1164e3 C000280
C      hccnst=2.86e5 C000290
C      hcam1=0.75 C000300
C      hcam2=1.5 C000310
C      hcam3=2.5 C000320
C      C000330
C      unit conversions C000340
C      fgrtlb=1./7.0e3 C000350
C      fgmtlb=2.2046226e-3 C000360
C      fgrtgm=fgrtlb/fgmtlb C000370
C      ffttm=0.3048 C000380
C      fftmm=1.e-3*ffttm C000390
C      fmftt=1.0/ffttm C000400
C      fi2tc2=6.4516 C000410
C      C000420
C      return C000430
C      C000440
C      end C000450
C      C000460
C      C000470
C***** C000480

C***** I000010
C      subroutine input(rndinp,rndsav,rndnxt,dfile,minvrt,afile) I000020
C      obtains input I000030
C      I000040
C      I000050
C      type declarations I000060
C      character dfile*25,afile*25 I000070
C      double precision rndinp,rndnxt,rndsav I000080
C      I000090
C      input randomization seed I000100
C      call rndin(rndinp,rndsav,rndnxt) I000110
C      I000120
C      input parameters I000130
C      call parin I000140
C      I000150
C      input donor data I000160
C      call donin(dfile,minvrt) I000170
C      I000180
C      input acceptor data I000190
C      call accin(afile) I000200
C      I000210
C      input range data I000220
C      call rngin I000230
C      I000240
C      return I000250
C      I000260
C      end I000270
C      I000280
C      I000290
C***** I000300

```

```

C***** R000010
C R000020
C      subroutine rndin(rndinp,rndsav,rndnxt) R000030
C R000040
C      obtains randomization seed input R000050
C R000060
C      type declarations R000070
C      double precision rndinp,rndsav,rndnxt R000080
C R000090
C 100 write(*,'(a\)' ) ' Enter Monte Carlo Seed (1 to 2147483646): ' R000100
C      read(*,*,err=100) rndinp R000110
C      if(rndinp.lt.1.0.or.rndinp.gt.2147483646.0) go to 100 R000120
C R000130
C      rndsav=rndinp R000140
C      rnddum=rndom(rndinp) R000150
C      rndnxt=0.0d0 R000160
C R000170
C      return R000180
C R000190
C      end R000200
C R000210
C***** R000220

C***** P000010
C P000020
C      subroutine parin P000030
C P000040
C      obtains parameter input P000050
C P000060
C      common blocks P000070
C      common/error/ meror,irerr,iferr,merout P000080
C      common/repls/ nreps,fnreps P000090
C      common/prmtr/ slcmin,slcmax,altmin,altmax, P000100
C      +      wspmin,wspmax,wdrmin,wdrmax P000110
C P000120
C      open(4,file='fp.prm',status='old') P000130
C      read(4,*) meror,irerr,iferr P000140
C      if(meror.gt.0) open(2,file='fp.err',status='unknown') P000150
C      read(4,*) nreps P000160
C      nreps=min(nreps,999) P000170
C      fnreps=float(nreps) P000180
C      read(4,*) slcmin,slcmax P000190
C      read(4,*) altmin,altmax P000200
C      read(4,*) wspmin,wspmax P000210
C      read(4,*) wdrmin,wdrmax P000220
C      close(4) P000230
C P000240
C      return P000250
C P000260
C      end P000270
C P000280
C***** P000290

```

```

C***** D000010
C D000020
      subroutine donin(dfile,minvrt) D000030
C D000040
      obtains donor input D000050
C D000060
      common blocks D000070
      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum D000080
      common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2, D000090
+      hcam3 D000100
      common/cnstu/ fgtrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmftf,fi2tc2 D000110
      common/error/ meror,irerr,iferr,merout D000120
      common/frmin/ fmgrmn,fmgmmn,ezndeg D000130
      common/zones/ elzdeg,azsdeg,azsrad D000140
      common/donor/ nunits,htstft,htbaft D000150
      common/frags/ nfrags,tfrags D000160
      common/fragd/ famil(900),vfifs(900),farma(900), D000170
+      fatn(900),ezldeg(900) D000180
      common/fragc/ facm2(900),fdmm(900),fmgm(900) D000190
      common/fragx/ vcrdx(900),vcrbx(900) D000200
      common/fragk/ frkec(900) D000210
      common/names/ dname,aname D000220
C D000230
      local array D000240
      dimension fmult(36) D000250
C D000260
      type declarations " D000270
      character dname*40,aname*40 D000280
      character dfile*25,answr*4 D000290
C D000300
      report call D000310
      if(meror.gt.0) write(2,'(4x,a)') 'donin called' D000320
C D000330
      user specification input D000340
      open(4,file='fp.dnr',status='old') D000350
      read(4,'(40x,a)') dname D000360
      read(4,'(40x,a)') dfile D000370
      read(4,'(40x,a)') answr D000380
      minvrt=0 D000390
      if(answr.eq.'Y') minvrt=1 D000400
      if(answr.eq.'y') minvrt=1 D000410
      if(answr.eq.'YES') minvrt=1 D000420
      if(answr.eq.'Yes') minvrt=1 D000430
      if(answr.eq.'yes') minvrt=1 D000440
      read(4,'(bn,40x,i5)') nunits D000450
      read(4,'(bn,40x,f10.2)') htstft D000460
      read(4,'(bn,40x,f10.2)') htbaft D000470
      read(4,'(bn,40x,f10.2)') fmgrmn D000480
      fmgmmn=fgtrgm*fmgrrn D000490
      read(4,'(bn,40x,f10.2)') ezndeg D000500
      close(4) D000510
C D000520
      donor lethality input D000530
      open(4,file=dfile,status='old') D000540
      read(4,*) elzdeg,azsdeg D000550
      azsrad=degrad*azsdeg D000560
      read(4,*) nfmlts,mdosh D000570

```

```

if(mdosh.gt.0) then
    read(4,*) dshin
    dshcm=2.54*dshin
    read(4,*) ccds, athds, bthds, cthds
    tcds=10.0**ccds
    read(4,*) ccdsm, athdsm, bthdsm, cthdsm
    tcdsm=10.0**ccdsm
endif
read(4,*) (fmult(ifm), ifm=1, nfm)
fnuaz=float(nunits)*azsdeg
ifrag=1
100 read(4,*,end=200) poldeg, fmgr, tfamil, tvfifs, tfarma
if(minvrt.eq.0) then
    tezdeg=90.0-poldeg
else
    tezdeg=poldeg-90.0
endif
if(tezdeg.lt.ezndeg) go to 100
tfmgm=0.06479891*fmgr
tfacm2=fi2tc2*tfamil*fgrtlb*fmgr
if(mdosh.gt.0) then
    call vmthr(dshcm, tfacm2, tvfifs, tfmgm, tcds, athds, bthds, cthds,
+          tcdsm, athdsm, bthdsm, cthdsm, tvfifs, tfmgm)
    if(tvfifs.le.0.0) go to 100
endif
if(tfmgm.lt.fmgmmn) go to 100
if(ifrag.gt.900) then
    write(*, ' (a,i3,a)')
+ ' Data file contains more than ',900,' fragments!'
    stop 'Computation Terminated'
endif
ezldeg(ifrag)=tezdeg
fmgm(ifrag)=tfmgm
famil(ifrag)=tfamil
vfifs(ifrag)=tvfifs
farma(ifrag)=tfarma
fatn(ifrag)=fnuaz*fmult(int(poldeg/elzdeg))
facm2(ifrag)=tfacm2
fdmm(ifrag)=2.0*sqrt(100.0*tfacm2/pi)
frkec(ifrag)=fmgr/tglbgr
ifrag=ifrag+1
go to 100
200 nfrags=ifrag-1
close(4)
c
c    report return
c    if(merror.gt.0) write(2, '(4x,a)') 'returning from donin'
c
c    return
c
c    end
c
c*****
c*****

```

```

C***** A000010
C A000020
  subroutine accin(afile) A000030
C A000040
  obtains acceptor input A000050
C A000060
  common blocks A000070
common/error/ meror,irerr,iferr,merout A000080
common/names/ dname,aname A000090
common/acptr/ htacft,wdacft,dpacft,htacfm A000100
common/acflg/ nweps,macwh,macrm,macsh,mstor A000110
common/geowp/ wplin,wpdin A000120
common/geopa/ pasin,pahin,pasft A000130
common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin A000140
common/jrdtx/ abjrx,cjrx A000150
common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw A000160
common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin A000170
common/jrdtp/ abjrp,cjrp A000180
common/thorr/ athr,bthr,cthr,dthr,vrrms,tcv A000190
common/geoas/ ashcm A000200
common/thras/ athasv,bthasv,cthasv,tcasv,
+ athasm,bthasm,cthasm,tcasm A000220
common/mechd/ wpke A000230
C A000240
  type declarations A000250
character dname*40,aname*40 A000260
character afile*25 A000270
C A000280
  report call A000290
  if(meror.gt.0) write(2,'(4x,a)') 'accin called' A000300
C A000310
  user specification input A000320
  open(4,file='fp.acc',status='old') A000330
  read(4,'(40x,a)') aname A000340
  read(4,'(40x,a)') afile A000350
  read(4,'(bn,40x,f10.2)') htacft A000360
  htacfm=htacft-0.2 A000370
  read(4,'(bn,40x,f10.2)') wdacft A000380
  read(4,'(bn,40x,f10.2)') dpacft A000390
  close(4) A000400
C A000410
  acceptor vulnerability input A000420
  open(4,file=afile,status='old') A000430
  read(4,*) nweps,macwh,macrm,macsh,mstor A000440
  read(4,*) wplin,wpdin A000450
  read(4,*) pasin,pahin A000460
  pasft=pasin/12. A000470
  if(macwh.ne.0) then A000480
    read(4,*) whlin,whdin,whhin,exlin A000490
    whhcm=2.54*whhin A000500
    whhmm=10.0*whhcm A000510
    read(4,*) ajrx,bjrxpl,cjrx A000520
    abjrx=ajrx*bjrxpl A000530
    read(4,*) ccw,athw,bthw,cthw,dthw,vrwms A000540
    tcw=10.0**ccw A000550
  endif A000560
  if(macrm.ne.0) then A000570
    read(4,*) rmlin,rmdin,rmhin,prlin A000580
    rmhcm=2.54*rmhin A000590
    rmhmm=10.0*rmhcm A000600
    read(4,*) ajrp,bjrppl,cjrp A000610
    abjrp=ajrp*bjrppl A000620

```

```

        read(4,*) ccr,athr,bthr,cthr,dthr,vrrms      A000630
        tcr=10.0**ccr                                A000640
    endif                                             A000650
    if(macsh.ne.0) then                               A000660
        read(4,*) ashin                              A000670
        ashcm=2.54*ashin                             A000680
        read(4,*) ccasv,athasv,bthasv,cthasv         A000690
        tcasv=10.0**ccasv                            A000700
        read(4,*) ccasm,athasm,bthasm,cthasm         A000710
        tcasm=10.0**ccasm                            A000720
    endif                                             A000730
    read(4,*) wpke                                    A000740
    close(4)                                          A000750
c                                                    A000760
c    report return                                  A000770
c    if(meror.gt.0) write(2,'(4x,a)') 'returning from accin' A000780
c                                                    A000790
c    return                                          A000800
c                                                    A000810
c    end                                             A000820
c                                                    A000830
c***** A000840

c***** R000010
c                                                    R000020
c    subroutine rngin                                R000030
c                                                    R000040
c    obtains range input                            R000050
c                                                    R000060
c    common blocks                                  R000070
c    common/error/ meror,irerr,iferr,merout          R000080
c    common/rangs/ nrsegs,nrsegp                    R000090
c    common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin R000100
c    common/arang/ rangem(97),ahfrn(97),ahtop(97)    R000110
c                                                    R000120
c    report call                                     R000130
c    if(meror.gt.0) write(2,'(2x,a)') 'rngin called' R000140
c                                                    R000150
c    user specification input                        R000160
c    open(4,file='fp.rng',status='old')              R000170
c    read(4,'(bn,40x,f10.2)') rngmax                 R000180
c    read(4,'(bn,40x,i5)') nrsegs                     R000190
c    nrsegs=min(nrsegs,96)                           R000200
c    nrsegp=nrsegs+1                                  R000210
c    rngseg=rngmax/float(nrsegs)                      R000220
c    rngsgh=0.5*rngseg                                R000230
c    rngscl=1.0e-2*rngseg                             R000240
c    rngsix=0.6*rngseg                                R000250
c    do 100 irange=1,nrsegp                           R000260
c        rangem(irange)=rngseg*(float(irange)-0.5)    R000270
100 continue                                         R000280
c    close(4)                                          R000290
c                                                    R000300
c    report return                                  R000310
c    if(meror.gt.0) write(2,'(4x,a)') 'returning from rngin' R000320
c                                                    R000330
c    return                                          R000340
c                                                    R000350
c    end                                             R000360
c                                                    R000370
c***** R000380

```

```

C***** D000010
C      subroutine doacc D000020
C      computes parameters requiring both donor and acceptor input D000030
C      common blocks D000040
C      common/error/ meror,irerr,iferr,merout D000050
C      common/repls/ nreps,fnreps D000060
C      common/zones/ elzdeg,azsdeg,azsrad D000070
C      common/frags/ nfrags,tfrags D000080
C      common/fragd/ fam1(900),vfifs(900),farma(900), D000090
+      fatn(900),ezldeg(900) D000100
C      common/fragc/ facm2(900),fdmm(900),fmgm(900) D000110
C      common/fragx/ vcrdx(900),vcrbx(900) D000120
C      common/fragp/ vcrdp(900),vcrbp(900) D000130
C      common/acflg/ nweps,macwh,macrm,macsh,mstor D000140
C      common/acptr/ htacft,wdacft,dpacft,htacfm D000150
C      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin D000160
C      common/jrdtx/ abjrx,cjrx D000170
C      common/thorw/ athw,bthw,cthw,dthw,vwrms,tcw D000180
C      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin D000190
C      common/thorr/ athr,bthr,cthr,dthr,vrrms,tcv D000200
C      common/jrdtp/ abjrp,cjrp D000210
C      common/rangs/ nrsegs,nrsegp D000220
C      common/range/ rngmax,rngseg,rngsqh,rngscl,rngsix,rngmin D000230
C      common/arang/ rangem(97),ahfrn(97),ahtop(97) D000240
C      report call D000250
C      if(meror.gt.0) write(2,'(2x,a)') 'doacc called' D000260
C      compute the total number of fragment trajectories D000270
C      tfrags=float(nreps*nfrags) D000280
C      compute the minimum range and range area arrays D000290
C      tahazs=tan(0.5*azsrad) D000300
C      if(tahazs.gt.0.0) then D000310
C      rngmin=wdacft/(2.0*tahazs) D000320
C      else D000330
C      rngmin=0.0 D000340
C      endif D000350
C      do 100 irange=1,nrsegs D000360
C      rangeo=rangem(irange)+rngsqh D000370
C      rangei=rangem(irange)-rngsqh D000380
C      ahfrn(irange)=azsrad*htacft*rangem(irange) D000390
C      ahtop(irange)=0.5*azsrad*(rangeo*rangeo-rangei*rangei) D000400
100 continue D000410
C      predetermine parameters characterizing D000420
C      fragment lethality against unshrouded acceptor D000430
C      if(macsh.eq.0) then D000440
C      if(macwh.ne.0) then D000450
C      a warhead component is present D000460
C      do 200 ifrag=1,nfrags D000470
C      vcrdx(ifrag)=vcrjif(abjrx,cjrx,whhmm,fdmm(ifrag)) D000480
C      vcrbx(ifrag)= D000490
+      vcrth(tcw,athw,bthw,whhcm,facm2(ifrag),fmgm(ifrag)) D000500
200 continue D000510
C      endif D000520
C      if(macrm.ne.0) then D000530
C      a rocket motor component is present D000540
C      do 300 ifrag=1,nfrags D000550
C      vcrdp(ifrag)=vcrjif(abjrp,cjrp,whhmm,fdmm(ifrag)) D000560
C      vcrbp(ifrag)= D000570
+      vcrth(tcr,athr,bthr,rmhcm,facm2(ifrag),fmgm(ifrag)) D000580
300 continue D000590
C      endif D000600
C      endif D000610
C      report return D000620
C      if(meror.gt.0) write(2,'(2x,a)') 'returning from doacc' D000630
C      return D000640
C      end D000650
C***** D000660

```

```

C***** V000010
C                                             V000020
C      subroutine vulna                                             V000030
C                                             V000040
C      computes the maximum vulnerable areas of the acceptor      V000050
C                                             V000060
C      common blocks                                             V000070
C      common/error/ meror,irerr,iferr,merout                     V000080
C      common/acflg/ nweps,macwh,macrm,macsh,mstor                V000090
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop          V000100
C                                             V000110
C      report call                                             V000120
C      if(meror.gt.0) write(2,'(2x,a)') 'vulna called'           V000130
C                                             V000140
C      initialize the vulnerable areas                             V000150
C      amfrn=0.0                                                 V000160
C      amtop=0.0                                                 V000170
C      axfrn=0.0                                                 V000180
C      axtop=0.0                                                 V000190
C      apfrn=0.0                                                 V000200
C      aptop=0.0                                                 V000210
C                                             V000220
C      if(mstor.eq.1) then                                       V000230
C                                             V000240
C      the storage arrangement is vertical                         V000250
C      call vulnv(ucfrn,uctop)                                    V000260
C                                             V000270
C      if a warhead component is present                          V000280
C      if(macwh.ne.0) call vulvw(ucfrn,uctop)                    V000290
C                                             V000300
C      if a rocket motor component is present                    V000310
C      if(macrm.ne.0) call vulvm(ucfrn,uctop)                    V000320
C                                             V000330
C      else                                                       V000340
C                                             V000350
C      the storage arrangement is horizontal                      V000360
C      call vulnh(ucfrn,uctop)                                    V000370
C                                             V000380
C      amin2=0.0                                                 V000390
C                                             V000400
C      if a warhead component is present                          V000410
C      if(macwh.ne.0) call vulhw(amin2,ucfrn,uctop)              V000420
C                                             V000430
C      if a rocket motor component is present                    V000440
C      if(macrm.ne.0) call vulhm(amin2,ucfrn,uctop)              V000450
C                                             V000460
C      amfrn=ucfrn*amin2                                         V000470
C      amtop=uctop*amin2                                         V000480
C                                             V000490
C      endif                                                       V000500
C                                             V000510
C      report return                                             V000520
C      if(meror.gt.0) write(2,'(2x,a)') 'returning from vulna'   V000530
C                                             V000540
C      return                                                    V000550
C                                             V000560
C      end                                                        V000570
C                                             V000580
C***** V000590

```

```

C***** V000010
C      V000020
C      subroutine vulnv(ucfrn,uctop) V000030
C      V000040
C      determines the number of front and top V000050
C      units for a vertical storage arrangement V000060
C      V000070
C      common blocks V000080
C      common/error/ meror,irerr,iferr,merout V000090
C      common/acptr/ htacft,wdacft,dpacft,htacfm V000100
C      common/geowp/ wplin,wpdin V000110
C      common/geopa/ pasin,pahin,pasft V000120
C      V000130
C      report call V000140
C      if(meror.gt.0) write(2,'(4x,a)') 'vulnv called' V000150
C      V000160
C      compute the front and top unit areas in sq in V000170
C      wdpps=wpdin+pasin V000180
C      aufrn=(wplin+pahin)*wdpps V000190
C      autop=wdpps*wdpps V000200
C      V000210
C      compute the numbers of front and V000220
C      top units (with conversion to sq ft) V000230
C      wdacp=wdacft+pasft V000240
C      ucfrn=wdacp*htacft/aufrn V000250
C      uctop=wdacp*(dpacft+pasft)/autop V000260
C      V000270
C      report return V000280
C      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulnv' V000290
C      V000300
C      return V000310
C      V000320
C      end V000330
C      V000340
C***** V000350

C***** V000010
C      V000020
C      subroutine vulvw(ucfrn,uctop) V000030
C      V000040
C      determines the front and top vulnerable V000050
C      warhead areas for vertical storage V000060
C      V000070
C      common blocks V000080
C      common/error/ meror,irerr,iferr,merout V000090
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum V000100
C      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin V000110
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop V000120
C      V000130
C      report call V000140
C      if(meror.gt.0) write(2,'(4x,a)') 'vulvw called' V000150
C      V000160
C      compute the front and top maximum V000170
C      vulnerable warhead areas in sq ft V000180
C      amfrn=ucfrn*whlin*whdin V000190
C      amtop=uctop*qpi*whdin*whdin V000200
C      V000210
C      compute the front and top maximum V000220
C      vulnerable explosive areas in sq ft V000230
C      exdin=whdin-whhin V000240
C      axfrn=ucfrn*exlin*exdin V000250
C      axtop=uctop*qpi*exdin*exdin V000260
C      V000270
C      report return V000280
C      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulvw' V000290
C      V000300
C      return V000310
C      V000320
C      end V000330
C      V000340
C***** V000350

```

```

C***** V000010
C      V000020
C      subroutine vulvm(ucfrn,uctop) V000030
C      V000040
C      determines the front and top vulnerable V000050
C      rocket motor areas for vertical storage V000060
C      (there are no top vulnerable areas if a warhead is also present) V000070
C      V000080
C      common blocks V000090
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum V000100
C      common/error/ meror,irerr,iferr,merout V000110
C      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin V000120
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop V000130
C      V000140
C      report call V000150
C      if(meror.gt.0) write(2,'(4x,a)') 'vulvm called' V000160
C      V000170
C      compute the front and top maximum V000180
C      vulnerable motor areas in sq ft V000190
C      amfrn=amfrn+ucfrn*rmlin*rmdin V000200
C      if(amtop.eq.0.0) amtop=uctop*qpi*rmdin*rmdin V000210
C      V000220
C      compute the front and top maximum V000230
C      vulnerable propellant areas in sq ft V000240
C      prdin=rmdin-rmhin V000250
C      apfrn=ucfrn*prlin*prdin V000260
C      if(axtop.eq.0.0) aptop=uctop*qpi*prdin*prdin V000270
C      V000280
C      report return V000290
C      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulvm' V000300
C      V000310
C      return V000320
C      V000330
C      end V000340
C      V000350
C***** V000360

C***** V000010
C      V000020
C      subroutine vulnh(ucfrn,uctop) V000030
C      V000040
C      determines the number of front and top units for a V000050
C      horizontal storage arrangement (nweps weapons high) V000060
C      V000070
C      common blocks V000080
C      common/error/ meror,irerr,iferr,merout V000090
C      common/acptr/ htacft,wdacft,dpacft,htacfm V000100
C      common/acflg/ nweps,macwh,macrm,macsh,mstor V000110
C      common/geowp/ wplin,wpdin V000120
C      common/geopa/ pasin,pahin,pasft V000130
C      V000140
C      report call V000150
C      if(meror.gt.0) write(2,'(4x,a)') 'vulnh called' V000160
C      V000170
C      compute the top and front unit areas in sq in V000180
C      autop=wplin*(wpdin+pasin) V000190
C      aufrn=autop+wplin*pahin/float(nweps) V000200
C      V000210
C      compute the numbers of front and V000220
C      top units (with conversion to sq ft) V000230
C      ucfrn=wdacft*(htacft+pasft)/aufrn V000240
C      uctop=wdacft*(dpacft+pasft)/autop V000250
C      V000260
C      report return V000270
C      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulnh' V000280
C      V000290
C      return V000300
C      V000310
C      end V000320
C      V000330
C***** V000340

```

```

C***** V000010
C      V000020
C      subroutine vulhw(amin2,ucfrn,uctop) V000030
C      V000040
C      determines the front and top vulnerable V000050
C      warhead areas for horizontal storage V000060
C      V000070
C      common blocks V000080
C      common/error/ meror,irerr,iferr,merout V000090
C      common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin V000100
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop V000110
C      V000120
C      report call V000130
C      if(meror.gt.0) write(2,'(4x,a)') 'vulhw called' V000140
C      V000150
C      compute the front and top maximum V000160
C      vulnerable warhead areas in sq ft V000170
C      amin2=whlin*whdin V000180
C      V000190
C      compute the front and top maximum V000200
C      vulnerable explosive areas in sq ft V000210
C      axin2=(whdin-whhin)*exlin V000220
C      axfrn=ucfrn*axin2 V000230
C      axtop=uctop*axin2 V000240
C      V000250
C      report return V000260
C      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulhw' V000270
C      V000280
C      return V000290
C      V000300
C      end V000310
C      V000320
C***** V000330

C***** V000010
C      V000020
C      subroutine vulhm(amin2,ucfrn,uctop) V000030
C      V000040
C      determines the front and top vulnerable V000050
C      rocket motor areas for horizontal storage V000060
C      V000070
C      common blocks V000080
C      common/error/ meror,irerr,iferr,merout V000090
C      common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin V000100
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop V000110
C      V000120
C      report call V000130
C      if(meror.gt.0) write(2,'(4x,a)') 'vulhm called' V000140
C      V000150
C      compute the front and top maximum V000160
C      vulnerable motor areas in sq ft V000170
C      amin2=amin2+rmlin*rmdin V000180
C      V000190
C      compute the front and top maximum V000200
C      vulnerable propellant areas in sq ft V000210
C      apin2=(rmdin-rmhin)*prlin V000220
C      apfrn=apfrn+ucfrn*apin2 V000230
C      aptop=aptop+uctop*apin2 V000240
C      V000250
C      report return V000260
C      if(meror.gt.0) write(2,'(4x,a)') 'returning from vulhm' V000270
C      V000280
C      return V000290
C      V000300
C      end V000310
C      V000320
C***** V000330

```

```

C***** 0000010
C      subroutine outin(rndsav,dfile,minvrt,afile) 0000020
C      outputs problem input 0000030
C      common blocks 0000040
C      common/repls/ nreps,fnreps 0000050
C      common/frmin/ fmgrmn,fmgmmn,ezndeg 0000060
C      common/zones/ elzdeg,azsdeg,azsrad 0000070
C      common/prmtr/ slcmin,slcmax,altmin,altmax, 0000080
+      wspmin,wspmax,wdrmin,wdrmax 0000090
C      common/donor/ nunits,htstft,htbaft 0000100
C      common/frags/ nfrags,tfrags 0000110
C      common/rangs/ nrsegs,nrsegs 0000120
C      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin 0000130
C      common/names/ dname,aname 0000140
C      common/acptr/ htacft,wdacft,dpacft,htacfm 0000150
C      type declarations 0000160
C      character dname*40,aname*40 0000170
C      character dfile*25,afile*25 0000180
C      double precision rndsav 0000190
C      open(3,file='fp.out',status='unknown') 0000200
C      write(3,'(//33x,a)') '***FRAGPROP***' 0000210
C      write(3,'(//14x,a)') 0000220
+      'REACTION PROPAGATION PROGRAM FOR FRAGMENTING MUNITIONS' 0000230
C      write(3,'(//33x,a)') '***PARAMETERS***' 0000240
C      call inout(' NUMBER OF REPLICATIONS: ',nreps) 0000250
C      call flout(' MINIMUM SOIL CONSTANT: ',slcmin,' ') 0000260
C      call flout(' MAXIMUM SOIL CONSTANT: ',slcmax,' ') 0000270
C      call flout(' MINIMUM ALTITUDE: ',altmin,' ft ') 0000280
C      call flout(' MAXIMUM ALTITUDE: ',altmax,' ft ') 0000290
C      call flout(' MINIMUM WIND SPEED: ',wspmin,' mph ') 0000300
C      call flout(' MAXIMUM WIND SPEED: ',wspmax,' mph ') 0000310
C      call flout(' MINIMUM WIND DIRECTION: ',wdrmin,' degrees') 0000320
C      call flout(' MAXIMUM WIND DIRECTION: ',wdrmax,' degrees') 0000330
C      write(3,'(//32x,a)') '***RANDOMIZATION***' 0000340
C      write(3,'(24x,a,i10)') ' MONTE CARLO SEED: ',idint(rndsav) 0000350
C      write(3,'(//36x,a)') '***DONOR***' 0000360
C      write(3,'(24x,a,a)') ' DESCRIPTION: ',dname 0000370
C      write(3,'(24x,a,a)') ' LETHALITY DATA FILE: ',dfile 0000380
C      if(minvrt.eq.0) then 0000390
C      write(3,'(24x,a)') ' ORIENTATION: Erect' 0000400
C      else 0000410
C      write(3,'(24x,a)') ' ORIENTATION: Inverted' 0000420
C      endif 0000430
C      call inout(' NUMBER OF UNITS: ',nunits) 0000440
C      call flout(' HEIGHT OF STACK: ',htstft,' ft ') 0000450
C      call flout(' BASE ELEVATION: ',htbaft,' ft ') 0000460
C      call inout(' NUMBER OF FRAGMENTS: ',nfrags) 0000470
C      call flout(' MINIMUM FRAGMENT MASS: ',fmgrmn,' grains ') 0000480
C      call flout(' MINIMUM ELEVATION ANGLE: ',ezndeg,' degrees') 0000490
C      write(3,'(//36x,a)') '***RANGE***' 0000500
C      call flout(' MINIMUM RANGE: ',rngmin,' ft ') 0000510
C      call flout(' MAXIMUM RANGE: ',rngmax,' ft ') 0000520
C      call inout(' NUMBER OF SEGMENTS: ',nrsegs) 0000530
C      write(3,'(//34x,a)') '***ACCEPTOR***' 0000540
C      write(3,'(20x,a,a)') ' DESCRIPTION: ',aname 0000550
C      write(3,'(20x,a,a)') ' VULNERABILITY DATA FILE: ',afile 0000560
C      call flout(' HEIGHT OF STACK: ',htacft,' ft ') 0000570
C      call flout(' WIDTH OF STACK: ',wdacft,' ft ') 0000580
C      call flout(' DEPTH OF STACK: ',dpacft,' ft ') 0000590
C      close(3) 0000600
C      return 0000610
C      end 0000620
C***** 0000630

```

```

C***** F000010
C                                             F000020
C      subroutine flout(alabel,flvar,aunit)      F000030
C                                             F000040
C      outputs a floating point variable      F000050
C      with an alphanumeric label and units    F000060
C                                             F000070
C      type declarations      F000080
C      character alabel*25,aunit*8      F000090
C                                             F000100
C      aflvar=abs(flvar)      F000110
C      if(aflvar.lt.1.0) then      F000120
C        if(flvar.lt.0.0) then      F000130
C          write(3,'(20x,2a,f2.1,a)') alabel,' -0',aflvar,aunit      F000140
C        else      F000150
C          write(3,'(20x,2a,f2.1,a)') alabel,' 0',flvar,aunit      F000160
C        endif      F000170
C      else      F000180
C        write(3,'(20x,a,f6.1,a)') alabel,flvar,aunit      F000190
C      endif      F000200
C                                             F000210
C      return      F000220
C                                             F000230
C      end      F000240
C                                             F000250
C***** F000260

C***** I000010
C                                             I000020
C      subroutine inout(alabel,invar)      I000030
C                                             I000040
C      outputs an integer variable with an alphanumeric label      I000050
C                                             I000060
C      type declaration      I000070
C      character alabel*25      I000080
C                                             I000090
C      write(3,'(20x,a,i4)') alabel,invar      I000100
C                                             I000110
C      return      I000120
C                                             I000130
C      end      I000140
C                                             I000150
C***** I000160

C***** A000010
C                                             A000020
C      subroutine arini(nrange,v,vd,vb,vm,value)      A000030
C                                             A000040
C      initializes variable arrays      A000050
C                                             A000060
C      local arrays      A000070
C      dimension v(97),vd(97),vb(97),vm(97)      A000080
C                                             A000090
C      do 100 irange=1,nrange      A000100
C        v(irange)=value      A000110
C        vd(irange)=value      A000120
C        vb(irange)=value      A000130
C        vm(irange)=value      A000140
C      100 continue      A000150
C                                             A000160
C      return      A000170
C                                             A000180
C      end      A000190
C                                             A000200
C***** A000210

```

```

C***** E000010
C      E000020
C      subroutine ertim(ihbgn,tsbgn,frcmp,timela,uniela,timrem,unirem) E000030
C      E000040
C      determines the elapsed time and estimates the remaining time E000050
C      E000060
C      type declaration E000070
C      character*8 uniela,unirem E000080
C      E000090
C      obtain the current time E000100
C      call gettim(ihnow,imnow,isnow,ilnow) E000110
C      E000120
C      compute the elapsed time E000130
C      if(ihnow.lt.ihbgn) ihnow=ihnow+24 E000140
C      tsnow=3600.0*float(ihnow)+60.0*float(imnow) E000150
C      +float(isnow)+float(ilnow)/100.0 E000160
C      timela=tsnow-tsbgn E000170
C      E000180
C      estimate the remaining time E000190
C      timrem=(1.0/frcmp-1.0)*timela E000200
C      E000210
C      convert the time units E000220
C      call ctime(timela,uniela) E000230
C      call ctime(timrem,unirem) E000240
C      E000250
C      return E000260
C      E000270
C      end E000280
C      E000290
C***** E000300

C***** C000010
C      C000020
C      subroutine ctime(time,units) C000030
C      C000040
C      converts time units C000050
C      C000060
C      type declaration C000070
C      character*8 units C000080
C      C000090
C      if(time.gt.60.0) then C000100
C      time=time/60.0 C000110
C      if(time.gt.60.0) then C000120
C      time=time/60.0 C000130
C      units=' hours ' C000140
C      else C000150
C      units=' minutes' C000160
C      endif C000170
C      else C000180
C      units=' seconds' C000190
C      endif C000200
C      C000210
C      return C000220
C      C000230
C      end C000240
C      C000250
C***** C000260

```

```

C***** R000010
C R000020
  subroutine rpini(rndnxt, slcnst,asldeg,alttud,wspeed,wdrdeg, R000030
+      wdrad,jxrng,jrkmax) R000040
C R000050
  initializes replication loop R000060
C R000070
  common blocks R000080
  common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum R000090
  common/error/ meror,irerr,iferr,merout R000100
  common/prmtr/ slcmin,slcmax,altmin,altmax, R000110
+      wspmin,wspmax,wdrmin,wdrmax R000120
  common/rangs/ nrsegs,nrsegs R000130
  common/pmiss/ pn(97),pnd(97),pnb(97),pnm(97) R000140
C R000150
  type declaration R000160
  double precision rndnxt R000170
C R000180
  report call R000190
  if(merout.gt.0) write(2,'(2x,a)') 'rpini called' R000200
C R000210
  randomize parameters R000220
  slcnst=rnval(slcmin,slcmax,rndnxt) R000230
  asldeg=10.8*slcnst**0.38 R000240
  alttud=rnval(altmin,altmax,rndnxt) R000250
  wspeed=rnval(wspmin,wspmax,rndnxt) R000260
  wdrdeg=rnval(wdrmin,wdrmax,rndnxt) R000270
  wdrad=degrad*wdrdeg R000280
C R000290
  set integration parameters R000300
  if((wspeed.eq.0.0).or.(wdrdeg.eq.0.0)) then R000310
    jxrng=0 R000320
    jrkmax=4 R000330
  else R000340
    jxrng=1 R000350
    jrkmax=6 R000360
  endif R000370
C R000380
  initialize the miss probabilities R000390
  call arini(nrsegs,pn,pnd,pnb,pnm,1.0) R000400
C R000410
  report return R000420
  if(merout.gt.0) write(2,'(2x,a)') 'returning from rpini' R000430
C R000440
  return R000450
C R000460
  end R000470
C R000480
C***** R000490

```

```

C***** F000010
C      subroutine frini(irep,ifrag,rndnxt, irico,lrange,htfrft,mfrht1, F000020
+      vfrag,seldeg,selrad,aeldeg,aelrad,cd1,cd2,cd3, F000030
+      deldst) F000040
C      F000050
C      initializes fragment loop F000060
C      F000070
C      F000080
C      common blocks F000090
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum F000100
C      common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2, F000110
+      hcam3 F000120
C      common/cnstu/ fgrrlb,fgmtlb,fgrrgm,ffttm,ffttmm,fmftt,fi2tc2 F000130
C      common/error/ meror,irerr,iferr,merout F000140
C      common/zones/ elzdeg,azsdeg,azsrad F000150
C      common/donor/ nunits,htstft,htbaft F000160
C      common/fragd/ famil(900),vfifs(900),farma(900), F000170
+      fatn(900),ezldeg(900) F000180
C      common/acptr/ htacft,wdacft,dpacft,htacfm F000190
C      common/rkvar/ rk(7) F000200
C      F000210
C      type declaration F000220
C      double precision rndnxt F000230
C      F000240
C      set the error output flag F000250
C      merout=0 F000260
C      if(meror.gt.0) then F000270
C      if(irep.gt.irerr) then F000280
C      if(ifrag.gt.iferr) then F000290
C      merout=1 F000300
C      endif F000310
C      endif F000320
C      endif F000330
C      F000340
C      report call F000350
C      if(merout.gt.0) F000360
+      write(2,'(2x,a,i3)') 'frini called at fragment ',ifrag F000370
C      F000380
C      initialize the ricochet count and last range segment index F000390
C      irico=0 F000400
C      lrange=0 F000410
C      F000420
C      randomize the initial fragment height F000430
C      htfrft=rndval(htbaft,htstft,rndnxt) F000440
C      F000450
C      set switch 1 for fragment height F000460
C      if(htfrft.ge.htacft) then F000470
C      mfrht1=1 F000480
C      else F000490
C      mfrht1=0 F000500
C      endif F000510
C      F000520
C      randomize the initial fragment velocity F000530
C      rndlog=sqrt(-2.*log(rndom(rndnxt))) F000540
C      rndnum=tpi*rndom(rndnxt) F000550
C      rndcos=rndlog*cos(rndnum) F000560
C      rndsin=rndlog*sin(rndnum) F000570
C      vfrag=vfifs(ifrag)+vfifs(ifrag)*0.035*rndcos F000580
C      F000590
C      randomize the initial fragment elevation angle F000600
C      seldeg=(ezldeg(ifrag)+elzdeg*rndom(rndnxt)) F000610
C      if(seldeg.gt.89.99) then F000620
C      seldeg=89.99 F000630
C      else F000640
C      if((seldeg.lt.1.0e-2).and.(seldeg.ge.0.0)) then F000650
C      seldeg=1.0e-2 F000660
C      else F000670
C      if((seldeg.lt.0.0).and.(seldeg.gt.-1.0e-2)) seldeg=-1.0e-2 F000680
C      endif F000690
C      endif F000700
C      selrad=degrad*seldeg F000710
C      aeldeg=abs(seldeg) F000720

```

```

c      aelrad=abs(selrad)
c
c      randomize drag coefficient parameters
cdmax=1.75*farma(ifrag)-1.27
cdmin=0.66*farma(ifrag)-0.26
cd0=rnval(cdmin,cdmax,rndnxt)
cd1=cd0+0.2
cd2=cd0+0.65
cd3=cd0+0.5
c
c      initialize the integration step
call dstep(htfrft,seldeg,selrad, deldst)
c
c      initialize the integration variables
rkv(1)=0.0
rkv(2)=vfrag*cos(selrad)
rkv(3)=0.0
rkv(4)=vfrag*sin(selrad)
rkv(5)=htfrft
rkv(6)=0.0
rkv(7)=0.0
c
c      report return
if(merout.gt.0) write(2,'(2x,a)') 'returning from frini'
c
c      return
c
c      end
c
c*****
c*****
c      subroutine dstep(htfrft,seldeg,selrad, deldst)
c
c      determiness integration distance step
c
c      common blocks
common/error/ meror,irerr,iferr,merout
common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin
c
c      report call
if(merout.gt.0) write(2,'(4x,a)') 'dstep called'
c
c      if(seldeg.lt.0.0) then
dtognd=htfrft/sin(-selrad)
if(dtognd.gt.rngsix) then
deldst=rngsgh
else
deldst=aint(dtognd/1.2)
endif
else
if(seldeg.gt.70.0) then
deldst=rngsgh
else
deldst=rngscl*aint((-0.02492*seldeg+2.20134)*seldeg+18.8306)
endif
endif
c
c      report return
if(merout.gt.0) write(2,'(4x,a)') 'returning from dstep'
c
c      return
c
c      end
c
c*****
c*****

```

```

C***** T000010
C T000020
C      subroutine tstep(deldst,irico,vfrag,aelrad, deltim) T000030
C T000040
C      determiness integration time step T000050
C T000060
C      common blocks T000070
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum T000080
C      common/error/ meror,irerr,iferr,merout T000090
C      common/acptr/ htacft,wdacft,dpacft,htacfm T000100
C      common/rkvar/ rk(7) T000110
C T000120
C      report call T000130
C      if(merout.gt.0) write(2,'(2x,a)') 'tstep called' T000140
C T000150
C      compute the standard time step T000160
C      deltim=deldst/vfrag T000170
C T000180
C      compute a special time step if approaching T000190
C      the top of thr hazard volume or the ground T000200
C      if(rkv(4).lt.0.0) then T000210
C T000220
C          elerad=max(aelrad,smlrad) T000230
C          deldsp=deldst+htacfm T000240
C T000250
C          if the fragment is below the top of the hazard volume T000260
C          if(rkv(5).le.htacft) then T000270
C T000280
C              compute a time step approaching the ground T000290
C              dtognd=rkv(5)/sin(elerad) T000300
C              if(dtognd.lt.deldsp) deltim=(dtognd-htacfm)/vfrag T000310
C              if(dtognd.lt.htacft) deltim=(dtognd-0.3)/vfrag T000320
C              if(dtognd.lt.0.4 ) deltim=dtognd/vfrag+2.0e-6 T000330
C T000340
C          else T000350
C T000360
C              if the fragment is low enough T000370
C              if(rkv(5).lt.(deldst+12.0)) then T000380
C T000390
C                  compute a time step approaching T000400
C                  the top of thr hazard volume T000410
C                  dtotop=(rkv(5)-htacft)/sin(elerad) T000420
C                  if(dtotop.lt.deldsp) deltim=(dtotop-htacfm)/vfrag T000430
C                  if(dtotop.lt.htacft) deltim=(dtotop-0.3)/vfrag T000440
C                  if(dtotop.lt.0.4 ) deltim=dtotop/vfrag+2.0e-6 T000450
C T000460
C              endif T000470
C T000480
C          endif T000490
C T000500
C      endif T000510
C T000520
C      limit the time step T000530
C      if((vfrag.lt.100.0).and.(rkv(4).gt.-30.0).and.(deltim.gt.0.1)) T000540
C      + deltim=0.1 T000550
C      if((irico.gt.1).and.(vfrag.lt.150.0).and.(deltim.gt.0.08)) T000560
C      + deltim=0.08 T000570
C T000580
C      report return T000590
C      if(merout.gt.0) write(2,'(2x,a)') 'returning from tstep' T000600
C T000610
C      return T000620
C T000630
C      end T000640
C T000650
C***** T000660

```

```

C***** R000010
C R000020
  subroutine rkint (wspeed,wdrdeg,wdrad,jxrng,jrkmax,cd1,cd2,cd3, R000030
+      deltim,alttud,ifrag) R000040
C R000050
C   performs Runge-Kutta integration R000060
C R000070
C   common blocks R000080
  common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum R000090
  common/cnstp/ gg,tglbgr,hrair0,racnst,vsound,hccnst,hcam1,hcam2, R000100
+      hcam3 R000110
  common/error/ meror,irerr,iferr,merout R000120
  common/fragd/ famil(900),vfifs(900),farma(900), R000130
+      fatn(900),ezldeg(900) R000140
  common/rkvar/ rk(7) R000150
C R000160
C   local arrays R000170
  dimension rko(7),rks(7),rk(4,7) R000180
C R000190
C   report call R000200
  if(merout.gt.0) write(2,'(2x,a)') 'rkint called' R000210
C R000220
C   begin Runge-Kutta outer loop R000230
  do 200 irk=1,4 R000240
C R000250
    if(irk.lt.3) then R000260
      rkcnst=0.5 R000270
    else R000280
      rkcnst=1. R000290
    endif R000300
C R000310
C   determine relative wind R000320
  if(wspeed.le.0.0) then R000330
    vfrel=sqrt(rkv(2)*rkv(2)+rkv(4)*rkv(4)) R000340
    aayrad=atan(rkv(4)/rkv(2)) R000350
    aaxrad=hpi-aayrad R000360
  else R000370
    if(wdrdeg.eq.0.0) then R000380
      vxfrel=rkv(2)-wspeed R000390
      vyfrel=rkv(4) R000400
      vfrel=sqrt(vxfrel*vxfrel+vyfrel*vyfrel) R000410
      aayrad=atan(vyfrel/abs(vxfrel)) R000420
      aaxrad=atan(vxfrel/abs(vyfrel)) R000430
    else R000440
      vxfrel=rkv(2)-wspeed*cos(wdrad) R000450
      vyfrel=rkv(4) R000460
      vzfrel=wspeed*sin(wdrad)-rkv(6) R000470
      vfrel=sqrt(vxfrel*vxfrel+vyfrel*vyfrel+vzfrel*vzfrel) R000480
      aayrad=atan(vyfrel/sqrt(vxfrel*vxfrel+vzfrel*vzfrel)) R000490
      aaxrad=atan(vxfrel/sqrt(vyfrel*vyfrel+vzfrel*vzfrel)) R000500
      aazrad=atan(vzfrel/sqrt(vxfrel*vxfrel+vyfrel*vyfrel)) R000510
    endif R000520
  endif R000530
C R000540

```

c	determine air density, Mach number and drag coefficient	R000550
	hdelt=-((rkv(5)+alttud)+rkv(4)*0.5*deltim)	R000560
	hrhair=hrair0*expon(hdelt/racnst)	R000570
	hcam=vfrel/(vsound*expon(hdelt/hccnst))	R000580
	if(hcam.ge.hcam1) then	R000590
	if(hcam.ge.hcam2) then	R000600
	if(hcam.ge.hcam3) then	R000610
	cd=cd3	R000620
	else	R000630
	cd=cd2-0.15/(hcam3-hcam2)*(hcam-hcam2)	R000640
	endif	R000650
	else	R000660
	cd=cd2-0.45/(hcam2-hcam1)*(hcam2-hcam)	R000670
	endif	R000680
	else	R000690
	cd=cd1-0.2/(hcam1-0.1)*(hcam1-hcam)	R000700
	endif	R000710
c		R000720
c	compute velocities and accelerations	R000730
	acdrag=hrhair*vfrel*vfrel*cd*famil(ifrag)/144.	R000740
	rko(2)=-acdrag*sin(aaxrad)	R000750
	rko(3)=rkv(2)	R000760
	rko(4)=-acdrag*sin(aayrad)-gg	R000770
	rko(5)=rkv(4)	R000780
	if(jxrng.ne.0) then	R000790
	rko(6)=acdrag*sin(aazrad)	R000800
	rko(7)=rkv(6)	R000810
	endif	R000820
c		R000830
c	begin Runge-Kutta inner loop	R000840
	do 100 jrk=2,jrkmax,2	R000850
	if(.not.(irk.gt.1)) rks(jrk)=rkv(jrk)	R000860
	rk(irk,jrk)=rko(jrk)*deltim	R000870
	if(irk.eq.4) then	R000880
	rkv(jrk)=rks(jrk)+	R000890
+	(rk(1,jrk)+2*rk(2,jrk)+2*rk(3,jrk)+rk(4,jrk))/6.0	R000900
	rkv(jrk+1)=rkv(jrk+1)+	R000910
+	deltim*(rks(jrk)+(rk(1,jrk)+rk(2,jrk)+rk(3,jrk))/6.0)	R000920
	else	R000930
	rkv(jrk)=rks(jrk)+rk(irk,jrk)*rkcnst	R000940
	endif	R000950
	100 continue	R000960
c		R000970
	200 continue	R000980
c		R000990
c	report return	R001000
	if(merout.gt.0) write(2,'(2x,a)') 'returning from rkint'	R001010
c		R001020
	return	R001030
c		R001040
	end	R001050
c		R001060
c	*****	R001070

```

C***** F000010
C F000020
  subroutine frvar(vfrag,selrad,seldeg,aelrad,aeldeg,range,irange) F000030
C F000040
  determines the fragment variable values F000050
C F000060
  common blocks F000070
  common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum F000080
  common/error/ meror,irerr,iferr,merout F000090
  common/rangs/ nrsegs,nrsegg F000100
  common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin F000110
  common/rkvar/ rk(7) F000120
C F000130
  report call F000140
  if(merout.gt.0) write(2,'(2x,a)') 'frvar called' F000150
C F000160
  compute the fragment velocity F000170
  vfrag=sqrt(rk(2)*rk(2)+rk(4)*rk(4)+rk(6)*rk(6)) F000180
C F000190
  compute the elevation angle F000200
  selrad=atan(rk(4)/sqrt(rk(2)*rk(2)+rk(6)*rk(6))) F000210
  seldeg=raddeg*selrad F000220
  aelrad=abs(selrad) F000230
  aeldeg=abs(seldeg) F000240
C F000250
  compute the range and segment index F000260
  range=sqrt(rk(3)*rk(3)+rk(7)*rk(7)) F000270
  irange=int(range/rngseg)+1 F000280
  call irlim(irange,nrsegs,nrsegg) F000290
C F000300
  report return F000310
  if(merout.gt.0) write(2,'(2x,a)') 'returning from frvar' F000320
C F000330
  return F000340
C F000350
  end F000360
C F000370
C***** F000380

C***** I000010
C I000020
  subroutine irlim(irange,nrsegs,nrsegg) I000030
C I000040
  limits the value of irange I000050
C I000060
  if(irange.lt.1) irange=1 I000070
  if(irange.gt.nrsegs) irange=nrsegg I000080
C I000090
  return I000100
C I000110
  end I000120
C I000130
C***** I000140

```

```

C***** H000010
C H000020
  subroutine hazch(iframe, irange, mfrht1, mfrht2, lrange, aelrad, vfrag) H000030
C H000040
  directs hazard computations if they are required H000050
C H000060
  common blocks H000070
  common/error/ meror, irerr, iferr, merout H000080
  common/rangs/ nrsegs, nrsegs H000090
  common/range/ rngmax, rngseg, rngsgh, rngscl, rngsix, rngmin H000100
  common/acptr/ htacft, wdacft, dpacft, htacfm H000110
  common/rkvar/ rk(7) H000120
C H000130
  report call H000140
  if(merout.gt.0) write(2, '(2x,a)') 'hazch called' H000150
C H000160
  set switch 2 for fragment height H000170
  if(rk(5).gt.htacft) then H000180
    mfrht2=1 H000190
  else H000200
    mfrht2=0 H000210
  endif H000220
C H000230
  if(mfrht2.eq.1) then H000240
    if(mfrht1.ne.1) then H000250
      mfrht1=1 H000260
      delrng=(rk(5)-htacft)/tan(aelrad) H000270
      irange= H000280
+      int((sqrt(rk(3)*rk(3)+rk(7)*rk(7))-delrng)/rngseg)+1 H000290
      call irlim(irange, nrsegs, nrsegs) H000300
      call hazrd(iframe, irange, lrange, aelrad, vfrag) H000310
    endif H000320
  else H000330
    call hazrd(iframe, irange, lrange, aelrad, vfrag) H000340
  endif H000350
C H000360
  report return H000370
  if(merout.gt.0) write(2, '(2x,a)') 'returning from hazch' H000380
C H000390
  return H000400
C H000410
  end H000420
C H000430
C***** H000440

```

```

C***** H000010
C H000020
C      subroutine hazrd(iframe, irange, lrange, aelrad, vfrag) H000030
C H000040
C      performs hazard computations H000050
C H000060
C      common blocks H000070
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum H000080
C      common/error/ meror,irerr,iferr,merout H000090
C      common/zones/ elzdeg,azsdeg,azsrad H000100
C      common/fragd/ famil(900),vfifs(900),farma(900),
+      fatn(900),ezldeg(900) H000120
C      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin H000130
C      common/arang/ rangem(97),ahfrn(97),ahtop(97) H000140
C      common/acptr/ htacft,wdacft,dpacft,htacfm H000150
C      common/acflg/ nweps,macwh,macrm,macsh,mstor H000160
C      common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw H000170
C      common/thorr/ athr,bthr,cthr,dthr,vrrms,tcv H000180
C      common/thras/ athasv,bthasv,cthasv,tcasv,
+      athasm,bthasm,cthasm,tcasm H000200
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop H000210
C      common/rkvar/ rk(7) H000220
C      common/pmiss/ pn(97),pnd(97),pnb(97),pnm(97) H000230
C      common/denst/ fd(97),fdd(97),fdb(97),fdm(97) H000240
C      common/phitt/ ph(97),phd(97),phb(97),phm(97) H000250
C H000260
C      report call H000270
C      if(merout.gt.0) write(2,'(4x,a)') 'hazrd called' H000280
C H000290
C      initialize parameters if range segment index has changed H000300
C      if(irange.ne.lrange) then H000310
C          call wiini(itrpt,pn0,pn0d,pn0b,pn0m,pnr,pnrd,pnrb,pnrm,
+          fd0,fd0d,fd0b,fd0m,fdr,fdrd,fdrb,fdrm) H000330
C      endif H000340
C H000350
C      increment within segment counter H000360
C      itrpm=itrpt H000370
C      itrpt=itrpt+1 H000380
C H000390
C      compute the sine and cosine of the elevation angle H000400
C      sinele=sin(aelrad) H000410
C      cosele=cos(aelrad) H000420
C H000430
C      compute the hazard volume presented area H000440
C      apvft2=ahfrn(irange)*cosele+ahtop(irange)*sinele H000450
C H000460
C      compute the total presented area of the acceptor H000470
C      apaft2=htacft*cosele H000480
C      if(rkv(4).lt.0.0) apaft2=apaft2+dpacft*sinele H000490
C      apaft2=apaft2*wdacft H000500
C H000510
C      check and adjust fragment parameters for shroud penetration H000520
C      call shpen(iframe,vfrag, vfraz,vfcdx,vfcdp,vfcbx,vfcbp) H000530
C H000540
C      accumulate the unconditional miss probability H000550
C      call nohit(iframe,apaft2,apvft2,irange,itrpm,itrpt,pnr,pn0,pn) H000560
C H000570
C      accumulate the unconditional fragment density H000580
C      call dnsty(iframe,apvft2,irange,itrpm,itrpt,fdr,fd0,fd) H000590

```

c		H000600
c	determine the detonation vulnerable area	H000610
	avdft2=0.0	H000620
	if(macwh.ne.0) then	H000630
	call vulnd(vfcdx,vfhaz,axfrn,axtop,cosele,sinele, avdft2)	H000640
	endif	H000650
	if(macrm.ne.0) then	H000660
	call vulnd(vfcdp,vfhaz,apfrn,aptop,cosele,sinele, avdft2)	H000670
	endif	H000680
c		H000690
c	accumulate the detonation miss probability and fragment density	H000700
	call nohit(iframe,avdft2,apvft2,irange,itrpm,itrpt,pnrd,pn0d,pnd)	H000710
	if(avdft2.ne.0.0) then	H000720
	call dnsty(iframe,apvft2,irange,itrpm,itrpt,fdrd,fd0d, added)	H000730
	endif	H000740
c		H000750
c	determine the burn vulnerable area	H000760
	avbft2=0.0	H000770
	if(macwh.ne.0) then	H000780
	call vulnb(vfcbx,cthw,dthw,vrrms,vfhaz,axfrn,axtop,cosele,	H000790
+	sinele, avbft2)	H000800
	endif	H000810
	if(macrm.ne.0) then	H000820
	call vulnb(vfcbp,cthr,dthr,vrrms,vfhaz,apfrn,aptop,cosele,	H000830
+	sinele, avbft2)	H000840
	endif	H000850
c		H000860
c	accumulate the burn miss probability and fragment density	H000870
	call nohit(iframe,avbft2,apvft2,irange,itrpm,itrpt,pnrb,pn0b,pnb)	H000880
	if(avbft2.ne.0.0) then	H000890
	call dnsty(iframe,apvft2,irange,itrpm,itrpt,fdrb,fd0b, added)	H000900
	endif	H000910
c		H000920
c	determine the mechanical damage vulnerable area	H000930
	call vulnm(iframe,vfhaz,cosele,sinele, avmft2)	H000940
c		H000950
c	accumulate the mechanical damage	H000960
c	miss probability and fragment density	H000970
	call nohit(iframe,avmft2,apvft2,irange,itrpm,itrpt,pnrm,pn0m,pnm)	H000980
	if(avmft2.ne.0.0) then	H000990
	call dnsty(iframe,apvft2,irange,itrpm,itrpt,fdrm,fd0m, added)	H001000
	endif	H001010
c		H001020
	lrange=irange	H001030
c		H001040
c	report return	H001050
	if(merout.gt.0) write(2,'(4x,a)') 'returning from hazrd'	H001060
c		H001070
	return	H001080
c		H001090
	end	H001100
c		H001110
c	*****	H001120

```

C***** W000010
C      W000020
C      subroutine wiini(itrpt,pn0,pn0d,pn0b,pn0m,pnr,pnrd,pnrb,pnrm, W000030
C      +      fd0,fd0d,fd0b,fd0m,fdr,fdrd,fdrb,fdrm) W000040
C      W000050
C      initializes within segment parameters W000060
C      W000070
C      common block W000080
C      common/error/ meror,irerr,iferr,merout W000090
C      W000100
C      report call W000110
C      if(merout.gt.0) write(2,'(6x,a)') 'wiini called' W000120
C      W000130
C      initialize the within segment point index W000140
C      itrpt=0 W000150
C      W000160
C      initialize the miss probability parameters W000170
C      pn0=1.0 W000180
C      pnr=0.0 W000190
C      pn0d=1.0 W000200
C      pnrd=0.0 W000210
C      pn0b=1.0 W000220
C      pnrb=0.0 W000230
C      pn0m=1.0 W000240
C      pnrm=0.0 W000250
C      W000260
C      initialize the fragment density parameters W000270
C      fdr=0.0 W000280
C      fd0=0.0 W000290
C      fdrd=0.0 W000300
C      fd0d=0.0 W000310
C      fdrb=0.0 W000320
C      fd0b=0.0 W000330
C      fdrm=0.0 W000340
C      fd0m=0.0 W000350
C      W000360
C      report return W000370
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from wiini' W000380
C      W000390
C      return W000400
C      W000410
C      end W000420
C      W000430
C***** W000440

```

```

C***** S000010
C subroutine shpen(ifrag,vfrag, vf haz,vfcdx,vfcdp,vfcbx,vfcbp) S000020
C checks and adjusts fragment parameters for shroud penetration S000030
C S000040
C common blocks S000050
C common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum S000060
C common/cnstu/ fgtrlb,fgmtlb,fgtrgm,ffttm,ffttmm,fmtft,fi2tc2 S000070
C common/error/ meror,irerr,iferr,merout S000080
C common/fragd/ famil(900),vfifs(900),farma(900), S000090
+ fatn(900),ezldeg(900) S000100
C common/fragc/ facm2(900),fdmm(900),fmgm(900) S000110
C common/fragx/ vcrdx(900),vcrbx(900) S000120
C common/fragp/ vcrdp(900),vcrbp(900) S000130
C common/acflg/ nweps,macwh,macrm,macsh,mstor S000140
C common/geowh/ whlin,whdin,whhin,whhcm,whhmm,exlin S000150
C common/jrdtx/ abjrx,cjrx S000160
C common/thorw/ athw,bthw,cthw,dthw,vrwms,tcw S000170
C common/georm/ rmlin,rmdin,rmhin,rmhcm,rmhmm,prlin S000180
C common/jrdtp/ abjrp,cjrp S000190
C common/thorr/ athr,bthr,cthr,dthr,vrrms,tcv S000200
C common/geoas/ ashcm S000210
+ common/thras/ athasv,bthasv,cthasv,tcasv, S000220
+ athasm,bthasm,cthasm,tcasm S000230
C report call S000240
C if(merout.gt.0) write(2,'(6x,a)') 'shpen called' S000250
C S000260
C if(macsh.eq.0) then S000270
C S000280
C no shroud is present S000290
C S000300
C S000310
C assign unadjusted values to hazard fragment parameters S000320
C vf haz=vfrag S000330
C vfcdx=vcrdx(ifrag) S000340
C vfcdp=vcrdp(ifrag) S000350
C vfcbx=vcrbx(ifrag) S000360
C vfcbp=vcrbp(ifrag) S000370
C S000380
C else S000390
C S000400
C a shroud is present S000410
C S000420
C adjust the fragment velocity and mass for shroud penetration S000430
C call vmthr(ashcm,facm2(ifrag),vfrag,fmgm(ifrag),tcasv,athasv, S000440
+ bthasv,cthasv,tcasm,athasm,bthasm,cthasm,vf haz, S000450
+ fmhaz) S000460
C S000470
C compute the hazard fragment area in sq cm S000480
C fhacm2=fi2tc2*famil(ifrag)*fgmtlb*fmhaz S000490
C S000500
C compute the hazard fragment diameter in mm S000510
C fhdmm=2.0*sqrt(100.0*fhacm2/pi) S000520
C S000530
C compute the explosive detonation critical velocity S000540
C vfcdx=vcrjr(abjrx,cjrx,whhmm,fhdmm) S000550
C S000560
C compute the propellant detonation critical velocity S000570
C vfcdp=vcrjr(abjrp,cjrp,rmhmm,fhdmm) S000580
C S000590
C compute the explosive burning critical velocity S000600
C vfcbx=vcrth(tcw,athw,bthw,whhcm,fhacm2,fmhaz) S000610
C S000620
C compute the propellant burning critical velocity S000630
C vfcbp=vcrth(tcr,athr,bthr,rmhcm,fhacm2,fmhaz) S000640
C S000650
C endif S000660
C S000670
C report return S000680
C if(merout.gt.0) write(2,'(6x,a)') 'returning from shpen' S000690
C S000700
C return S000710
C S000720
C end S000730
C S000740
C S000750
C S000760
C***** S000770

```

```

C***** V000010
C      subroutine vmthr(shhcm,facm2,vffs,fmgm,tcsv,athsv,bthsv,cthsv,
+      tcsm,athsm,bthsm,cthsm, vf, fm) V000020
C      V000030
C      adjusts the fragment velocity and mass for shroud penetration V000040
C      V000050
C      common blocks V000060
C      common/cnstu/ fgrrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmft,fi2tc2 V000070
C      common/error/ meror,irerr,iferr,merout V000080
C      report call V000090
C      if(merout.gt.0) write(2,'(8x,a)') 'vmthr called' V000100
C      V000110
C      compute the plug volume in sq cm V000120
C      ha=shhcm*facm2 V000130
C      V000140
C      convert the fragment velocity from ft/s to m/s V000150
C      vfms=ffttm*vffs V000160
C      V000170
C      adjust the fragment velocity in m/s using THOR (zero-obliquity) V000180
C      vf=vfms-tcsv*(ha**athsv)*(fmgm**bthsv)*(vfms**cthsv) V000190
C      if(vf.lt.0.0) vf=0.0 V000200
C      V000210
C      adjust the fragment mass in gm using THOR (zero-obliquity) V000220
C      fm=fmgm-tcsm*(ha**athsm)*(fmgm**bthsm)*(vfms**cths) V000230
C      if(fm.lt.0.0) fm=0.0 V000240
C      V000250
C      convert the hazard fragment velocity from m/s to ft/s V000260
C      vf=fmft*vf V000270
C      V000280
C      report return V000290
C      if(merout.gt.0) write(2,'(8x,a)') 'returning from vmthr' V000300
C      V000310
C      return V000320
C      V000330
C      end V000340
C      V000350
C      V000360
C      V000370
C***** V000380
C***** V000390
C***** V000010
C      subroutine vulnd(vcrd,vfhaz,afrr,atop,coesele,sinele, avdft2) V000020
C      V000030
C      determines the detonation vulnerable area V000040
C      V000050
C      common blocks V000060
C      common/cnstu/ fgrrtlb,fgmtlb,fgrtgm,ffttm,ffttmm,fmft,fi2tc2 V000070
C      common/error/ meror,irerr,iferr,merout V000080
C      report call V000090
C      if(merout.gt.0) write(2,'(6x,a)') 'vulnd called' V000100
C      V000110
C      convert the fragment velocity from ft/s to mm/us V000120
C      vfmmus=ffttmm*vfhaz V000130
C      V000140
C      check the detonation condition V000150
C      if(vfmmus.gt.vcrd) then V000160
C      V000170
C      the fragment can produce detonation V000180
C      V000190
C      compute the maximum obliquity V000200
C      cosobx=vcrd/vfmmus V000210
C      V000220
C      compute the vulnerable area for initiation of detonation V000230
C      call iniar(cosobx,coesele,sinele,afrr,atop, avdft2) V000240
C      V000250
C      endif V000260
C      V000270
C      report return V000280
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from vulnd' V000290
C      V000300
C      return V000310
C      V000320
C      end V000330
C      V000340
C      V000350
C      V000360
C***** V000370

```

```

C***** V000010
C V000020
C      subroutine vulnb(vcrb,cth,dth,vrms,vfhaz,afrn,atop,cosele,sinele, V000030
+          avbft2) V000040
C V000050
C      determines the burn vulnerable area V000060
C V000070
C      common blocks V000080
C      common/cnstu/ fgtrtlb,fgmtlb,fgtrgm,ffttm,ffttmm,fmtft,fi2tc2 V000090
C      common/error/ meror,irerr,iferr,merout V000100
C V000110
C      report call V000120
C      if(merout.gt.0) write(2,'(6x,a)') 'vulnb called' V000130
C V000140
C      convert the fragment velocity from ft/s to m/s V000150
C      vfms=ffttm*vfhaz V000160
C V000170
C      compute the burn critical condition using THOR V000180
C      vright=vcrb*vfms**cth V000190
C      vleft=vfms-vrms V000200
C V000210
C      check the burn condition V000220
C      if(vleft.gt.vright) then V000230
C V000240
C          the fragment can produce burning V000250
C V000260
C          compute the maximum obliquity V000270
C          cosobx=(vright/vleft)**dth V000280
C V000290
C          compute the vulnerable area for initiation of burning V000300
C          call iniar(cosobx,cosele,sinele,afrn,atop, avbft2) V000310
C V000320
C      endif V000330
C V000340
C      report return V000350
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from vulnb' V000360
C V000370
C      return V000380
C V000390
C      end V000400
C V000410
C***** V000420

```

```

C***** I000010
C      subroutine iniar(cosobx,cosele,sinele,afrn,atop, avft2) I000020
C      computes the vulnerable area for initiation I000030
C      I000040
C      common blocks I000050
C      common/error/ meror,irerr,iferr,merout I000060
C      common/acflg/ nweps,macwh,macrm,macsh,mstor I000070
C      common/rkvar/ rk(7) I000080
C      I000090
C      report call I000100
C      if(merout.gt.0) write(2,'(8x,a)') 'iniar called' I000110
C      I000120
C      if(mstor.eq.1) then I000130
C      I000140
C      the storage arrangement is vertical I000150
C      I000160
C      compute the maximum tangency I000170
C      costax=cosobx/cosele I000180
C      I000190
C      if the maximum obliquity exceeds the elevation angle I000200
C      if(costax.lt.1.0) then I000210
C      I000220
C      compute the front vulnerable area I000230
C      avft2=avft2+cosele*sqrt(1.0-costax*costax)*afrn I000240
C      I000250
C      endif I000260
C      I000270
C      if the vertical fragment velocity component is negative I000280
C      if(rkv(4).lt.0.0) then I000290
C      I000300
C      and if the elevation angle exceeds the I000310
C      complement of the maximum obliquity I000320
C      if(sinele.gt.cosobx) then I000330
C      I000340
C      compute the top vulnerable area I000350
C      avft2=avft2+sinele*atop I000360
C      I000370
C      endif I000380
C      I000390
C      endif I000400
C      I000410
C      endif I000420
C      I000430
C      else I000440
C      I000450
C      the storage arrangement is horizontal I000460
C      I000470
C      compute the sine of the maximum obliquity I000480
C      sinobx=sqrt(1.0-cosobx*cosobx) I000490
C      I000500
C      compute the front vulnerable area I000510
C      avft2=avft2+sinobx*afrn I000520
C      I000530
C      if the vertical fragment velocity component is negative I000540
C      if(rkv(4).lt.0.0) then I000550
C      I000560
C      compute the top vulnerable area I000570
C      avft2=avft2+sinobx*atop I000580
C      I000590
C      endif I000600
C      I000610
C      endif I000620
C      I000630
C      report return I000640
C      if(merout.gt.0) write(2,'(8x,a)') 'returning from iniar' I000650
C      I000660
C      return I000670
C      I000680
C      end I000690
C      I000700
C***** I000710

```

```

C***** V000010
C      subroutine vulnm(iframe,vfhaz,cosele,sinele, avmft2) V000020
C      determines mechanical damage vulnerable area V000030
C      common blocks V000040
C      common/error/ meror,irerr,iferr,merout V000050
C      common/fragk/ frkec(900) V000060
C      common/mechd/ wpke V000070
C      common/vulns/ amfrn,amtop,axfrn,axtop,apfrn,aptop V000080
C      common/rkvar/ rk(7) V000090
C      report call V000100
C      if(merout.gt.0) write(2,'(6x,a)') 'vulnm called' V000110
C      compute the kinetic energy V000120
C      frke=frkec(iframe)*vfhaz*vfhaz V000130
C      check the mechanical damage condition V000140
C      if(wpke.lt.frke) then V000150
C      the fragment produces mechanical damage V000160
C      avmft2=amfrn*cosele V000170
C      if(rk(4).lt.0.0) avmft2=avmft2+amtop*sinele V000180
C      else V000190
C      the fragment does not produces mechanical damage V000200
C      avmft2=0.0 V000210
C      endif V000220
C      report return V000230
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from vulnm' V000240
C      return V000250
C      end V000260
C***** V000270
C      subroutine nohit(iframe,aptft2,apvft2,irange,itrpm,itrpt,pnrt,pn0t, V000280
C      + pnrt) V000290
C      accumulates miss probability V000300
C      common blocks V000310
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum V000320
C      common/error/ meror,irerr,iferr,merout V000330
C      common/fragd/ famil(900),vfifs(900),farma(900), V000340
C      + fatn(900),ezldeg(900) V000350
C      local array V000360
C      dimension pnrt(97) V000370
C      report call V000380
C      if(merout.gt.0) write(2,'(6x,a)') 'nohit called' V000390
C      vfarat=-fatn(iframe)*aptft2/apvft2 V000400
C      pnrt=(pnrt*float(itrpm)+expon(vfarat))/float(itrpt) V000410
C      pnrt(irange)=max(pnrt(irange)*pnrt/pn0t,0.0) V000420
C      pn0t=pnrt V000430
C      report return V000440
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from nohit' V000450
C      return V000460
C      end V000470
C***** N000010
C      subroutine nohit(iframe,aptft2,apvft2,irange,itrpm,itrpt,pnrt,pn0t, N000020
C      + pnrt) N000030
C      accumulates miss probability N000040
C      common blocks N000050
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum N000060
C      common/error/ meror,irerr,iferr,merout N000070
C      common/fragd/ famil(900),vfifs(900),farma(900), N000080
C      + fatn(900),ezldeg(900) N000090
C      local array N000100
C      dimension pnrt(97) N000110
C      report call N000120
C      if(merout.gt.0) write(2,'(6x,a)') 'nohit called' N000130
C      vfarat=-fatn(iframe)*aptft2/apvft2 N000140
C      pnrt=(pnrt*float(itrpm)+expon(vfarat))/float(itrpt) N000150
C      pnrt(irange)=max(pnrt(irange)*pnrt/pn0t,0.0) N000160
C      pn0t=pnrt N000170
C      report return N000180
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from nohit' N000190
C      return N000200
C      end N000210
C***** N000220
C      subroutine nohit(iframe,aptft2,apvft2,irange,itrpm,itrpt,pnrt,pn0t, N000230
C      + pnrt) N000240
C      accumulates miss probability N000250
C      common blocks N000260
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum N000270
C      common/error/ meror,irerr,iferr,merout N000280
C      common/fragd/ famil(900),vfifs(900),farma(900), N000290
C      + fatn(900),ezldeg(900) N000300
C      local array N000310
C      dimension pnrt(97) N000320
C      report call N000330
C      if(merout.gt.0) write(2,'(6x,a)') 'nohit called' N000340
C      vfarat=-fatn(iframe)*aptft2/apvft2 N000350
C      pnrt=(pnrt*float(itrpm)+expon(vfarat))/float(itrpt) N000360
C      pnrt(irange)=max(pnrt(irange)*pnrt/pn0t,0.0) N000370
C      pn0t=pnrt N000380
C      report return N000390
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from nohit' N000400
C      return N000410
C      end N000420
C***** N000430

```

```

C***** D000010
C D000020
C      subroutine dnsty(ifrag,apvft2,irange,itrpm,itrpt,fdrt,fd0t,fdt) D000030
C      accumulates fragment density D000040
C      D000050
C      common blocks D000060
C      common/error/ meror,irerr,iferr,merout D000070
C      common/repls/ nreps,fnreps D000080
C      common/fragd/ famil(900),vfifs(900),farma(900), D000090
C      + fatn(900),ezldeg(900) D000100
C      D000110
C      local array D000120
C      dimension fdt(97) D000130
C      D000140
C      report call D000150
C      if(merout.gt.0) write(2,'(6x,a)') 'dnsty called' D000160
C      D000170
C      fdrt=(fdrt*float(itrpm)+fatn(ifrag)/apvft2)/float(itrpt) D000180
C      fdlt=fdrt/fnreps D000190
C      fdt(irange)=fdt(irange)+fdlt-fd0t D000200
C      fd0t=fdlt D000210
C      D000220
C      D000230
C      report return D000240
C      if(merout.gt.0) write(2,'(6x,a)') 'returning from dnsty' D000250
C      D000260
C      return D000270
C      D000280
C      end D000290
C      D000300
C***** D000310

```

```

C***** R000010
C subroutine ricoc(slcnst,vfrag,aeldeg,selrad,vrico,ricrad) R000020
C determines ricochet conditions R000030
C common block R000040
C common/error/ meror,irerr,iferr,merout R000050
C report call R000060
C if(merout.gt.0) write(2,'(2x,a)') 'ricoc called' R000070
C R000080
C R000090
C R000100
C R000110
C R000120
C R000130
C R000140
C R000150
C R000160
C R000170
C R000180
C R000190
C R000200
C R000210
C R000220
C R000230
C R000240
C R000250
C R000260
C R000270
C R000280
C R000290
C R000300
C R000310
C R000320
C R000330
C R000340
C R000350
C R000360
C R000370
C R000380
C R000390
C R000400
C R000410
C R000420
C R000430
C R000440
C R000450
C R000460
C R000470
C R000480
C R000490
C R000500
C R000510
C R000520
C R000530
C R000540
C R000550
C R000560
C R000570
C R000580
C R000590
C R000600
C R000610
C R000620
C R000630
C R000640

```

```

C***** R000010
C R000020
C      subroutine reini(irico,vrico,ricrad,wspeed,wdrdeg,vfrag,selrad, R000030
+      seldeg,aelrad,aeldeg,deldst,mfrht1) R000040
C R000050
C      reinitializes the velocity integration variables after ricochet R000060
C R000070
C      common blocks R000080
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum R000090
C      common/error/ meror,irerr,iferr,merout R000100
C      common/rkvar/ rk(7) R000110
C R000120
C      report call R000130
C      if(merout.gt.0) write(2,'(2x,a)') 'reini called' R000140
C R000150
C      increment the ricochet count R000160
C      irico=irico+1 R000170
C R000180
C      assign the ricochet values to the fragment variables R000190
C      vfrag=vrico R000200
C      selrad=ricrad R000210
C      seldeg=raddeg*selrad R000220
C      aelrad=abs(selrad) R000230
C      aeldeg=abs(seldeg) R000240
C R000250
C      assign the fragment variable values to the Runge-Kutta variables R000260
C      rk(4)=vfrag*sin(selrad) R000270
C      if((wdrdeg.eq.0.0).or.(wspeed.eq.0.0)) then R000280
C          rk(2)=vfrag*cos(selrad) R000290
C          rk(6)=0.0 R000300
C      else R000310
C          angrad=atan(rk(2)/rk(6)) R000320
C          rk(2)=vfrag*cos(selrad)*sin(angrad) R000330
C          rk(6)=vfrag*cos(selrad)*cos(angrad) R000340
C      endif R000350
C R000360
C      call dstep(rk(5),seldeg,selrad,deldst) R000370
C R000380
C      mfrht1=0 R000390
C R000400
C      report return R000410
C      if(merout.gt.0) write(2,'(2x,a)') 'returning from reini' R000420
C R000430
C      return R000440
C R000450
C      end R000460
C R000470
C***** R000480

```

```

c***** H000010
c
c      subroutine hpacc H000020
c H000030
c      accumulates hit probabilities H000040
c H000050
c H000060
c      common blocks H000070
c      common/error/ meror,irerr,iferr,merout H000080
c      common/repls/ nreps,fnreps H000090
c      common/rangs/ nrsegs,nrsegs H000100
c      common/phitt/ ph(97),phd(97),phb(97),phm(97) H000110
c      common/pmiss/ pn(97),pnd(97),pnb(97),pnm(97) H000120
c H000130
c      report call H000140
c      if(merout.gt.0) write(2,'(2x,a)') 'hpacc called' H000150
c H000160
c      do 100 irange=1,nrsegs H000170
c          ph(irange)=ph(irange)+(1.0-pn(irange))/fnreps H000180
c          phd(irange)=phd(irange)+(1.0-pnd(irange))/fnreps H000190
c          phb(irange)=phb(irange)+(1.0-pnb(irange))/fnreps H000200
c          phm(irange)=phm(irange)+(1.0-pnm(irange))/fnreps H000210
100 continue H000220
c H000230
c      report return H000240
c      if(merout.gt.0) write(2,'(2x,a)') 'returning from hpacc' H000250
c H000260
c      return H000270
c H000280
c      end H000290
c H000300
c***** H000310

c***** O000010
c O000020
c      subroutine outpt O000030
c O000040
c      writes output to appropriate files O000050
c O000060
c      common blocks O000070
c      common/phitt/ ph(97),phd(97),phb(97),phm(97) O000080
c      common/denst/ fd(97),fdd(97),fdb(97),fdm(97) O000090
c O000100
c      output the hit probabilities O000110
c      call outfi('fp.php',ph,phd,phb,phm) O000120
c O000130
c      output the fragment densities O000140
c      call outfi('fp.pfd',fd,fdd,fdb,fdm) O000150
c O000160
c      return O000170
c O000180
c      end O000190
c O000200
c***** O000210

```

```

C***** 0000010
C      0000020
      subroutine outf1(fname,v,vd,vb,vm)      0000030
C      0000040
      outputs hit probabilities or fragment densities 0000050
C      0000060
      common blocks      0000070
      common/repls/ nreps,fnreps      0000080
      common/donor/ nunits,htstft,htbaft 0000090
      common/rangs/ nrsegs,nrsegg      0000100
      common/range/ rngmax,rngseg,rngsgh,rngscl,rngsix,rngmin 0000110
      common/arang/ rangem(97),ahfrn(97),ahtop(97) 0000120
      common/names/ dname,aname      0000130
      common/acptr/ htacft,wdacft,dpacft,htacfm 0000140
C      0000150
      local arrays      0000160
      dimension v(97),vd(97),vb(97),vm(97) 0000170
C      0000180
      type declarations      0000190
      character fname*6,dname*40,aname*40 0000200
C      0000210
      open(3,file=fname,status='unknown') 0000220
C      0000230
      write(3,'(3a)') char(39),dname,char(39) 0000240
      write(3,'(2i3,1p2e16.8)') nreps,nunits,htstft,htbaft 0000250
      write(3,'(3a)') char(39),aname,char(39) 0000260
      write(3,'(1p3e16.8)') htacft,wdacft,dpacft 0000270
      write(3,'(1p2e16.8)') rngmin,rngmax 0000280
      do 100 irange=1,nrsegs      0000290
        write(3,'(1p5e16.8)') rangem(irange),vd(irange),
+          vb(irange),vm(irange),v(irange) 0000300
100 continue      0000310
C      0000320
      close(3)      0000330
C      0000340
      return      0000350
C      0000360
      end      0000370
C      0000380
C      0000390
C***** 0000400

```

```

C***** R000010
C      function rnaval(valmin,valmax,rndnxt) R000020
C      generates a random value between two specified values R000030
C      type declaration R000040
C      double precision rndnxt R000050
C      R000060
C      rnaval=valmin+(valmax-valmin)*rdom(rndnxt) R000070
C      return R000080
C      end R000090
C      R000100
C      R000110
C      R000120
C      R000130
C      R000140
C      R000150
C***** R000160

C***** R000010
C      function rdom(uix) R000020
C      generates random numbers R000030
C      type declarations R000040
C      double precision ua,up,uix,uiy,ub15,ub16, R000050
C      + uxhi,uxalo,ulftlo,ufhi,uk R000060
C      data statement R000070
C      data ua/16807.0d0/,ub15/32768.0d0/, R000080
C      + ub16/65536.0d0/,up/2147483647.0d0/ R000090
C      R000100
C      if(uix.eq.0) uix=uiy R000110
100 uxhi=uix/ub16 R000120
    uxhi=uxhi-dmod(uxhi,1.0d0) R000130
    uxalo=(uix-uxhi*ub16)*ua R000140
    ulftlo=uxalo/ub16 R000150
    ulftlo=ulftlo-dmod(ulftlo,1.0d0) R000160
    ufhi=uxhi*ua+ulftlo R000170
    uk=ufhi/ub15 R000180
    uk=uk-dmod(uk,1.0d0) R000190
    uix=((uxalo-ulftlo*ub16)-up)+(ufhi-uk*ub15)*ub16+uk R000200
    if(uix.lt.0.0) uix=uix+up R000210
    uiy=uix R000220
    rdom=int((uix*4.656612875d-10)*1.0d06)/1.0d06 R000230
    if(rdom.eq.0.0) go to 100 R000240
C      return R000250
C      end R000260
C      R000270
C      R000280
C      R000290
C      R000300
C      R000310
C      R000320
C      R000330
C***** R000340

```

```

C***** V000010
C                                             V000020
C      function vcrjr(abjr,cjr,hmm,fdmm)      V000030
C                                             V000040
C      computes the Jacobs-Roslund critical velocity V000050
C                                             V000060
C      common block                          V000070
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum V000080
C                                             V000090
C      if(fdmm.eq.0.0) then                  V000100
C          vcrjr=bignum                      V000110
C      else                                  V000120
C          vcrjr=abjr*(1.0+cjr*hmm/fdmm)/sqrt(fdmm) V000130
C      endif                                V000140
C                                             V000150
C      return                               V000160
C                                             V000170
C      end                                   V000180
C                                             V000190
C***** V000200

C***** V000010
C                                             V000020
C      function vcrth(tc,ath,bth,hcm,facm2,fmgm) V000030
C                                             V000040
C      computes the THOR critical velocity      V000050
C                                             V000060
C      common block                          V000070
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum V000080
C                                             V000090
C      if(fmgm.eq.0.0) then                  V000100
C          vcrth=bignum                      V000110
C      else                                  V000120
C          vcrth=tc*((hcm*facm2)**ath)*(fmgm**bth) V000130
C      endif                                V000140
C                                             V000150
C      return                               V000160
C                                             V000170
C      end                                   V000180
C                                             V000190
C***** V000200

C***** E000010
C                                             E000020
C      function expon(arg)                   E000030
C                                             E000040
C      computes the exponential function        E000050
C      limited for large negative arguments     E000060
C                                             E000070
C      common block                          E000080
C      common/cnstm/ qpi,hpi,pi,tpi,raddeg,degrad,smlrad,smlnum,bignum E000090
C                                             E000100
C      expon=max(exp(arg),smlnum)             E000110
C                                             E000120
C      return                               E000130
C                                             E000140
C      end                                   E000150
C                                             E000160
C***** E000170

```

APPENDIX C:
LETHALITY AND VULNERABILITY DATA

INTENTIONALLY LEFT BLANK.

The data required for M107 and TOW-2A lethality and vulnerability descriptions is summarized in Tables C-1 through C-5.

Table C-1. Weapon Dimensions and Materials

Component	M107	TOW-2A
Weapon:		
Outside Diameter	6.10 in	6.187 in
Length	23.90 in	60.000 in*
Warhead:		
Outside Diameter	6.10 in	5.850 in
Length	23.90 in	25.570 in
Casing Material	steel	aluminum
Casing Thickness	0.63 in	0.050 in
Explosive Type	Comp. B	LX-14
Charge Diameter	4.84 in	5.750 in
Charge Length	20.30 in	9.574 in
Rocket Motor:		
Outside Diameter	N/A	5.837 in
Length	N/A	18.700 in
Casing Material	N/A	steel
Casing Thickness	N/A	0.055 in
Propellant Type	N/A	GCV
Charge Diameter	N/A	5.782 in
Charge Length	N/A	6.320 in
Container:		
Material	N/A	steel
Outside Diameter	N/A	10.000 in
Thickness	N/A	0.040 in
Pallet:		
Orientation	vertical	horizontal
Arrangement (H x W x D)	1 x 4 x 2	3 x 1 x 4
Weapon Spacing	1.20 in	1.000 in*
Elevation	8.10 in	8.100 in*
Height (with pallet)	32.00 in	40.220 in
Width	28.00 in	60.000 in
Depth	13.40 in	43.160 in

* estimated value

Table C-2. Energetic Material Performance Parameters

Material	Density (g/cm ²)	Detonation Pressure (kbar)	Gurney Constant (m/s)	Product Gases (mole/g)	Molecular Weight	Heat of Detonation (cal/g)
LX-14	1.83	370.0	2,948.0	Not Required	Not Required	Not Required
GCV Propellant	1.8	Unavailable	Unavailable	0.042156	24.635	890.0

Table C-3. Jacobs-Roslund Constants

Material	a_{jr} (mm ^{3/2} /μs)	b_{jr}	c_{jr}
Composition B	3.065	0.0	1.70
LX-14*	3.6	0.0	1.59
GCV Propellant*	4.8	0.0	1.3

* estimated values

Table C-4. THOR Velocity Equation Constants

Material	a_v	b_v	c_v	d_v	e_v
mild steel	3.6901	0.889	-0.945	0.019	1.262
aluminum	3.9356	1.029	-1.072	-0.139	1.251

Table C-5. THOR Mass Equation Constants

Material	a_m	b_m	c_m	d_m
mild steel	-2.478	0.138	0.835	0.761

APPENDIX D:
DONOR FRAGMENTATION GENERATOR LISTING

INTENTIONALLY LEFT BLANK.

```

C***** D000010
C D000020
C   program dogen D000030
C D000040
C           DONOR FRAGMENTATION DATA GENERATOR D000050
C           (Horizontal Storage) D000060
C D000070
C   common blocks D000080
C   common/frgwh/ iazwh(2500),fmwhgm(2500) D000090
C   common/frgrm/ iazrm(2500),fmrmgm(2500) D000100
C D000110
C   type declaration D000120
C   double precision rndnxt D000130
C D000140
C   define constants D000150
C   call const D000160
C D000170
C   read and compute warhead parameters D000180
C   call elein('wh.inp',nfrwh,fmavwh,vfwhfs,plzwh,fmnwh) D000190
C D000200
C   read and compute rocket motor parameters D000210
C   call elein('rm.inp',nfrmr,fmavrm,vfrmfs,plzrm,fmnr) D000220
C D000230
C   input the randomization seed D000240
C   call rndin(rndnxt) D000250
C D000260
C   generate the warhead fragment distriubtion D000270
C   call frdst(rndnxt,nfrwh,fmavwh,fmnwh,nazwh,lfrwh,iazwh,fmwhgm, D000280
C   + iazwhx) D000290
C D000300
C   generate the rocket motor fragment distriubtion D000310
C   call frdst(rndnxt,nfrmr,fmavrm,fmnr,nazrm,lfrmr,iazrm,fmrmgm, D000320
C   + iazrmx) D000330
C D000340
C   generate fragmentation data file D000350
C   call mkfil(plzwh,nazwh,lfrwh,nfrwh,vfwhfs,fmnwh,iazwhx, D000360
C   + plzrm,nazrm,lfrmr,nfrmr,vfrmfs,fmnr,iazrmx) D000370
C D000380
C   stop ' ' D000390
C D000400
C   end D000410
C D000420
C***** D000430

C***** C000010
C C000020
C   subroutine const C000030
C C000040
C   defines constants C000050
C C000060
C   common block C000070
C   common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg C000080
C C000090
C   pi=4.0*atan(1.0) C000100
C   twth=2.0/3.0 C000110
C   dazdeg=10.0 C000120
C   idzdeg=int(dazdeg) C000130
C   azxdeg=110.0 C000140
C C000150
C   return C000160
C C000170
C   end C000180
C C000190
C***** C000200

```

```

C***** E000010
C      subroutine elein(infile,nfrag,frmav,vffs,plzdeg,frmmin) E000020
C      reads and computes cylindrical element parameters E000030
C      common block E000040
C      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg E000050
C      type declaration E000060
C      character*6 infile E000070
C      open data file E000080
C      open(3,file=infile,status='old') E000090
C      read casing data E000100
C      read(3,*) rhca,romm,drmm,htmm,plzdeg,frmmin E000110
C      read fill data E000120
C      read(3,*) rhfl,pdfl,vgfl E000130
C      if(pdfl.eq.0.0) read(3,*) fln,flm,flq E000140
C      close data file E000150
C      close(3) E000160
C      convert dimensions to centimeters E000170
C      rocm=0.1*romm E000180
C      drcm=0.1*drmm E000190
C      htcn=0.1*htmm E000200
C      compute the casing thickness and inside diameter in inches E000210
C      drin=drmm/25.4 E000220
C      diin=(romm-drmm)/25.4 E000230
C      compute the inside radius in centimeters E000240
C      ricm=rocm-drcm E000250
C      ricm2=ricm*ricm E000260
C      compute the casing mass in grams E000270
C      camgm=rhca*pi*(rocm*rocm-ricm2)*htcm E000280
C      write(*,'(a,lpe12.5,a)') ' casing mass =',camgm,' g' E000290
C      compute the fill mass in grams E000300
C      flmgm=rhfl*pi*ricm2*htcm E000310
C      write(*,'(a,lpe12.5,a)') ' fill mass =',flmgm,' g' E000320
C      compute the ratio of casing to fill mass E000330
C      camfl=camgm/flmgm E000340
C      compute the detonation pressure if necessary E000350
C      if(pdfl.eq.0.0) pdfl=15.58*rhfl*rhfl*fln*sqrt(flm*flq) E000360
C      write(*,'(a,lpe12.5,a)') ' detonation pressure =',pdfl,' kbar' E000370
C      compute the Gurney constant if necessary E000380
C      if(vgfl.eq.0.0) vgfl=233.0*sqrt(pdfl)/rhfl**0.6 E000390
C      write(*,'(a,lpe12.5,a)') ' Gurney constant =',vgfl,' m/s' E000400
C      compute the fragment velocity E000410
C      vfms=vgfl/sqrt(camfl+0.5) E000420
C      vffs=3.2808399*vfms E000430
C      write(*,'(a,lpe12.5,a)') ' fragment velocity =',vfms,' m/s' E000440
C      compute the average fragment mass in grams E000450
C      frmav= E000460
C      + 6.762e2*sqrt(1.0+0.5*camfl)*(drin*((diin+drin)**1.5)/diin)/pdfl E000470
C      write(*,'(a,lpe12.5,a)') ' average fragment mass =',frmav,' g' E000480
C      compute the number of fragments (adjust sector size if needed) E000490
C      100 nfrag=int(camgm*azxdeg/(360.0*frmav)) E000500
C      if(nfrag.gt.2500) then E000510
C      azxdeg=2500.0*azxdeg/float(nfrag) E000520
C      go to 100 E000530
C      endif E000540
C      write(*,'(a,i5)') ' number of fragments =',nfrag E000550
C      write(*,'(a,lpe12.5,a/)') ' azimuthal sector size =',azxdeg,' deg' E000560
C      return E000570
C      end E000580
C***** E000590
C***** E000600
C***** E000610
C***** E000620
C***** E000630
C***** E000640
C***** E000650
C***** E000660
C***** E000670
C***** E000680
C***** E000690
C***** E000700
C***** E000710
C***** E000720
C***** E000730
C***** E000740
C***** E000750
C***** E000760
C***** E000770
C***** E000780
C***** E000790
C***** E000800
C***** E000810

```

```

C***** F000010
C      F000020
C      subroutine frdst(rndnxt,nfrag,frmav,frmmin,naz,lfrag,iaz,frmgm, F000030
+          iazmax) F000040
C      F000050
C      generates a fragment distriubtion F000060
C      F000070
C      common block F000080
C      common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg F000090
C      F000100
C      type declaration F000110
C      double precision rndnxt F000120
C      F000130
C      local arrays F000140
C      dimension iaz(2500),frmgm(2500) F000150
C      F000160
C      generate fragment masses with Mott distribution F000170
C      and azimuthal angles with even distribution F000180
C      hfrmav=0.5*frmav F000190
C      iazmax=0 F000200
C      lfrag=0 F000210
C      do 100 ifrag=1,nfrag F000220
C          rnddum=rndom(rndnxt) F000230
C          frmgm(ifrag)=-hfrmav*alog(rnddum*rnddum) F000240
C          iaz(ifrag)=idzdeg*int(azxdeg*rndom(rndnxt)/dazdeg)+idzdeg F000250
C          iazmax=max(iazmax,iaz(ifrag)) F000260
C          if(frmgm(ifrag).gt.frmmin) lfrag=lfrag+1 F000270
100 continue F000280
C      naz=iazmax/idzdeg F000290
C      F000300
C      return F000310
C      F000320
C      end F000330
C      F000340
C***** F000350

```

```

C***** M000010
C M000020
  subroutine mkfil(plzwh,nazwh,lfrwh,nfrwh,vfwhfs,fmnwh,iazwhx, M000030
+   plzrm,nazrm,lfrrm,nfrmm,vfrmmfs,fmnrm,iazrmx) M000040
C M000050
C   generates fragmentation data file M000060
C M000070
C   common blocks M000080
common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg M000090
common/frgwh/ iazwh(2500),fmwhgm(2500) M000100
common/frgrm/ iazrm(2500),fmrmmgm(2500) M000110
C M000120
C   output fragmentation data with M000130
C   azimuthal and polar angles interchanged M000140
naz=max(nazwh,nazrm) M000150
iazmax=max(iazwhx,iazrmx) M000160
lfrag=lfrwh+lfrmm M000170
open(3,file='fr.out',status='unknown') M000180
write(3,'(3x,1p2e15.7)') dazdeg,max(plzwh,plzrm) M000190
write(3,'(3x,3i5)') naz,lfrag,1 M000200
write(4,1p15.7) 0.04 M000210
write(4,1p4e15.7) 3.6901,0.889,-0.945,0.019 M000220
write(4,1p4e15.7) -2.478,0.138,0.835,0.761 M000230
write(3,'(3x,1p4e15.7)') (0.05,iz=1,naz) M000240
do 100 iwrite=idzdeg,iazmax,idzdeg M000250
  call adfrg(iwrite,nfrwh,fmwhgm,iazwh,fmnwh,vfwhfs) M000260
  call adfrg(iwrite,nfrmm,fmrmmgm,iazrm,fmnrm,vfrmmfs) M000270
100 continue M000280
close(3) M000290
C M000300
C   return M000310
C M000320
C   end M000330
C M000340
C***** M000350

C***** A000010
C A000020
  subroutine adfrg(iwrite,nfrag,frmgm,iaz,fmn,vffs) A000030
C A000040
C   common blocks A000050
common/cnstm/ pi,twth,dazdeg,idzdeg,azxdeg A000060
C A000070
C   local arrays A000080
dimension iaz(2500),frmgm(2500) A000090
C A000100
C   do 100 ifrag=1,nfrag A000110
  if(frmgm(ifrag).gt.fmn) then A000120
    if(iaz(ifrag).eq.iwrite) then A000130
      acm2=0.5199*frmgm(ifrag)**twth A000140
      ain2=0.1550003*acm2 A000150
      frmlb=2.2046226e-3*frmgm(ifrag) A000160
      frmgr=7000.0*frmlb A000170
      aminlb=ain2/frmlb A000180
      write(3,'(i3,1p4e15.7)') iaz(ifrag),frmgr,aminlb,vffs,1.5 A000190
    endif A000200
  endif A000210
100 continue A000220
C A000230
C   return A000240
C A000250
C   end A000260
C A000270
C A000280
C***** A000290

```

```

C***** R000010
C R000020
C      subroutine rndin(rndnxt) R000030
C R000040
C      obtains randomization seed input R000050
C R000060
C      type declarations R000070
C      double precision rndinp, rndnxt R000080
C R000090
C 100 write(*, '(a\)' ) ' Enter Monte Carlo Seed (1 to 2147483646): ' R000100
C      read(*, *, err=100) rndinp R000110
C      if(rndinp.lt.1.0.or.rndinp.gt.2147483646.0) go to 100 R000120
C R000130
C      rnddum=rndom(rndinp) R000140
C      rndnxt=0.0d0 R000150
C R000160
C      return R000170
C R000180
C      end R000190
C R000200
C***** R000210

C***** R000010
C R000020
C      function rndom(uix) R000030
C R000040
C      generates random numbers R000050
C R000060
C      type declarations R000070
C      double precision ua, up, uix, uiy, ub15, ub16, uxhi, uxalo, ulftlo, ufhi, R000080
C      + uk R000090
C R000100
C      data statement R000110
C      data ua/16807.0d0/, ub15/32768.0d0/, ub16/65536.0d0/, R000120
C      + up/2147483647.0d0/ R000130
C R000140
C      if(uix.eq.0) uix=uiy R000150
C 100 uxhi=uix/ub16 R000160
C      uxhi=uxhi-dmod(uxhi,1.0d0) R000170
C      uxalo=(uix-uxhi*ub16)*ua R000180
C      ulftlo=uxalo/ub16 R000190
C      ulftlo=ulftlo-dmod(ulftlo,1.0d0) R000200
C      ufhi=uxhi*ua+ulftlo R000210
C      uk=ufhi/ub15 R000220
C      uk=uk-dmod(uk,1.0d0) R000230
C      uix=((uxalo-ulftlo*ub16)-up)+(ufhi-uk*ub15)*ub16)+uk R000240
C      if(uix.lt.0.0) uix=uix+up R000250
C      uiy=uix R000260
C      rndom=int((uix*4.656612875d-10)*1.0d06)/1.0d06 R000270
C      if(rndom.eq.0.0) go to 100 R000280
C R000290
C      return R000300
C R000310
C      end R000320
C R000330
C***** R000340

```

INTENTIONALLY LEFT BLANK.

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
2	ADMINISTRATOR DEFENSE TECHNICAL INFO CTR ATTN DTIC DDA CAMERON STATION ALEXANDRIA VA 22304-6145

1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TA 2800 POWDER MILL RD ADELPHI MD 20783-1145
---	---

3	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TL 2800 POWDER MILL RD ADELPHI MD 20783-1145
---	---

1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TP 2800 POWDER MILL RD ADELPHI MD 20783-1145
---	---

ABERDEEN PROVING GROUND

5	DIR USARL ATTN AMSRL OP AP L (305)
---	---------------------------------------

<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>	<u>NO. OF COPIES</u>	<u>ORGANIZATION</u>
1	DIRECTOR USA TECH CTR FOR EXPLOSIVE SAFETY ATTN SMCAC ESL C DOYLE SAVANNA IL 61074-9639	1	OFFICER IN CHARGE WHITE OAK LABORATORY NSWC ATTN R15 M SWISDAK 10901 NEW HAMPSHIRE AVE SILVER SPRING MD 20902-5000
1	DIRECTOR USA TECH CTR FOR EXPLOSIVE SAFETY ATTN SMCAC DO N MCCULLOUGH SAVANNA IL 61074-9639	2	NAVAL FACILITIES ENGINEERING SERV CTR ATTN CODE LS1 J TANCRETO R MURTHE PORT HUENEME CA 93043
3	US ARMY WATERWAYS EXP STATION ATTN CEWES SE K DAVIS M FORD C JOACHIM 3909 HALLS FERRY RD VICKSBURG MS 39180-6199	1	DIRECTOR EXPLOSIVES HAZARD REDUCTION DIR ATTN ASC YOCO EHR J JENUS EGLIN AFB FL 32542
2	CHAIRMAN DOD EXPLOSIVE SAFETY BOARD ATTN DDESB KT J WARD C CANADA 2461 EISENHOWER AVE ALEXANDRIA VA 22331-0600	2	US ARMY CORPS OF ENGINEERS ATTN CEHND ED CS R WRIGHT R HASSE PO BOX 1600 HUNTSVILLE AL 35807-4301
4	DEFENSE AMMUNITION LOGISTICS ACTIVITY ATTN AMCPM AL R ROSSI R HO E GOON B WILLIAMSON PICATINNY ARSENAL NJ 07806-5000		<u>ABERDEEN PROVING GROUND, MD</u>
1	COMMANDER ATTN SMCAR AEE W P LU US ARMY ARDEC PICATINNY ARSENAL NJ 07806-5000	13	DIR, USARL ATTN: AMSRL-WT-T, W. MORRISON AMSRL-WT-TB, P. BAKER R. FREY O. LYMAN J. WATSON F. GREGORY V. BOYLE W. HILLSTROM E. MCDUGAL W. LAWRENCE K. BENJAMIN T. DORSEY AMSRL-SL-BV, J. PLOSONKA
1	COMMANDER ATTN SMCAR AEM B WILLIAMSON US ARMY ARDEC PICATINNY ARSENAL NJ 07806-5000	2	DIR, USA AMSAA ATTN: AMXSY-GC, E. CHRISTMAN A. WONG
1	COMMANDER ATTN SMCAR ASA D MILLER US ARMY ARDEC PICATINNY ARSENAL NJ 07806-5000		

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-TR-949 Date of Report January 1996
2. Date Report Received _____
3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT
ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD
ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)

DEPARTMENT OF THE ARMY

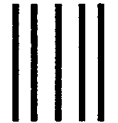
OFFICIAL BUSINESS

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 0001,APG,MD

POSTAGE WILL BE PAID BY ADDRESSEE

**DIRECTOR
U.S. ARMY RESEARCH LABORATORY
ATTN: AMSRL-WT-TB
ABERDEEN PROVING GROUND, MD 21005-5066**



**NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES**

