

DTIC COPY

2

TECHNICAL REPORT BRL-TR-3156

BRL

AD-A228 136

PROGRAMMER/ANALYST GUIDE FOR THE
ARMY UNIT RESILIENCY ANALYSIS (AURA)
COMPUTER SIMULATION MODEL,
VOLUME 1: AURA METHODOLOGY

ROBERT M. SHEROKE
JOHN M. ABELL
STEPHANIE S. JUARASCIO
J. TERRENCE KLOPCIC
LISA K. ROACH

OCTOBER 1990

DTIC
ELECTE
NOV 02 1990
S B D

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

U.S. ARMY LABORATORY COMMAND

BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

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.

UNCLASSIFIED

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 October 1990	3. REPORT TYPE AND DATES COVERED Final		
4. TITLE AND SUBTITLE Programmer/Analyst Guide to the Army Unit Resiliency Analysis (AURA) Computer Simulation Model, Volume 1 : AURA Methodology.			5. FUNDING NUMBERS 1L162618AH80 DA31 6061	
6. AUTHOR(S) Robert M. Sheroke, John M. Abell, Stephanie S. Juarascio, J. Terrence Klopke, Lisa K. Roach			8. PERFORMING ORGANIZATION REPORT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)				
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Ballistic Research Laboratory ATTN: SLCBR-DD-T Aberdeen Proving Ground, MD 21005-5066			10. SPONSORING / MONITORING AGENCY REPORT NUMBER BRL-TR-3156	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This, the first of a three volume report, presents the programmer/analyst with a comprehensive understanding of the methodologies which embody the Army Unit Resiliency Analysis (AURA) model. The approach taken was to progress from a general overview of the AURA model to a detailed description of the derivation, capability and primary algorithms for each AURA methodology. Throughout the report, a simple, hypothetical combat support unit is used as a 'working' example to describe the role of each methodology in the overall combat simulation process. Volume 2 of this report contains a detailed description of the organization and conventions of the FORTRAN source code that embodies the AURA methodology. Volume 3 of this report will contain an in depth description of the conduct of an AURA analysis, from data preparation through output analysis, as presented from an analyst's perspective.				
14. SUBJECT TERMS AURA Combat model Optimization Unit Analysis Effectiveness Lethality Chemical Model Nuclear model			15. NUMBER OF PAGES 170	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED			16. PRICE CODE	
			20. LIMITATION OF ABSTRACT SAR	
18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED			

INTENTIONALLY LEFT BLANK.

CONTENTS

I. Introduction	1
1. Background	1
2. Scope	3
II. Technical Background	3
1. AURA Terminology	3
2. AURA Functional Structures	7
3. Statistical Modes used in AURA	9
a. Deterministic Mode	9
b. Stochastic Mode	10
c. Comparison of AURA's Statistical Modes	10
4. AURA's Coordinate Systems	11
a. The Unit Coordinate System	11
b. The Incoming Fire (Range-Deflection) System	11
c. The Wind Direction (Downwind-Crosswind) System	12
5. General Information	12
a. Operation of the AURA Model	12
b. Event Types	14
c. AURA Inputs	18
d. AURA Outputs	18
III. Scope of AURA Methodology	20
1. A Hypothetical Case Study	20

By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

2. The Hypothetical Case - An Ammunition Supply Point.....	20
a. Unit Discussion	20
b. Applications of the AURA Model.....	24
IV. Reconstitution Methodologies.....	26
1. The AURA Asset Allocation Algorithm.....	26
a. Mathematical Description of the AURA Asset Allocation Algo- rithm.....	32
b. Asset Allocation Algorithm Decision Rules.....	35
c. Subroutines of the Asset Allocation Algorithm	36
(1) The Commander Algorithm.....	37
(2) Optimization of Compound Links.....	38
(3) Optimization of Orlinks.....	38
(4) Optimization of Subchains.....	39
(5) Optimization of Crews.....	40
(6) Optimization of Links.....	40
(7) Calculation of Compound Link Effectiveness.....	42
(8) Calculation of Orlink Effectiveness	42
(9) Calculation of Subchain Effectiveness	43
(10) Calculation of Crew Effectiveness	43
(11) Calculation of Link Effectiveness	44
d. Application of the Asset Allocation Algorithm to the Hypothet- ical Case.....	46
2. The AURA Repair/Return Methodology.....	52
a. Repairs.....	52

b. Ordering and Completion of Repairs	53
c. Repair Mnemonics.....	54
d. Other Repair Related Mnemonics	54
(1) Failures.....	54
(2) Expendables.....	55
(3) General Repair.....	55
(4) Maximum Number.....	55
(5) No Repair	56
e. The Repair Subroutines	56
(1) Determination of Reliability-Type Failures.....	56
(2) Verification of Repair/Failure Inputs.....	57
(3) Calculation of Reliability Type Failures	58
(4) Main PREFL Algorithm	58
(a) Calculation of Prefailed Assets.....	60
(5) Removal of Jobs from Repair Shop	60
(6) Initiation of New Repairs and Calculation of Repair Effectiveness.....	61
(7) Ordering of Repairs	61
(8) Determination of Further Damage to Ongoing Repairs.....	63
(9) Return Repaired Assets to Survivor Pool.....	63
(10) Determination of Repairable Assets and Repair Prior- ity	64
f. Application of Repair to the Hypothetical Case	64

3. The Fatigue/Sleep Deprivation Model.....	65
a. Overview of the Fatigue Model	65
(1) SLUNIT Expenditure.....	65
(2) Job Degradation	68
b. The Fatigue Subroutines.....	69
(1) SLUNIT Build-Up.....	69
(2) Designating Potential Sleepers.....	69
(3) Effects of SLUNITs.....	71
4. Methodology for Degradation Due to Heat Stress.....	71
V. Modeling of Lethality Events and Effects	72
1. Targeting Methodology	72
a. Overview of Targeting and Munition Delivery Methodol- ogy	72
b. Targeting Mnemonics.....	73
(1) Weapons	73
(2) Agent.....	73
(3) Yield	73
(4) Acquisition Probability	73
(5) Target Location Error.....	74
(6) Reliability of Weapon	74
(7) Delivery Errors	75
(8) Miss Distance Option	77
(9) Round and Volley Inputs.....	77

(10) Incoming Fire Direction	78
(11) Wind Direction	79
c. Targeting Methodology Algorithms.....	79
d. Reading and Storing Targeting Information	80
(1) Round Inputs and Calculation of Maximum Range	80
(2) Volley Inputs and Calculation of Maximum Range.....	80
(3) Target Location Error Input and Conversion of CEP TLE to Sigmas	81
(4) Delivery Error Input and Conversion of CEP Errors to Sigmas	82
(5) Probability of Acquisition Input	82
(6) Reliability of Weapon Input	82
(7) Miss Distance Input	82
e. Sampling Before Each Replication	82
f. Coordinate Transformation Subroutine	84
g. Event Processing.....	86
(1) Processing of Other Event Types that Change with Time	87
(2) Processing Lethality Events.....	87
(3) Calculating Actual Ground Zero.....	89
h. Application of Targeting Methodology to Hypothetical Case	92
(1) Single Incoming Rounds.....	92
(2) Cluster Munitions	92
(3) Rectangular Sheaf.....	93

(4) Rolling Barrage.....	94
2. Conventional Lethality Modeling	95
a. Overview of the Conventional Lethality Model	95
(1) Conventional Kill Criteria	96
(2) Application to Hypothetical Case	96
(3) Target Posture.....	98
(4) Lethality Data File.....	98
(a) Application of Lethality Data File to Hypothetical Case	98
b. Triggering of Conventional Scenario	99
c. Modeling Kills Deterministically Versus Stochastically	99
(1) Deterministic Mode.....	100
(2) Stochastic Mode.....	100
d. Overview of the Conventional Lethality Methodology	100
(1) Weapons and Unit Deployment Information Needed by the Lethality Model	100
(2) Conventional Lethality Subroutines.....	101
e. Conventional Lethality Subroutines and Their Functions.....	103
(1) Calculation of Conventional Losses/Damage	104
(2) Calculation of Probability of Kill	105
(3) The Carleton Von Neumann Algorithm.....	105
(4) The Cookie-Cutter Algorithm.....	108
f. Application of Hypothetical Case to the Conventional Model	109

3. Chemical Lethality Model	112
a. Triggering a Toxic Scenario	112
b. Overview of the Chemical Lethality Model	112
(1) Inputs	112
(a) Weapons	112
(b) Unit Deployment.....	113
(c) Toxic Kill Criteria (T.K.C.) Code	113
(d) Chemical Dispersion File (unit #4).....	113
(e) Degradation Inputs for Sublethal Chemical Doses.....	114
(f) Chemical Alarms.....	114
(2) Definition of the Term - Casualty.....	114
(3) Overview of the Chemical Lethality Methodology	114
(a) Personnel	115
(b) Equipment	117
c. Chemical Lethality Subroutines and Their Function	117
(1) Calculation of Alarm Activation Time.....	117
(2) Calculation of Liquid Contamination (per Target Point)	117
(a) Main CNTM Subroutine	118
(b) Calculation of Contamination Arrival and Evapora- tion Time	118
(c) Calculation of Evaporation of Contaminant	119
(3) Computation of Travel Time for a Chemical Cloud	119

(4)	Initialization of Chemical Degradation Values.....	119
(5)	Evaluation of Degraded Performance Due to Chemical Dose(s)	119
(6)	Removal of Assets from Dose Bins.....	120
(7)	Calculation of Vapor Dosage (per Target Point).....	120
(8)	Change of Personnel MOPP Posture	121
(a)	Main MOPP Subroutine	121
(b)	Calculation of Posture Change Time	121
(c)	Reset of Personnel to Original MOPP Posture.....	121
(9)	Calculation of Personnel Dose Response	122
(10)	Accumulation of Chemical Dosages	123
(11)	Calculation of Personnel Permanent Zero Performance Time	126
(12)	Calculation of Chemical Casualties.....	126
(13)	Calculation of Amount of Vapor Dosage Over Time (per Target Point).....	126
d.	Application of the Hypothetical Case to the Chemical Lethality Model	127
4.	Nuclear Vulnerability Model	129
a.	Triggering a Nuclear Scenario.....	129
b.	Overview of Nuclear Vulnerability Model	130
c.	Inputs.....	130
(1)	Weapons	130
(2)	Unit Deployment	130
(3)	Shielding Factors	131

(4) Nuclear Vulnerability file.....	131
(5) Degradation Input for Sublethal Dose of Radiation	132
d. Nuclear Vulnerability Methodology.....	132
(1) Personnel	133
(2) Equipment	133
e. Nuclear Vulnerability Subroutines and their Functions	134
(1) Calculation of Probability of Kill due to Nuclear Attack.....	134
(a) PK for Personnel.....	134
(b) PK for Equipment.....	135
(2) Calculation of Nuclear Environment	136
(3) Calculation of Electro-magnetic Pulse in an EMP Environment.....	139
(4) Calculation of Probability of Survival from Neutron Flu- ence.....	139
(5) Calculation of Probability of Survival due to Nuclear Blast Dose	139
(6) Calculation of Probability of Equipment Survival from Neutron Fluence	140
(7) Calculation of Casualties Resulting from a Nuclear Attack.....	141
(8) Accumulation of Radiation Dosages.....	141
(9) Calculation of Time Dependent Casualties from Radia- tion Dosages.....	143
(10) Calculation of Probability of NOT Experiencing Early Transient Incapacitation.....	143

(11) Calculation of Probability of Personnel Survival as Function of Radiation Dose and Time.....	143
(12) Calculation of Degraded Performance as Function of Radiation Dose and Time.....	144
f. Application of Hypothetical Case to Nuclear Vulnerability Model.....	145
5. Combined Weapon/Lethality Effects in AURA.....	148
VI. Current Efforts in AURA Methodology.....	149
VII. Summary.....	150
REFERENCES.....	151
DISTRIBUTION LIST.....	155

LIST OF ILLUSTRATIONS

Figure 1. The AURA Family of Methodologies	4
Figure 2. General Illustration of AURA Constructs	8
Figure 3. AURA's Coordinate Systems	13
Figure 4. Operation of the AURA Model	15
Figure 5. General Flowchart of the AURA Model.....	17
Figure 6. ASP Layout	22
Figure 7. ASP Functions.....	23
Figure 8. General Form of a LINK Effectiveness Curve Illustrating the Link Parameters	28
Figure 9. Examples of Link Effectiveness Curves	29
Figure 10. Hierarchy and Relationships of AURA Constructs	33
Figure 11. General Form of a Link Effectiveness Curve	45
Figure 12. Link Effectiveness Calculation	47
Figure 13. Effectiveness Derivations of AURA Constructs	48
Figure 14. Links and Associated Effectiveness Curves for Unit Assets.....	49
Figure 15. Segments Representing the Subtasks of Example Unit	50
Figure 16. Chain Structure for the Example Unit	51
Figure 17. SLUNIT Accumulation (Sleep Efficiency) Curve.....	66
Figure 18. SLUNIT Expenditure and Recovery	67

Figure 19. Relationship between SLUNIT Balance and Job Performance	68
Figure 20. SLUMIN, SLUMAX, and SLPMIN	70
Figure 21. Graphical Descriptions of AURA's Delivery Errors.....	76
Figure 22. The Unit Deployment (xy-coordinate) and the Incoming Attack (x'y' coordinate) systems.....	85
Figure 23. Examples of Conventional Lethality Footprints	102
Figure 24. Range to Deflection Ratio (RY/RX) versus Fall Angle for HE munitions.....	107
Figure 25. Example of Unit Effectiveness as a Function of Denial Radius	111
Figure 26. Illustration of the Chemical Probit Slope	124

I. Introduction

1. Background

In the mid 1970's, the U.S. Army analysis community was faced with the problem of quantifying the expected survival of certain combat capabilities in the event of a European war. The study team that was formed to address the problem consisted of 10-20 different agencies, with specialties ranging from signature and acquisition to vulnerability, unit structure and manpower. It was soon apparent that a major hurdle in producing an overall evaluation was the need to incorporate the widely different, highly detailed technical data into a coherent analysis. It was to fill this need that the large, integrated family of methodologies, known as AURA, was conceived.

AURA, the Army Unit Resiliency Analysis methodology, is a large, interconnected collection of analysis models which provides a detailed evaluation of the ability of a military unit to accomplish a series of missions in a combat scenario. Briefly, AURA is an event sequenced, one-sided combat simulation methodology. The methodology consists of an (expanding) number of highly detailed models from the various technical communities interfaced into a large, time-dependent event playing and optimization routine. The interfaces are varied, involving such diverse kill probabilities as lethal footprints for conventional munitions, log normal kill probabilities for nuclear effects, toxic chemical dispersions and evaporations, MOPP degradation, reliability, and target acquisition probabilities. The optimization is a dedicated, non-linear routine which models the commander's reallocation of surviving, degraded assets in order to minimize the choke points in the optimal functional path. The logic process required the development of a general model for the functional structure of a military unit. Such a model was developed and forms an essential part of the AURA methodology.

AURA is distinctive from other models in several ways. First is the versatility and breadth of the AURA model. Driven by user defined inputs, AURA accepts a great deal of detail in describing the unit structure, weapon capabilities, and unit mission to be modeled. These inputs are simultaneously and interactively integrated with a broad range of phenomena at an equal level of detail. Second, unlike many models, AURA inputs are directly related to physically measureable quantities such as the interrelationships between unit tasks and asset deployment. Finally, AURA's most important distinction is the ability to measure unit effectiveness via a realistic commander model, known as the AURA Asset Allocation Algorithm. The calculation of the highly detailed profile of unit effectiveness is based not only on the capability of surviving (and degraded) assets but also on the commander's decisions regarding the reallocation of these assets.

In the hierarchy of combat simulation models, AURA is designed to assess combat unit effectiveness at the battalion level or lower. AURA is capable of assessing larger units (upwards of 1200 people have been used), however, modeling units of larger than battalion size results in the loss of the meticulous detail provided by AURA.

The Integrated Battlefield Assessment Branch, Vulnerability/Lethality Division (VLD), Ballistic Research Laboratory (BRL), is the developer, maintainer and a primary user of AURA. However, many other agencies now run the AURA model, as do a number of federal contractors. The AURA code is currently run on UNIVAC, VAX, IBM, CDC and Data General computers, as well as the CRAY 'supercomputers' on which it resides at the BRL.

As generalized above, the AURA model (formerly called the Residual Combat Capability [RCC] Model) is an amalgamation of analysis techniques, algorithms, and data sources gathered from the laboratories that specialize in the various areas which impact upon the resiliency of a military unit. AURA currently has the capability to model such phenomena as:

- * Quantified Unit Effectiveness (including asset reallocation);
- * Deployment of Personnel and Equipment (including dynamically changing deployment postures);
- * Incoming Weapons;
- * Conventional Lethality;
- * Chemical Lethality;
- * Nuclear Vulnerability;
- * Equipment Repair;
- * Personnel Factors including:
 - Degradation due to Fatigue/Sleep Deprivation;
 - Degradation due to wearing of chemical protective ensemble (MOPP);
 - Degradation due to Heat Stress;
 - Suboptimal training;
 - Sub-lethal dose effects.

As a result of its breadth and versatility, AURA is finding application in a variety of studies conducted by a number of agencies. This growth in the number of ongoing studies is increasing the number of analysts who conduct AURA studies and who, therefore, need to fully understand all aspects of the AURA methodology.

2. Scope

This report is designed to provide a comprehensive understanding of the methodologies of the Army Unit Resiliency Analysis (AURA) model. A hypothetical unit with a corresponding mission is provided and used throughout the report to describe practical examples of each AURA methodology. The AURA methodologies are described in detail, concentrating on the areas such as the AURA Asset Allocation Algorithm (the commander's decision model) which were derived especially for AURA, and the spectrum of methodologies which have been combined to form the AURA model. Figure 1 illustrates the AURA "family of methodologies" as well as the corresponding source of each contributing methodology.

AURA is unique in its ability to integrate highly detailed inputs into these methodologies and produce a quantitatively meaningful measure of unit effectiveness. Since this measure is highly dependent upon the AURA Asset Allocation Algorithm, the first part of this report describes the derivation and methodology of this model. Subsequent sections describe the remaining methodologies which combine to comprise the AURA model.

II. Technical Background

The following sections provide the general terminology and concepts necessary to understand the AURA methodologies described in this report.

1. AURA Terminology

ASSETS - Personnel and equipment belonging to and deployed with the unit.

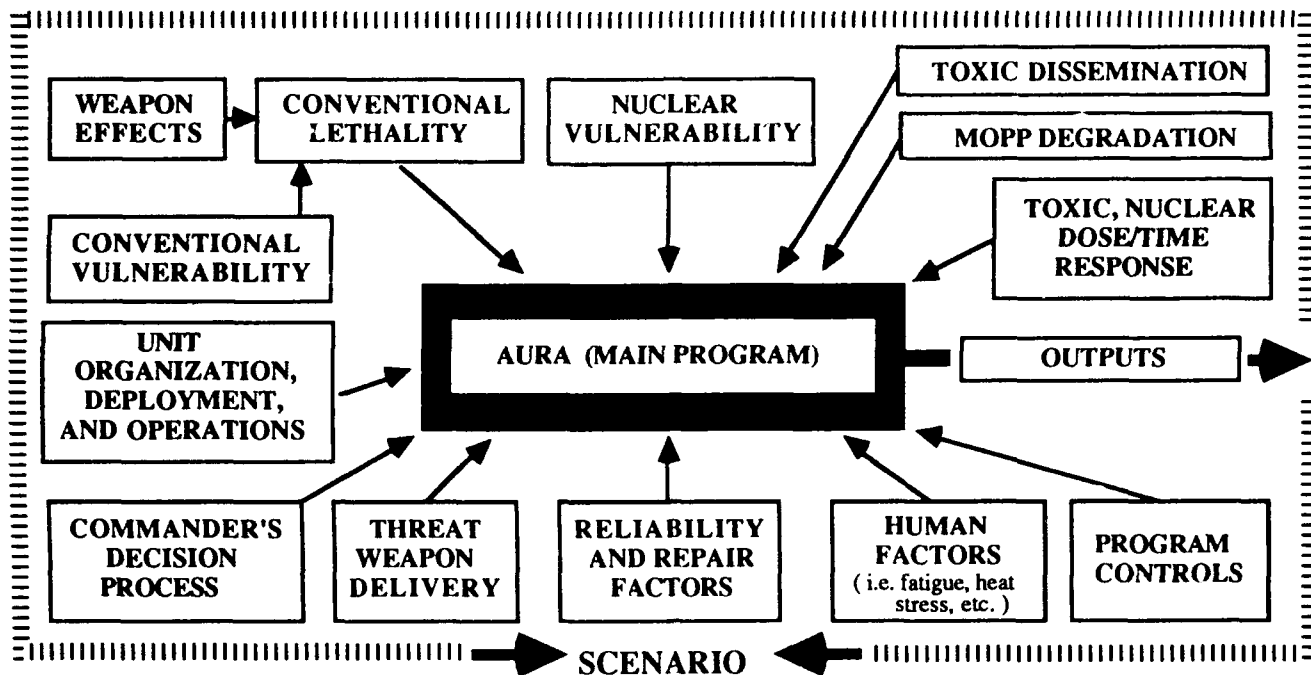
BASELINE - Initial set of input data from which all related AURA executions for a specific study will be generated.

CASUALTY - The removal of a personnel asset from the simulation. This is a function of the kill criteria selected by the user input.

CHAIN - A collection of AURA constructs which span the activities of a unit to represent the unit mission. A chain is the highest level AURA construct. The mission being modeled may be comprised of one or more chains.

CONSTRUCT - A task or combination of tasks. Each construct indicates a specific relationship between its elements (i.e. an ORLINK represents an 'OR' relationship, SUBCHAIN represents an 'AND' relationship, etc.).

COMPOUND LINK - An AURA construct used to represent several groups of



Methodology	Source
WEAPON EFFECTS	AMSAA, BRL, and JMEMs
CONVENTIONAL LETHALITY	AMSAA
CONVENTIONAL VULNERABILITY	BRL
NUCLEAR VULNERABILITY	HDL and DNA
TOXIC DISSEMINATION	CRDEC and HEL
MOPP DEGRADATION	BRL and HEL
TOXIC, NUCLEAR DOSE/TIME RESPONSE	DNA (Nuclear), USANCA (Nuclear/Toxic), CRDEC and MRDC (Toxic)
HUMAN FACTORS	MRDC
RELIABILITY & REPAIR FACTORS	AMSAA and Ordnance School
THREAT WEAPON DELIVERY	DIA, AMSAA, and BRL
COMMANDER'S DECISION PROCESS	BRL
UNIT ORGANIZATION AND OPERATIONS	TRADOC School(s), ITAC, FSTC, DIA

AMSAA - Army Materiel System Analysis Activity
 BRL - Ballistic Research Laboratory
 CRDEC - Chemical Research, Development, and Engineering Center
 DNA - Defense Nuclear Agency
 DIA - Defense Intelligence Agency
 FSTC - Foreign Science & Technology Center
 HDL - Harry Diamond Laboratories
 HEL - Human Engineering Laboratory
 ITAC - Intelligence Threat & Analysis Center
 JMEM - Joint Munitions Effectiveness Manual
 MRDC - Medical Research and Development Command
 USANCA - U. S. Army Nuclear and Chemical Agency

Figure 1. The AURA Family of Methodologies.

tasks/jobs such that the total activity is a summation of the independent contributions of the different constructs. A compound link may be comprised of orlinks, subchains, crews, and/or links.

CREW - An AURA construct used to represent sets of subtasks that must work together to accomplish part of the mission. The relationship between these subtasks are generalized parallel structures and all crew tasks are links.

DUMMYLINK - A job which has no asset of the same name. Dummylinks are links which have no particular assets assigned to them but are filled, when needed, by substitutes.

EXCURSION - An execution of the AURA model in which specific input parameters are varied from the baseline to determine the sensitivity of the results to these particular input parameters.

FUNCTIONAL STRUCTURE - The organization of activities and tasks that result in the accomplishment of a unit's mission. These activities are represented by the organization of AURA constructs. A detailed description of the organization and content of the AURA functional structures is included in section 1.2.

HOMELINK - A link which has the same name as the asset that fills it. An asset is immediately available for its 'homelink' task and contributes to the homelink's accomplishment at 100 percent effectiveness unless degraded otherwise.

LETHALITY - The ability of a weapon to inflict damage.

LETHALITY FILE - The input file containing the lethality data for the environment(s) being modeled.

Environment	Lethality file
Conventional	FORTTRAN unit # 2
Nuclear	FORTTRAN unit # 3
Toxic/Chemical	FORTTRAN unit # 4

LETHALITY EVENT - A process by which the AURA model simulates the effects (damage, contamination, dosage, etc.) resulting from a conventional, chemical, or nuclear attack.

LINK - The lowest task/job delineation of all AURA constructs. Links may be used as independent constructs or combined to represent any higher AURA construct.

MISSION - Performance rate or capability upon which the study unit is being evaluated. For example, a supply unit might be evaluated on its ability to issue

2000 tons of supplies per day.

OPTIMIZATION - The process by which the AURA model allocates available assets in the attempt to maximize the effectiveness of the unit. The optimization process is performed over all mission constructs which can be improved during the course of the mission.

ORLINK - An AURA construct representing mutually exclusive choices for the accomplishment of part of the mission. Orlinks may be comprised of subchains, crews, and/or links.

RECONSTITUTION - A 'snapshot' of the unit's status taken at specified time periods within an AURA execution.

RECONSTITUTION EVENT - A process by which the AURA Asset Allocation Algorithm (the commander's decision model) simulates the reallocation of surviving assets in order to optimize the unit's mission accomplishment.

REPLICATION - One complete pass through the events of an AURA model. Due to variances resulting from random number generation, several (50 or more) replications are generally specified and the results are averaged over all replications. This process insures a more statistically sound representation of the unit effectiveness.

RUNSTREAM - The input file (FORTRAN unit # 5) containing the AURA commands and parameters required to model a unit mission scenario.

SEGMENT - An element in a chain. A segment can be a Compound Link, Orlink, Subchain, Crew, or a Link.

SLUNIT - (SLeep UNIT). Unit of measure used to represent the amount of rest associated with a personnel asset. 1 Slunit = 1 minute of fully efficient rest.

SUBCHAIN - An AURA construct used to represent a group of tasks which must work together to accomplish part of the mission. A subchain may be comprised of crews and/or links.

SURVIVOR - An asset that is still available to the unit after an attack.

UNIT EFFECTIVENESS - The capability of the unit to perform its selected mission. Note: The unit effectiveness is evaluated based upon its ability to perform selected mission(s) which have an associated rate defined as 100% performance.

VULNERABILITY - Susceptibility of a target to damage.

WEAK LINK - The individual task(s) in a chain which has the most harmful effects on the capability of the construct.

WEAPON - Incoming weapon. The term 'weapon' includes warheads and delivery systems.

2. AURA Functional Structures

Since the performance of, and relationship between, individual jobs is one of the most complicated facets of any human joint venture, it is inevitable that a realistic, yet general, tool for modeling a unit must itself appear complex. The approach taken in constructing the functional structure and associated optimization portions of AURA was to isolate and quantify subtasks, and then describe the relationship between them. This approach appears to fit well with the way that unit personnel think of their units. It is, therefore, easy to model units using information gathered through literature searches and by asking questions of field experienced people. Examples of such questions are:

- 1) What tasks/jobs are done in your unit ?
- 2) Who does them ?
- 3) How well and how fast can the tasks be accomplished ?
- 4) What is the task flow? , i.e., Where does a job normally commence ? What happens from there ?, etc.
- 5) What are the variations on the above ?

In AURA, the organization of activities and tasks that result in the accomplishment of a unit's mission(s) are known as **FUNCTIONAL STRUCTURES**. It should be noted that the functional structure is made up of jobs that are done, not the assets that do them. (The connection between job positions and assets that can fill the positions is made through the **LINK** input parameters.)

It is convenient to consider the functional structure for a mission as being constructed through three levels of aggregation. The lowest delineation of subtasks or job positions is called a **LINK**. For example, one might define a radio operator's job as a **LINK** if one were not concerned with subtasks within the job.

Jobs combine together in different ways to perform the various activities that take place within a unit. For example, an artillery unit might have several activities occurring at one time during a fire mission: firing, loading, fire direction calculation and new position reconnaissance. The AURA name for such an activity, which is the second level of aggregation, is **SEGMENT**. Segments are made up of links; however, the various links within a segment may be combined to show different relationships. Thus, a segment may be a single job (**LINK**), a group of jobs that must be done together (**CREW** and/or **SUBCHAIN**), a choice between groups of jobs (**ORLINK**) or several groups of jobs such that the total activity is a summation of the independent (weighted) contributions of the

different groups (COMPOUND LINK).

The essential feature of segments is that they all contribute to the mission in such a way that mission accomplishment is limited by the weakest segment. The collection of segments, which span the activities of the unit toward mission accomplishment is called a CHAIN, the final level of aggregation in the AURA functional structure model. Figure 2 depicts the physical representation of the AURA constructs which form the unit's functional structure.

A complete derivation of the AURA functional structures is provided in section 5 of this report.

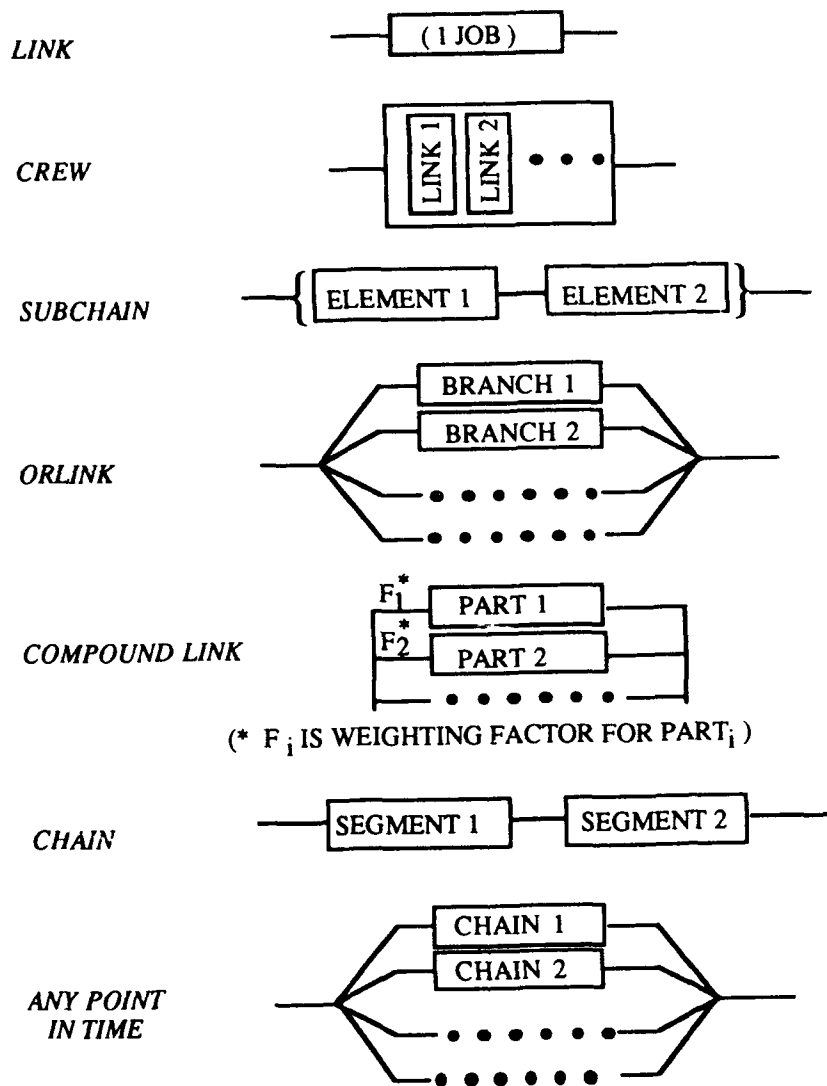


Figure 2. General Illustration of AURA Constructs

3. Statistical Modes used in AURA

AURA analyses can be run in either of two modes: deterministic or stochastic. In this context, "stochastic" refers to analyses in which probabilistic phenomena are handled by making binary choices ((0 or 1), here or not here, killed or not killed, etc.) in such a way that, after many replications, the results of the choices reflects the underlying probability distribution. Stochastic models generally involve the use of random number (Monte Carlo) techniques. On the other hand, "deterministic" refers to using a probability distribution as though it were an actual material distribution. Thus, if an item has 0.4 chance of being at point A, a deterministic approach will put 0.4 of an item at A, whereas 100 stochastic runs will find 1 item at point A in about 40 iterations.

a. **Deterministic Mode.** The original, and currently the default, mode for AURA is deterministic. Actually, even the deterministic mode involves some stochastic features. For example, the selection of actual weapon burst points is done stochastically in both modes: in each iteration, every weapon is given a specific burst point which reflects a particular, probability-weighted choice of acquisition and delivery errors. Similarly, the exact time in which each individual changes posture and the precise manifestation time of certain nuclear radiation effects are stochastically chosen. However, two major features are modeled deterministically: the deployment and kill of assets.

In both deterministic and stochastic modes, assets may be fractionally assigned. For example, a single man in a fire direction center may be split between communicating and plotting tasks. In the deterministic mode, that fractional assignment is manifested in a fractional deployment: the man would be partially deployed at the communicating location and the rest at the place where plotting is done. Similarly, if there are two equally likely locations at which the commander might be located, the deterministic mode will deploy 0.5 of the commander at each point.

The second important deterministic area is in assessing kills. If a weapon hits at a distance from point A such that the probability of killing a particular item is 0.6, then upon proper functioning of such a weapon, 60% of all such items located at point A will be designated killed.

Note that the fractional procedures involved in deterministic modeling can be compounded. Thus, if 0.4 of an asset is assigned to a job which had four possible locations (0.25 each) and a weapon causes a 0.3 probability of kill at one point, the computed loss at that point would be:

$$0.4 * 0.25 * 0.3 = 0.03$$

Although clearly involving significant assumptions, the deterministic mode has the outstanding advantage of requiring fewer replications for convergence, which will be discussed below.

b. Stochastic Mode. As indicated above, the stochastic mode does not resort to fractional assets to account for various probabilities. Rather, assets are treated as entities and probabilities are used as probabilities. As such, the stochastic mode is more realistic. It is also more "instantaneous", in that it makes specific decisions (e.g. exactly where is the asset at this instant) rather than using average values.

In the areas of asset kills and deployment, the stochastic mode proceeds as follows: first, the handling of kills is straightforward. When a round has hit, the probability of kill (P_k) for each asset is determined by the lethality routines (as in the deterministic mode). However, the whole asset is then either killed or not depending upon the value of the random number (RN) drawn from a uniform distribution in the range 0 to 1. If RN is less than P_k , the item is dead; otherwise, it survives.

Deployment in the stochastic mode begins with the same (optimum) job assignment algorithm as used in the deterministic mode. Thus, fractional assignments may be made. From the tasks which have a non-zero assignment, the stochastic deployment must then select the specific task that each asset will be filling at that specific instant in time. To do this, random numbers are drawn against the fractional assignments. Then, AURA interrogates all target points at which each occupied task can be done. Again, random numbers are drawn, this time against the fraction of the task done at each location. As before, the relationship between the random number and the task fraction determines the deployment of integral amounts of the asset.

An advantage of using the stochastic mode is the ability to compute variances. These are totally unknown in the deterministic mode.

c. Comparison of AURA's Statistical Modes. Because of the greater realism offered by the stochastic mode, one might ask why the deterministic mode is even offered, let alone chosen as AURA's default. The reasons are two-fold. First, it was brought out that several replications are required to properly sample the probability distributions involved in each stochastically modeled feature. In the deterministic example given above, the answer was 0.03. For sake of a quantitative example, assume the number of successes (kills) to be normally distributed such that the standard deviation in the number of kills, K , is given by $\text{SQRT } K$. If we want a reasonable assurance that the answer is within 10%, we require an error in the number of kills, divided by N (the number of replications), such that:

$$\sqrt{K} / N < 0.003$$

Assuming that K is approximately $0.03 * N$, we have $N > 3300$! The problem is intensified by the compounding effect of several interplaying stochastic models. In the above example, a stochastically modeled weapon reliability of 0.3 would have increased the required number of iterations to 10000! Clearly, confidence in low probability events is expensive with a stochastic model.

The second reason for using the deterministic option is unique to AURA. In order to support the deterministic model, a special accounting procedure was devised for maintaining personnel dosages in AURA. A quirk of the system is the need to uniquely name each asset in the stochastic mode. Thus, while the user can define a class of persons (e.g. "driver") and deploy several of them in deterministic mode, it is currently necessary to uniquely name each person ("driver1", "driver2", etc.) in the stochastic mode.

In summary, AURA provides the user the flexibility of balancing the realism of a simulation with the attendant demands for computer time. If desired, the user can track each individual person and piece of equipment through several replications in which the various probabilistic phenomena randomly interplay. Such an application may be correct for a detailed study of a particular factor. On the other hand, when a large number of runs are required, the model allows the user the ability to efficiently converge to average results.

4. AURA's Coordinate Systems

Three natural coordinate systems are used in AURA to express data that refer to geographical locations or extents. These coordinate systems are defined as follows:

a. **The Unit Coordinate System.** Any right-handed coordinate system may be used to lay out the study unit. Any particular point in the unit, such as the geographical center or "lower left" corner, may be designated as the origin of the unit coordinate system, and all other unit elements are deployed relative to that point. Note: All target and deployment points from the fire direction and wind direction coordinate systems are defined relative to points within the unit coordinate system.

b. **The Incoming Fire (Range-Deflection) System.** Threat weapon parameters are specified in range and deflection, where range is in the direction of the incoming fire, and deflection is perpendicular to range such that range-deflection define a right-handed, horizontal coordinate system. The user can specify the orientation of the range direction relative to the unit coordinate system.

c. **The Wind Direction (Downwind-Crosswind) System.** Chemical cloud dispersion is specified in the downwind and crosswind directions, which are defined to form a right-handed, horizontal coordinate system. The user can specify the orientation of the downwind direction relative to the unit coordinate system.

Usages of the AURA coordinate systems is summarized in Table 1, while Figure 3 illustrates the AURA coordinate systems.

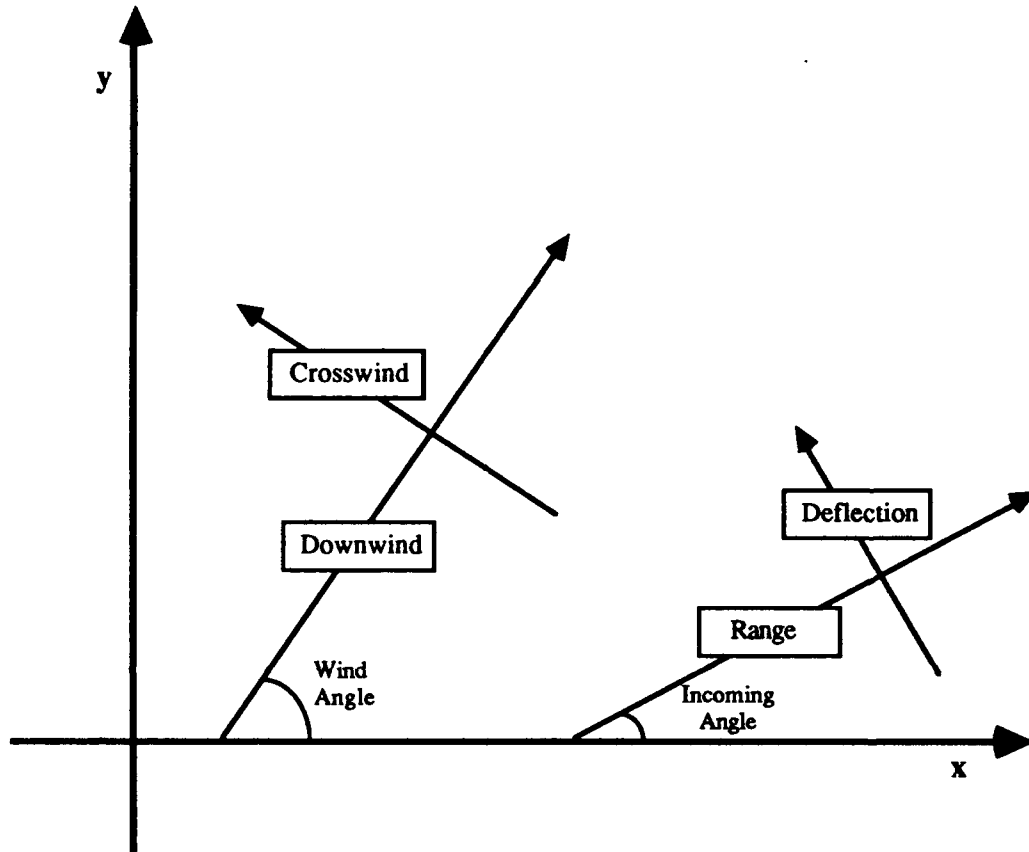
TABLE 1. Coordinate Systems for Geographically Related Parameters

Parameter	Coordinate System	Comment
Deployment of items	UNIT	Define X-Y System
Aimpoint	UNIT	e.g. Signature Point
Target Location error	RANGE-DEFLECTION	Measured from aimpoint
Delivery errors	RANGE-DEFLECTION	Measured from aimpoint
Burst point of munitions	UNIT	Internally computed by AURA
Conventional weapon effects (Lethal radii)	RANGE-DEFLECTION	Measured from burst point
Chemical contamination and vapor clouds	DOWNWIND-CROSSWIND	Measured from burst point
Volley parameters (length, volley, etc.)	RANGE-DEFLECTION	Measured from aimpoint

5. General Information

a. **Operation of the AURA Model.** As stated above, AURA is a family of methodologies covering a broad spectrum of technical areas. Figure 1 depicted the methodologies (and associated sources) which comprise the AURA model. As shown, the various models are interfaced together into a combat simulation through a computer program which is also called AURA. Volume 2 of this report contains a comprehensive explanation of the organization, source code preparation and execution, and algorithms of the AURA model. The general operation of the AURA computer code is described in Table 2 and is illustrated in Figure 4.

UNIT Coordinate System



The Incoming Fire and Wind Direction coordinate systems are defined relative to the Unit coordinate system. AURA ultimately converts any coordinates specified in Incoming Fire (or Wind Direction) coordinates to Unit system coordinates.

Figure 3. AURA's Coordinate Systems

Table 2. General Operation of AURA

1. User inputs the runstream data, which includes:
 - a. Scenario
 - b. Threat
 - c. Lethality
 - d. Unit Information:
 1. Mission(s);
 2. Assets (personnel/equipment);
 3. Organization and Operation.
2. The code processes the data and sets up the simulation.
3. The code runs the simulation several times.
 - a. Time dependent phenomena are updated before each event
 - b. Lethality events cause damage, contamination, dosages, etc.
 - c. Reconstitution events cause the commander to reallocate his surviving, degraded assets in order to optimize his unit's mission accomplishment
4. The code outputs total and averaged statistics for:
 - a. Mission accomplishment;
 - b. Asset survival, degradations, dosages, etc;
 - c. Reasons for shortcomings;
 - d. Decisions made (by commander);
 - e. Other items and actions, selected by the user.

b. **Event Types.** As described in Table 2, the third stage in the general operation of the AURA model is the processing of the different event types. In AURA, an event is associated with a specific instant in time within the scenario. The processing sequence of the events is user-specified within the input runstream, and represents the aggregation of actions which combine to simulate the scenario. The types of AURA events are: reconstitution events, lethality events, and phenomena such as changes in delivery errors/target location errors, known in AURA as 'other' events. Each of the event types are described below.

A reconstitution event is the process by which the commander simulates taking an inventory of surviving assets and allocating them to subtasks in an effort to improve the effective capability of the unit. By use of the RECONSTITUTION option, the user may specify the time points at which a reconstitution should occur, thus providing a 'snapshot' of the unit's status at any given time point during the simulation.

Since a major function of AURA is to measure the effects of events upon a unit, the user generally wants reconstitutions and evaluations to take place at specified times relative to certain events, rather than at specific 'clock' times in the scenario. An example of such an event is a lethality event (discussed in next section). Rather than specifying the effectiveness status at 100, 200, and 1000 minutes into the scenario, the user may be more concerned with the effectiveness

General Processing Sequence

Events

Methodologies

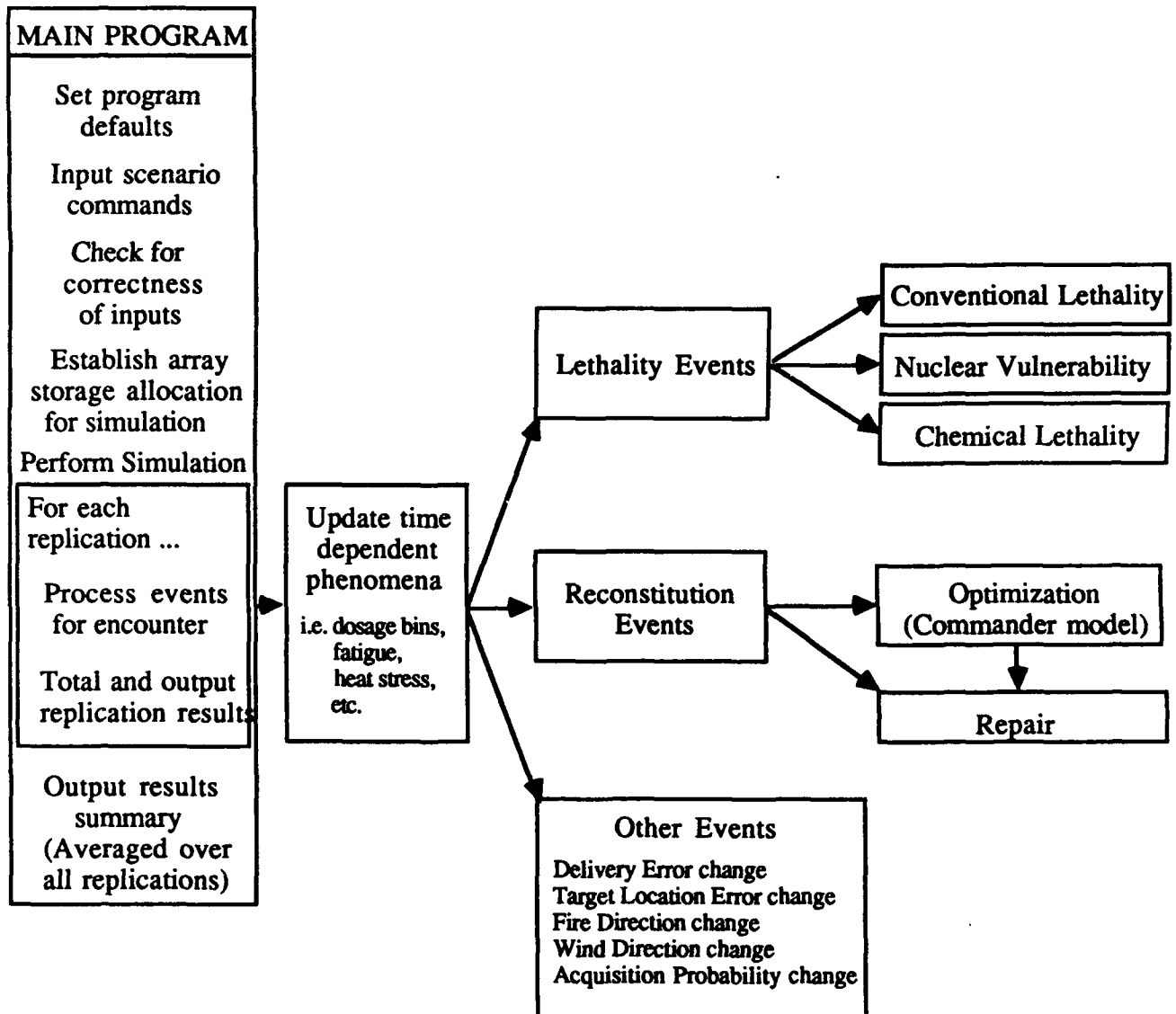


Figure 4. General Operation of the AURA Model

at 100, 200, and 1000 minutes after the arrival of a warhead. The INTERNAL RECONSTITUTION TIMES input option enables the user to specify the relative time points to trigger a reconstitution event.

A lethality event is the process by which the AURA model simulates the effects resulting from a conventional, chemical, or nuclear attack. Lethality events are designated time points which signify the arrival of a round (or volley of rounds) and the computation of immediate effects.

Table 3 outlines the weapon effects methodologies which have been incorporated into the AURA model. A comprehensive description of the role of these methodologies in AURA is the subject of Section 4 of this report.

Table 3. AURA Weapon Effects Models

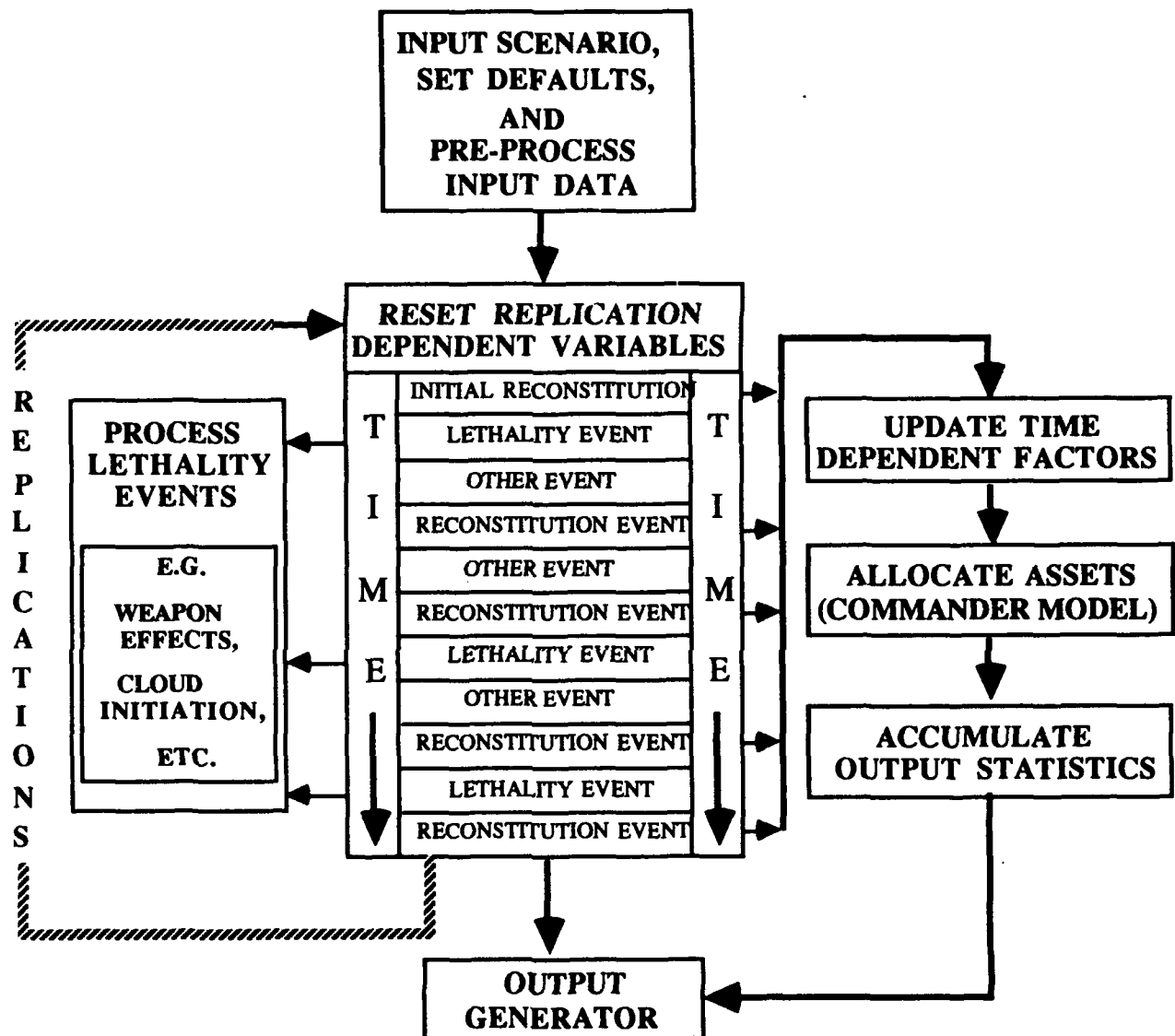
Conventional Lethality - Derived from outputs of JTCG/ME FULL SPRAY Model¹
Nuclear Vulnerability - NUDACC Methodology²
Chemical Effects - NUSSE3 Methodology³

In AURA, phenomena which impact the general operation of the scenario are categorically called 'other' events. The user can establish the parameters for each 'other' event in the input runstream. For example, in modeling a conventional scenario, suppose the user wants another attack to occur at the 5 hour mark, employing new weapon delivery errors. At the 5 hour mark, AURA will perform its calculations based upon the new delivery errors for the weapon. This would be considered a delivery error change event which falls under the 'other' events as follows:

Delivery Error;
Target Location Error;
Incoming Fire Direction;
Wind Direction;
Acquisition Probability.

-
1. "Computer Program for General Full Spray Material MAE Computations", Joint Technical Coordinating Group for Munitions Effectiveness, 61 JTCG/ME-79-1-1, (SECRET)
 2. Vault, W.L., "Vulnerability Data Array: The Agreed Data Base - Final Report (U)," Harry Diamond Laboratories, HDL-TR-1906, JUL 80, (SECRET).
 3. Saucier, R., "NUSSE3 Model Description", U.S. Army Chemical Research, Development and Engineering Center, CRDEC-TR-80746, May 1987, (UNCLASSIFIED).

Figure 5 depicts a flowchart of the general processing of the AURA model using an arbitrarily selected event processing sequence. Recall, 'other' events are user inputs which are specified in the runstream. In Figure 5, the 'other' events are assumed to have been declared in the input.



NOTE:

The event ordering sequence shown here has been arbitrarily selected. The ordering of events is dependent upon the scenario specified in the input runstream.

Figure 5. General Flowchart of the AURA Model

c. **AURA Inputs.** As described in Table 2, operation of the AURA model requires such inputs as: weapon threat/lethality data, unit organization, and the components of the mission to be simulated. These inputs are organized into files created by the user. The two types of AURA input files are the runstream and weapon threat/lethality file. Table 4 briefly describes the contents of the AURA input files. A detailed description of the organization and contents of the AURA input files are contained in Volume 2 of this report.

Table 4. AURA Input Files

File	Contents
runstream	Assets (personnel/equipment) in unit Deployment of assets Weapon parameters Unit organization/functional structure parameters Program controls etc.
lethality file(s)	Weapon effects data for: Conventional Lethality; Chemical Dispersion; Nuclear Vulnerability.

In Volume 3 of this report we will describe the methods by which to prepare the inputs needed to perform an AURA analysis. Included in the data preparation section of Volume 3 will be detailed examples of how to create both an AURA runstream and threat/lethality file.

d. **AURA Outputs.** Analyses involving the AURA model can generate large amounts of data. It is possible, for example, to output the impact point of every incoming round: for 100 replications of a study involving a heavy artillery barrage, the impact point output alone could consume upwards of 10,000 pages of computer paper. For this reason, AURA provides the capability to optionally output only those entities that are of interest to the analyst for his/her specific needs. When no options are invoked, the defaults in AURA result in a moderate amount of output which includes a consolidation of the inputs and a report of the final, average result at each time point.

Table 5 provides a general outline of the primary AURA outputs. The collection of AURA output options/commands is described in detail in the AURA Input Manual.⁴

Table 5. General Outline of AURA Outputs

I. Consolidation of Inputs

- A. Commands specified
- B. Table of events to be processed
- C. Weapon information
- D. Assets (personnel/equipment)
- E. Functional structure of mission
- F. Link substitutability matrix
- G. Deployment table/plot

II. Intermediate Results

- A. Actual weapon impact points
- B. Casualties, contaminations
- C. Chemical or Nuclear dosages
- D. Repair status
- E. Optimization status (decisions made by commander model)
- F. Replication summaries

III. Final Results -versus- Time

- A. Effectiveness, statistics, and distribution
- B. Survivors
- C. Degradation due to fatigue, dosages, or contaminations
- D. Job status (LINK table)
- E. Mission results (CHAIN table)

IV. Averaged Results (over all replications)

- A. Repair results
- B. Asset status
 - 1. Degradation
 - 2. Reliability failures
- C. Unit effectiveness

4. Klopceic, J.T., "Input Manual for the Army Unit Resiliency Analysis (AURA) Methodology: 1988 Update", USA Ballistic Research Laboratory, Technical Report No. 2914, MAY 88 , (UNCLASSIFIED)

Volume 3 of this report will provide a complete 'walkthrough' of an AURA study with an emphasis on the analysis of the entire spectrum of AURA outputs.

III. Scope of AURA Methodology

1. A Hypothetical Case Study

Unit effectiveness is specifically tied to a quantifiable unit mission such as to fire x number of rounds per minute, move x number of meters per hour or load x number of stacks per day. While unit effectiveness is considered to be the primary measure of effectiveness (MOE) provided by the AURA model, there are others as well. MOEs are chosen based on the intent of each analysis and may include such things as the number of personnel casualties or deaths, the number of damaged and/or killed items of equipment, the number of repairs made and identification of tasks that are limiting the effectiveness of the unit.

Typically, analyses are accomplished through the systematic variation of the vulnerability and lethality data inputs of interest. The impact of these changes on the chosen MOEs form the database from which weapons effects and unit vulnerabilities may be assessed.

For illustrative purposes in discussing the AURA methodology, a hypothetical case study of an ammunition supply point is provided; this case study is based upon a previous BRL analysis of the ASP. Throughout the discussions of the targeting, lethality and asset allocation algorithms, this example will be used to discuss specific phenomena of interest.

2. The Hypothetical Case - An Ammunition Supply Point

a. **Unit Discussion.** The Ammunition Supply Point (ASP) is the primary source of conventional ammunition in the division sector. The ASP is responsible for establishing and operating ammunition supply facilities for the receipt, storage, rewarehousing and issue of conventional ammunition. This is the ASP's primary mission, which is accomplished through use of materials handling equipment (MHE), such as forklifts and cranes. The unit modeling was based on a 24 hour, 2-shift operation. Included within the mission of the ASP is the ability of the unit to maintain a finite level of ammunition to supply the user unit's requirements. Although the unit maintains a 3-5 day supply of ammunition, it is assumed that only one-third of this amount was mission essential, that is, the amount needed to satisfy user requirements until resupply could be affected. The ASP mission may be characterized as a rate based on the above information: load and unload 465 short tons of ammunition per day. The following table is a list of the types and amounts of ammunition maintained by the ASP.

Category	Type of Ammunition	Short Tons
A	Fixed and semifixed ammunition	242.4
B	Separate loading artillery projectiles	709.5
C	Mortar and hand grenades	78.9
D	Pyrotechnics and chemical	118.5
E	Demolition explosives	203.6
F	Rockets, rocket motors, rifle grenades and anti-tank	42.4

The ASP has 215 personnel. Equipment required to fulfill the unit mission are forklifts and cranes of which the unit has eleven and five, respectively. Also organic to the unit are various trucks and trailers as well as communications equipment. The unit was deployed with each unit member, piece of equipment and stack of ammunition assigned a specific Cartesian coordinate. Care was given to the establishment of an effective road network to allow the easy flow of resupply vehicles throughout the unit. Figure 6 contains a simplified view of the ASP layout.

The functions inherent to the ASP are summarized in Figure 7. These functions were defined in terms of those tasks, performed by personnel or pieces of equipment, which are necessary to provide each required capability.

This unit has a number of physical attributes which make it an interesting case study. First, the presence of ammunition stacks makes the unit particularly vulnerable to conventional attack. In addition to being damaged by fragmenting artillery rounds, the ammunition stacks may also be detonated by direct hit. The nature of this problem includes not only the problem of subsequent fires but the potential for a chain reaction of secondary explosions. Typically, areas of terrain and equipment around potential secondary explosion sources are restricted from use in an ASP for reasons of safety in the event of such an occurrence. The detonation of the ammunition also presents further threat to personnel and equipment.

Personnel are the most vulnerable attribute of any unit. They are susceptible to the effects of conventional fragmentation, chemical dosage and mission oriented protective posture (MOPP) degradation, and nuclear blast, thermal and radiation effects. Consequently, much detailed attention has been paid to the handling of these effects against personnel. Personnel in the ASP were deployed in the following types of conventional postures: in the open-standing, in the open-prone, forest-standing and forest-prone. Typically, personnel reduced their vulnerability by falling to the ground (prone posture) upon any incoming round. In the special case where the scenario dictates a chemical attack, however, all personnel may first be required to change their MOPP posture. In the ASP, personnel were initially in MOPP2 (overgarment and overboots worn, mask/hood and gloves carried) changing to MOPP4 (entire ensemble, closed) on any incoming round.

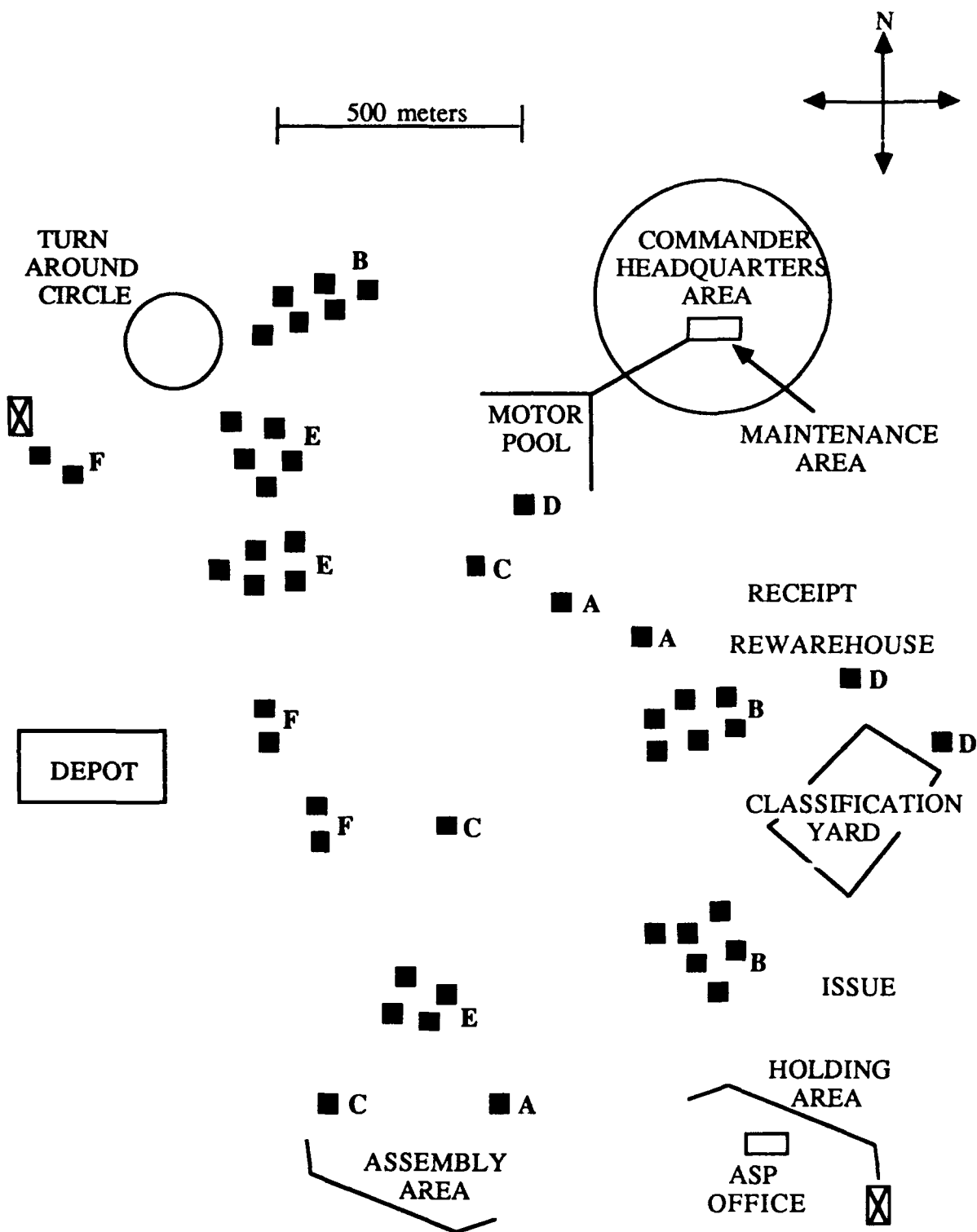


Figure 6. ASP Layout

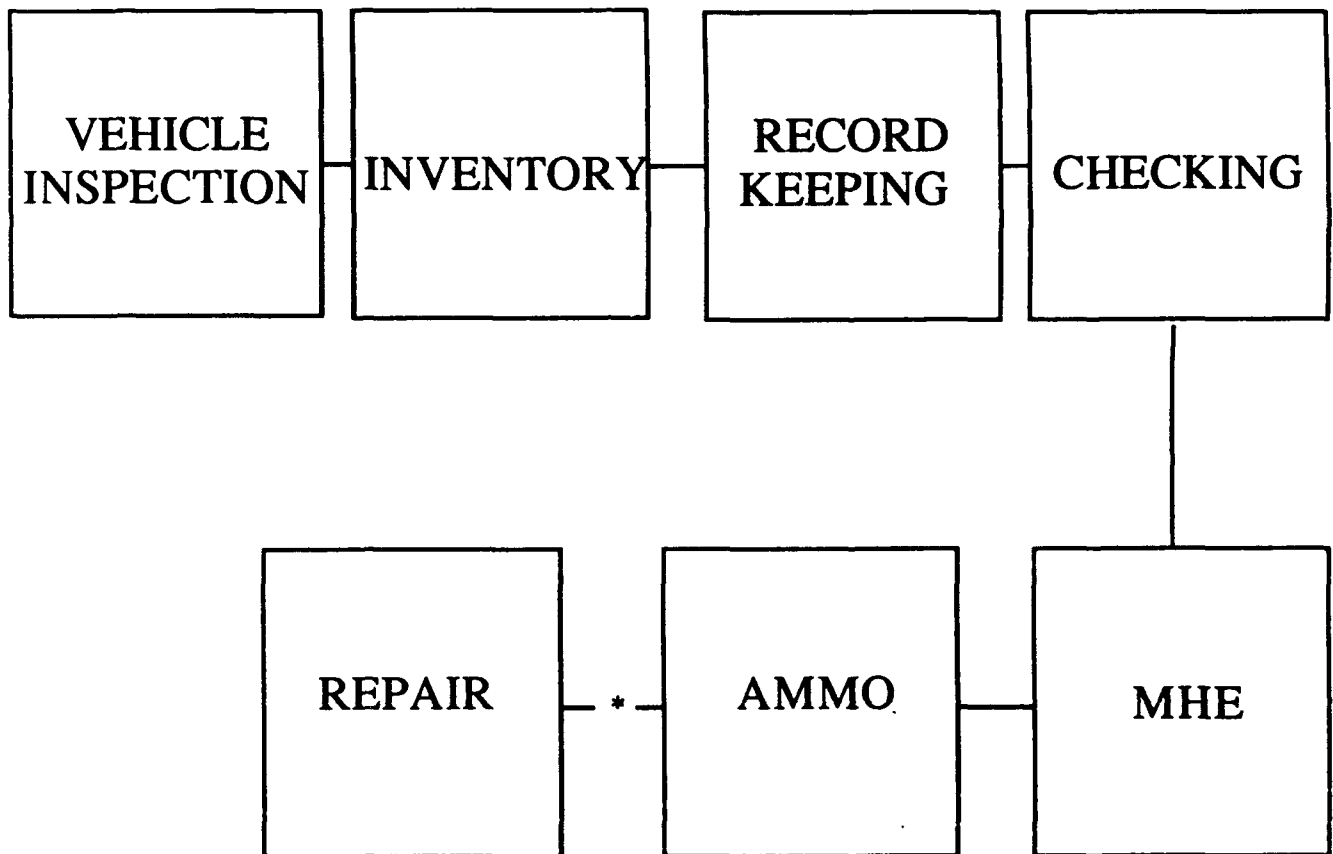


Figure 7. ASP Functions

As with personnel, equipment is susceptible to conventional fragmentation and blast, chemical contamination and decontamination, and nuclear electromagnetic pulse (EMP), transient radiation effects on electronics (TREE), blast or thermal damage. Reliability failures and repairs are also considered as is expenditure of repair items such as spare parts. A related chemical phenomenon which affects equipment is chemical agent persistence. The standard chemical persistence time produced by the toxic dissemination code (NUSSE) relates to the persistence of the agent on grassland. One must account for the persistency of the agent on different surface types (such as paints or tires) through use of an AURA option which allows the user to input a factor by which the NUSSE calculated persistency is multiplied.

b. Applications of the AURA Model. There are a number of areas and/or problems that can be addressed using the AURA model. A sample will be discussed to give the reader an idea of the various uses.

One of the initial applications of the AURA model was the analysis of tables of organization and equipment (T.O. and E.) structure and cross-training of unit personnel. This allowed several proponent schools (such as the Quartermaster School and the Missile and Munition Center and School) to analyze various T.O. and E. changes and decide on the best formulation of the unit.^{5 6 7} It also allowed the user to analyze the impact of personnel cross-training. Using the substitution matrix required by the AURA model, changes could be made and their impact on unit operations studied.

Equipment reliability can also be studied with the AURA model. Using first a baseline case with current reliability values, the user could then make changes to the reliability numbers and determine their impact on unit operations.⁸ A

5. Stark, M.M., Klopac, J.T., "The Resiliency of an Ammunition Supply Point to Combat Damage (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2614, December 1984, (SECRET).

6. Stark, M.M., Klopac, J.T., "The Resiliency of Ammunition Supply Points in a Pre-Defined, Integrated Battlefield Scenario (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2609, November 1984, (SECRET).

7. Roach, L.K., "The Resiliency of a Supply and Service Company to Combat Damage (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2669, August 1985, (SECRET).

8. Juarascio, S.S., "Evaluation of an 8-Gun M109A2 Artillery Battery with Replacement of Combat Damaged Mission Essential Components (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2682, October 1985, (SECRET).

related application is the analysis of the nuclear and Nuclear, Biological, and Chemical (NBC) contamination survivability criteria of equipment. Again, baseline cases using the current survivability criteria are analyzed, followed by cases using alternate survivability criteria. This allows the user, for example, the USA Nuclear and Chemical Agency (USANCA), to study the impact of criteria changes. This approach has been used in the past⁹ by USANCA to determine what impact relaxed criteria have on unit operations.

A major use of the AURA model over the past few years has been the analysis of the degradation associated with operations in an NBC environment.¹⁰

¹¹ These studies have concentrated mainly on the effects of MOPP on personnel performance and unit operations. Related phenomena are the effects of heat stress and heat stress casualties on unit operations and the ability of personnel to operate unit equipment while in MOPP.

Other applications of the AURA model include analyses of weapons effectiveness and changes in delivery accuracy.¹² Also, as with the ASP, studies have looked at the benefits of improved survivability of ammunition stacks and the impact of reducing distances between the stacks.¹³ The applications discussed in this section are only a sampling of the types of analyses that have been, and can be, performed using the AURA model.

9. Juarascio, S.S., Abell, J.M., "Impact of Chemical Survivability Criteria on Unit Performance (U)", USA Ballistic Research Laboratory Report No. BRL-TR-2870, November 1987, (SECRET).

10. Stark, M.M., Klopac, J.T., Juarascio, S.S., "The Effects of Chemical Contamination/Decontamination of a M109A2 8-Gun Artillery Battery (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2633, February 1985, (SECRET).

11. Roach, L.K., "The Effects of Chemical Contamination/Decontamination on an M1 Tank Company (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2723, April 1986, (SECRET).

12. Roach, L.K., Juarascio, S.S., "The Resiliency of Selected Soviet Units to Candidate Chemical Warheads for Use in the Corps Support Weapon System (CSWS) (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2724, April 1986, (SECRET).

13. Stark, M.M., Klopac, J.T., op. cit., BRL-TR-2614

IV. Reconstitution Methodologies

This section describes the modeling techniques, which have been derived especially for the AURA model, for use during the reconstitution process. These methodologies include: the AURA Asset Allocation Algorithm (commander's decision model), the repair/failure model, and the fatigue/heat stress models. A detailed discussion of the methodology, organization, description, and algorithms is provided for each of these models.

1. The AURA Asset Allocation Algorithm

The AURA Asset Allocation Algorithm (AAAA) is the method by which the unit commander's decision making process is modeled in AURA. The general function of the AAAA is to optimally allocate surviving, possibly degraded assets in an effort to maximize the effective capability of a unit to perform a mission. This section describes the organization and design of the methodology which embodies the AAAA.

The formulation of the asset allocation model in AURA was guided by a number of essential factors:

1. Asset allocation is driven by mission accomplishment. Optimum allocation of surviving, possibly degraded, assets is that allocation which results in the best accomplishment of the overall job or jobs that the unit must perform. The asset allocation model is, therefore, inseparably tied to the unit functional description model.
2. Military units vary widely, especially in their functional descriptions. To be of general use, a functional description model must be flexible enough to describe many possible relationships between assets. These relationships should be user-specifiable during input preparation.
3. To be easily usable, the elements of the functional model should have a recognizable association with real-world elements. Similarly, although generally more difficult to describe, the relationships between model tasks should intuitively resemble the relationships between tasks actually done in the unit.
4. A somewhat subtle, but pervasive factor: the asset allocation model must be compatible with the mathematical behavior of the input data. Thus, for example, if assets are to be degraded (such as a man working at half the normal rate), then the asset allocation model cannot be intrinsically an integer model.
5. The output must be quantitative and must reflect the ability of the

unit to perform the specified mission. The reason for any shortfall must be traceable.

Using these factors as a guideline, the AURA Asset Allocation Algorithm was formulated. The first step was to define a fundamental building block and introduce quantification. Following some work done for the Theater Nuclear Force Survivability Study by BDM (in a model called Combat Capability Degradation Model (CCD))¹⁴, the individual subtask, which is called a LINK, was chosen as the building block. Fundamental in CCD is the assumption that the ability to do a subtask depends upon the amount of assets allocated to that subtask.

From that point, the AURA Asset Allocation Algorithm deviates from CCD. First, the functional relationship between the effectiveness of a subtask with respect to overall mission completion and the amount of assets allocated is generalized in the form shown in Figure 8. To illustrate the flexibility of the form, note that the three graphs in Figure 9 are just different cases of the same function, viz., a straight line from the left to the point (N_0, E_0) , a slanted line up to the point (N_{100}, MAX) , and a straight line off to the right. The curves in Figure 9 describe the different kinds of actual jobs:

TOP CURVE: There is an optimum amount of assets (number of men, e.g.), shown as N_{100} , at which the subtask effectiveness reaches its maximum. Allocating more assets does not add anything to the task, while taking away assets reduces it.

MIDDLE CURVE: Same as TOP CURVE, except this job requires a non-zero minimum number of assets (N_0) to begin to get any effectiveness in the task.

BOTTOM CURVE: Some "tasks" have a residual effectiveness, even with no assets assigned. For example, a well-trained crew can function without a supervisor. Thus, although the N_{100} amount of supervisory assets may be required for maximum performance, the effective supervisory function drops only to E_0 at 0 assets.

14. Baum, W., "Theatre Nuclear Force Combat Capability Degradation Methodology, Development and Demonstration", (U), 2 Volumes, Interim Note No. SV-7, April 1978, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland. (SECRET)

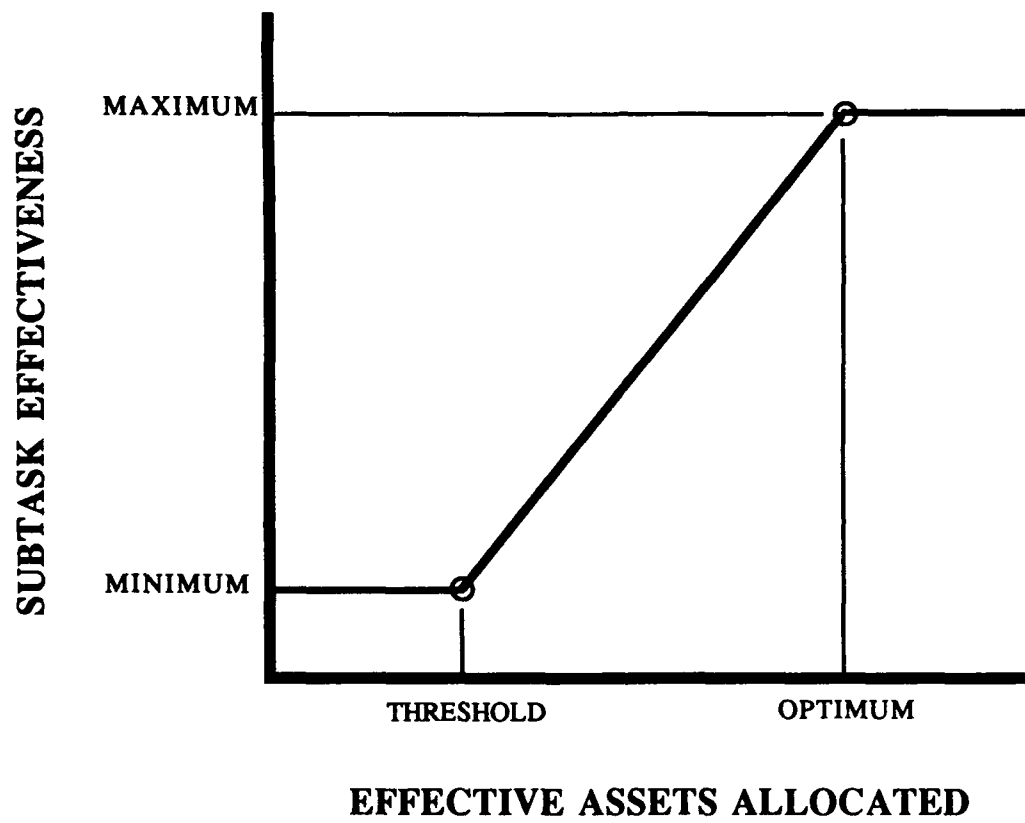
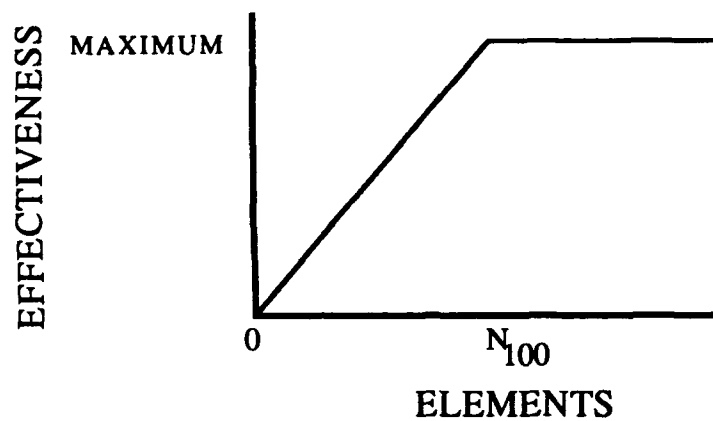
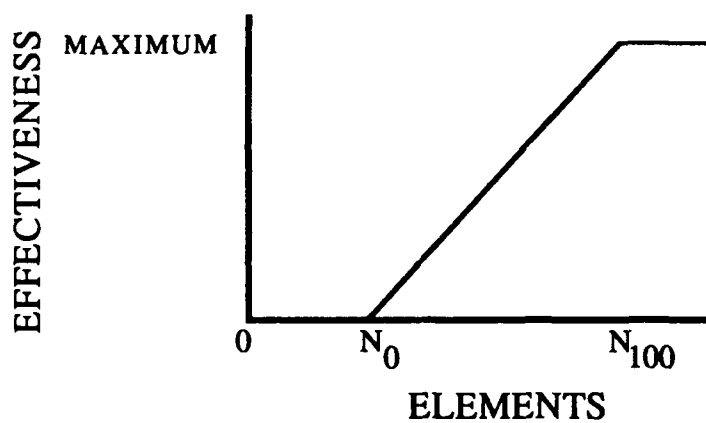


Figure 8. General Form of a Link Effectiveness Curve

TOP CURVE



MIDDLE CURVE



BOTTOM CURVE

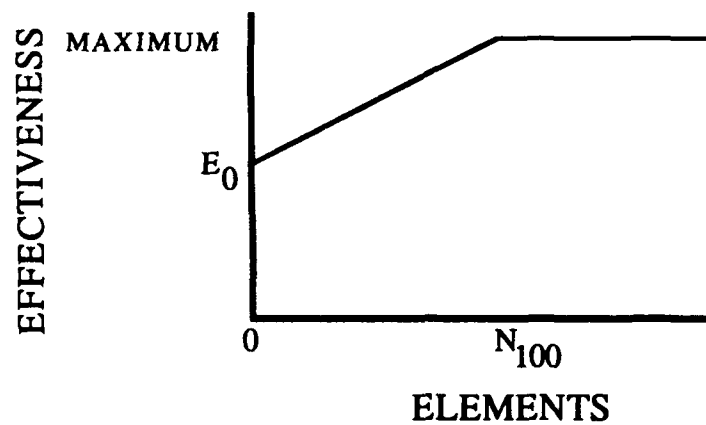


Figure 9. Examples of Link Effectiveness Curves

The next step is to generalize the meaning of "amount of assets" to include degradations. Thus, putting a less than fully capable man into a subtask, or degrading the performance of a man (by putting him into protective clothing, by imposing radiation sickness, etc.) is equated to allocating less than a whole asset to the subtask. Since the effectiveness curves (Figure 9) are continuous, this generalization required no change in the functional structure.

(Note, however, that the use of continuous (non-integer) functions was made possible by the development of the non-integer (and non-linear) optimization algorithm described below.)

This mathematical model of a subtask, which is called a LINK, meets the requirements for a quantitative basic building block. It is easily associated with real jobs (driving a tank, receiving radio messages), and smoothly allows for non-integer assets. The parameters are also possible to evaluate.

In the evaluation of an artillery battery, for example, one can ask:

How many gunners are needed ($= N_{100}$)?
Can they do this mission ($= \text{MAX}$)?
What happens if there are none ($= E_0, N_0$)?

While evaluating the LINK parameters, it is also convenient to ask:

Who normally does this job?
Who can substitute?
How well (how much slower) does the substitute perform?
How long does it take for the substitute to take over?

The data obtained from the above questions form the substitution matrix, which is another fundamental part of the AAAA. Note that substitutes are governed by time to substitute and relative ability, as well as operational and physical degradations.

Once the link parameters have been evaluated, the next step for the AAAA is to account for the different relationships that subtasks (LINKs) could have to each other with respect to overall mission accomplishment. First, some tasks require that other tasks also be effective in order to contribute to the overall mission. In AURA, this relationship may be represented in two ways: CREWS and SUBCHAINS. A CREW is described as a parallel construct (made of LINKs) which must work together to perform a task. In a CREW, the members are able to "cover" for each other in an effort to improve the overall effectiveness of the CREW. For example, the effectiveness of a job that requires 3 assets (crew members) may be increased if one of the crew members has a high level of effectiveness. That is, the highly effective crew member can assist one or more of the less effective crew members resulting in an increase in the overall crew

effectiveness. However, a crew member having 0% effectiveness cannot be improved and will cause the overall crew effectiveness to be 0%. The mathematical model for AURA's CREW was derived from a crew performance analysis by the Pacific-Sierra Research Corporation in conjunction with a study sponsored by the Defense Nuclear Agency.¹⁵

The second type of relationship is a SUBCHAIN. For example, the job that a forklift operator does is useless without the job that the forklift itself does, and vice-versa. Mathematically, this can be thought of as an AND relationship between the forklift and forklift operator LINKs. In AURA, LINKs having an AND relationship are said to form a SUBCHAIN.

To summarize, the relationship between tasks in a CREW are generalized parallel structures, as opposed to strict ANDs as are those in a SUBCHAIN. The user specifies the CREWs and/or SUBCHAINS he/she wishes to form via the input runstream.

Note: A SUBCHAIN is a higher order construct than a CREW. That is, the elements of a SUBCHAIN may be CREWs or LINKs.

Another possible relationship between LINKs, CREWs, and SUBCHAINS is an exclusive OR: In such cases the user wants the code to choose the best of several alternate ways of accomplishing some function, where each choice (branch) may be composed of a single subtask (LINK), a CREW, or a SUBCHAIN. For example, it may be possible to use the forklift team (SUBCHAIN composed of forklift and forklift operator) to load a truck, or to handload it manually (i.e., by using a CREW of men.) In AURA, the specification of alternative procedures is done via the ORLINK construct.

(Note: The ORLINK is used for alternative "procedures", that is, combinations of subtasks, which perform the same function. The use of alternative "personnel" or equipment to perform the "same" procedures is automatically done by the optimization algorithm for every subtask. In AURA, a great deal of care was taken to differentiate between subtasks (LINKs) and the people/equipment which perform those subtasks. The distinction between ORLINKs (choice of tasks) and substitutions (choice of performers) is just one indication of that differentiation.)

15. Dore, M.A., Anno, G.H., "Effects of Ionizing Radiation on the Performance of Selected Tactical Combat Crews", (U), Pacific-Sierra Research Corporation under contract to the Defense Nuclear Agency, PSR Report No. 1846, Contract No. DNA 001-85-C-0352, July 1988, (UNCLASSIFIED)

There are some jobs which involve a number of procedures that are additively related. For example, consider loading a truck with 75 percent light and 25 percent heavy items, which requires two different loading procedures. Clearly the relationship between the two procedures is not an OR (only one is chosen), nor is it an AND (no capability unless both are accomplished). Rather, the total fraction of the truck loaded is a weighted sum of the light and heavy loading capabilities, where the weighting factors, 0.75 and 0.25, reflect the relative demands. To model this relationship, AURA has a construct called COMPOUND LINKS (CPLINKs). CPLINK parts can be ORLINKs, SUBCHAINs, CREWs and/or simple LINKs.

The next higher level of aggregation is the CHAIN, a series of ANDs. The segments of a CHAIN can be CPLINKs, ORLINKs, SUBCHAINs, CREWs and/or simple LINKs. This level of ANDs is convenient for assembling a complete mission, i.e., command AND control AND communication AND transportation AND.....

Finally, a unit can be given a number of missions to consider at any given time. Each mission is described by a CHAIN. Each CHAIN has a series of time intervals during which the associated mission is to be done. If two time intervals overlap, implying two missions competing for the unit's attention during that time, AURA chooses the CHAIN which can be done most effectively. This, in effect, gives the user the capability of an overall OR relationship between CHAINs.

The hierarchy of relationships was depicted in Figure 2 and summarized in Figure 10. Note that, in all cases, the constructs (CHAINs, ORLINKs, etc.) can be made of combinations of any of the lower echelon constructs. This ability to combine simple tasks, quantified in terms of effective assets, into a wide variety of complex structures gives the AAAA the flexibility to be applied to a broad spectrum of unit types and missions.

a. Mathematical Description of the AURA Asset Allocation Algorithm. The mathematical optimization algorithm used in AURA to allocate assets is a sizable extension of the GREEDY Algorithm. Basically, the GREEDY algorithm solves a maximization problem by a sequence of single steps. At each step, the algorithm selects that choice, consistent with the imposed constraints, which produces the maximum gain in the value function (the function whose value is to be maximized). In the case of AURA, the value function is the effectiveness of the unit, which is dictated by the value of the choke point. The GREEDY algorithm, therefore, indicates allocation of asset(s) to the weakest segment until its capability has improved above that of some other segment, which then becomes the weakest. This process continues until a point is reached at which no further allocation can be made. Intuitively, this process corresponds to a commander considering his mission's activity demands one at a time, in order of

Construct	Operator	Operands	Operand may be:
Point in Time	Exclusive OR	CHAINS	CHAIN
CHAIN	AND	Segments	COMPOUND LINK ORLINK SUBCHAIN CREW LINK
COMPOUND LINK	Weighted Sum	Compound Link parts	ORLINK SUBCHAIN CREW LINK
ORLINK	Exclusive OR	Branches	SUBCHAIN CREW LINK
SUBCHAIN	AND	Elements	CREW LINK
CREW	"Parallel"	Positions	LINK
LINK	Fundamental building block		

Figure 10. Hierarchy and Relationships of AURA Constructs

decreasing stringency, and allocating just enough assets to satisfy each demand before considering the next one. The GREEDY algorithm has been found to be a good model of human decision making in several important classes of problems.¹⁶ Unfortunately, like greedy humans, the GREEDY algorithm guarantees the optimal solution for only a limited class of value functions and constraints.¹⁷

In AURA, a unit whose members were completely cross-trained would fall into that class. However, in more common units, it is possible to "fool" the greedy commander, which mathematically corresponds to making a series of allocations which lead to a local maximum. The most likely way of doing this is to specify a unit structure and substitution effectiveness matrix in such a way that the algorithm chooses asset A over asset B early in the optimization process, then cannot fill a task that only A can do later in the sequence. To avoid this error, AURA incorporates three processes - preventative look-ahead, local dynamic look-ahead, and first-order look-back - which intuitively correspond to "experience", "limited foresight", and "limited error correction".

In AURA, preventative look-ahead is accomplished in two ways. First, a preprocessor evaluates the versatility (i.e. the number of possible job assignments) of each asset. In all subsequent allocation events, assets are considered in inverse order of versatility. In effect, the commander assigns his least useful assets first, reserving his more- assignable ones for later.

Secondly, before every allocation event, the algorithm counts the actual number of remaining assets which could possibly be assigned to each segment, then orders the segments for consideration in increasing order of potential assignees. This corresponds to the commander knowing that some jobs may be hard to fill and considering those jobs ahead of more redundant ones.

Local dynamic look-ahead is applied to improving segments which require a number of LINKs (job positions) to be filled before any gain is realized. In the truck loading example, the commander can load light parts by using a forklift and operator or by assigning several men to handload. When AURA considers the first alternative, the assignment of an operator is made tentatively; only when the forklift is found to be assignable and the gain from the two assignments is recognized is the set of assignments made "firm". Should the forklift not be available, the operator remains available for assignment to the handloading crew.

16. Lawler, E., "Combinatorial Optimisation: Networks and Matroids", Holt, Rinehart and Winston (1975), (UNCLASSIFIED).

17. Papadimitriou, C. and Steiglitz, K., "Combinatorial Optimization: Algorithms and Complexity", Prentice Hall (1982), (UNCLASSIFIED).

This "look-ahead" is applied within every segment optimization step; however, it is not done from segment to segment.

Finally, first order look-back is applied as follows. If a point is reached at which a needed asset is unavailable, the algorithm checks all previous assignments of assets which could satisfy the need. If such a previous assignment is found and there is an unassigned asset available which could be substituted for the needed one, a "switch" is made: the unassigned asset takes the place of the needed one, and the needed one becomes available for reassignment to the current choke point. This look-back is limited to one level, however; C cannot replace B so that B can replace A so that A can become available.

In AURA, the commander considers the possibility of improving unit performance through repair/decontamination activity. In order to effect repairs, the commander must allocate the personnel and equipment to repair/decontamination tasks IN ADDITION to the tasks required for his mission. After determining an optimum allocation of resources to perform this augmented mission, he weighs the immediate cost in mission performance versus the possible gain in deciding whether or not to include the repair/ decontamination activity. Thus, the repair/decontamination model constitutes a fairly complex look-ahead process. A detailed description of the repair/decontamination methodology is discussed in the following section. (Note: Unlike the other processes discussed above, inclusion of repair/ decontamination alternatives is optional.)

The success of the algorithm in solving actual unit assignments has improved over the years as the "commander has become smarter" (i.e., as the above look-ahead and look-back features were added) Mistakes by the algorithm in actual practice have become fairly rare, and are usually traceable to unlikely arrangements of skills (e.g. a senior person with a unique, non-essential capability and no capability to do the tasks of his juniors), in conjunction with a complex unit functional structure. As with any analysis tool, however, the final judgement lies in the hands of the analyst. For this reason, AURA also includes several output options which can be used to analyze results in detail, including the commander's decisions which lead to the results.

b. Asset Allocation Algorithm Decision Rules. The decision rules followed by the asset allocation algorithm (the "commander") in assigning assets to LINKs are as follows:

HOMELINK - A LINK is filled by its HOMELINK asset (an asset having the same name as the LINK) if one is available. If no HOMELINK asset is available, the commander will attempt to fill the LINK with a substitute. Also, if the available HOMELINK asset is degraded (e.g. because of sickness or fatigue) below a user-settable level (sicklv) and if there is a substitute available at a performance

level more than (1/sicklv) greater than the best HOMELINK asset, then a substitute will be selected.

SUBSTITUTES - A potential substitute does not become available until the elapsed time (time since the need for a substitute developed) exceeds the substitute's (user-specified) substitution time (see LINKS, section IV.E.2.). If more than one substitute is available in the elapsed time involved, a particular substitute is chosen by the following criteria.

EFFECTIVENESS - Any potential substitute which is more than a user-settable level (signif) less effective than the best substitute is automatically dropped from consideration.

VERSATILITY - The commander will assign a less versatile asset in preference to assigning a more versatile asset. (Versatility, an integer number, is defined as the number of LINKs to which an asset can be assigned. AURA internally predetermines the versatility of each asset by analysis of the substitution matrix.)

USER-INPUT-ORDER - The allocation algorithm numbers the substitutes for a particular LINK in the order in which they were named (see LINKS, section IV.E.2). The commander will assign a lower-numbered substitute in preference to a higher-numbered one. (Note, however, that several assets may have equal order numbers, since several substitutes may be specified by the same common name.)

The normal operation of the allocation algorithm is to take the decision criteria in the order presented above: a decision passes to the next criterion only if there is a "tie" in all preceding criteria.

The decision rules and values can be modified by the user. As stated above, the user can set the values of signif and sicklv. Furthermore, the order of consideration of criteria 2 and 3 (VERSATILITY and USER-INPUT-ORDER) can be reversed.

Finally, note that any decision made by the above rules can be over-ruled by a correction made through the look-back capability of the algorithm, as described in the preceding section.

c. **Subroutines of the Asset Allocation Algorithm.** A number of algorithms exist within AURA which embody the methodology of the AURA Asset Allocation Algorithm. The algorithms described in this section provide an understanding of the optimization process and the derivation of effectiveness for each AURA construct.

(1) **The Commander Algorithm.** Subroutine OPTMIZ coordinates the optimization process for AURA's asset allocation scheme. (The optimization process is the means by which the AURA model determines how to best allocate surviving assets in an effort to maximize the effective capability of the unit to do a mission.) OPTMIZ serves two functions in the optimization process. First, OPTMIZ serves as the "manager" for the overall optimization process by determining which structures can and should be optimized. Secondly, OPTMIZ is responsible for optimizing chains, the highest level of constructs. Subroutine OPTMIZ determines the most effective chain by traversing each chain to determine the weakest (i.e. the least effective) segment of the chain. If a weak segment exists, the asset allocation algorithm optimizes the segment by apportioning those available assets which can be assigned to maximize the effectiveness. If a weak segment cannot be optimized, the chain effectiveness remains only as strong as its weakest segment. OPTMIZ processes all chains by this means and stores the most effective chain to be used in the determination of overall unit effectiveness at any user defined time period.

Within the optimization process, if a weak segment can not be optimized to the goal effectiveness, it is referred to as the construct's "choke point". If a choke point is found, the attained value of variable EFFSEG is that chain's effectiveness. The fundamental premise for the optimization process is:

A CHAIN IS ONLY AS EFFECTIVE AS ITS WEAKEST SEGMENT.

The general flow of the optimization process is based upon the hierarchical structure of the segments that make up each chain. Table 6 outlines the optimization process by depicting the hierarchy of AURA constructs and the corresponding algorithms used to calculate the effectiveness of the segments. Included in Table 6 is the description of variable IXX which specifies the structure type of each segment. The values of IXX are used throughout the optimization algorithms (described below) to specify the type of segments comprising each structure.

Table 6. Algorithms of the AURA Optimization Process

Construct	Optimization Subroutine	Effectiveness Evaluator	Segment Type (variable IXX)
CHAIN	OPTMIZ	OPTMIZ	
COMPOUND LINK	CPLOPT	CPLEFF	IXX > 2000
ORLINK	ORLOPT	ORLEFF	IXX > 1000
SUBCHAIN	SBCOPT	SBCEFF	IXX < 0
CREW	CRWOPT	CRWEFF	IXX < -1000
LINK	LNKOPT	LNKEFF	IXX = any other value

The optimization algorithms process each construct by breaking down the construct into the (lower-lying) segments, calling the appropriate optimization algorithm, and evaluating the construct effectiveness. This process is performed for each construct to optimize all lower-lying segments. The optimization of a construct is considered to be complete when the lowest level of each segment (links) has been evaluated.

(2) Optimization of Compound Links. Subroutine CPLOPT is the compound link optimization algorithm. The compound link parts are processed in the order of their potential to improve the effectiveness of the compound link. This potential is referred to as the "promise" of the compound link part. The best result for compound link effectiveness is stored as variable EFF.

First, CPLOPT determines the number of compound link parts that have the most "promise" for optimization (NP). NP can equal zero only if the promise of all potential compound links is equal to zero, in which case CPLOPT should not have been called. All compound link parts are processed to determine if each part has enough assets to properly function.

CPLOPT then determines the type of the compound link part (i.e. ORLINK, SUBCHAIN, etc.) and calls the appropriate optimization algorithm to determine the effectiveness of the part. For example, if the compound link part is a subchain, subroutine SBCEFF will be called from subroutine SBCOPT to calculate the subchain's effectiveness.

If there has been an improvement in effectiveness, the most promising compound link part is retained. The optimization algorithm requires an effectiveness increase of more than 0.1% for a compound link part to be considered improved.

If the PRIORITY option has been specified, the compound link parts are considered in order of entry and failure to improve any compound link part will terminate the optimization process. Otherwise, the compound link parts will be considered by their versatility.

If the compound link effectiveness cannot be improved, the ineffective compound link part contains the weak link. Failure to improve one compound link part terminates the compound link optimization process. Program control is then returned to subroutine OPTMIZ via the failure (alternate) return.

(3) Optimization of Orlinks. Subroutine ORLOPT is the orlink optimization algorithm. The highest attained result for orlink effectiveness is stored in variable EFF.

ORLOPT first processes all of the orlink branches to determine which are candidates to be optimized. The status of each orlink branch is assigned a value in the variable IORFL. The possible values for IORFL are as follows:

IORFL = 1 means active branch;
IORFL = 0 means untried branch;
IORFL = -1 failed when not being used;
IORFL = -2 failed when active.

If there are no active orlink branches, (which can only happen if all branches for this orlink have failed) the failure return is taken. Otherwise, the type of orlink branch is determined and the appropriate optimization subroutine for the orlink branch type is called. If the branch is a subchain, subroutine SBCOPT is called to optimize the subchain. Likewise, CRWOPT is called for crews and LNKOPT for links.

If the currently active branch cannot be improved, all repair (if any) information for the branch is stored. IORFL is set to -2 to indicate that the branch failed while active and ORLOPT attempts to optimize another branch. For each branch, ORLOPT calculates the branch type and calls the corresponding optimization subroutine as described above. If a branch cannot be improved, IORFL is set to -1 (failed when not being used). Finally, if there was successful improvement of the orlink, the improved effectiveness is assigned to variable EFF and the results are returned to the calling subroutine.

(4) Optimization of Subchains. Subroutine SBCOPT optimizes subchains. Subchains work like chains in that they are only as effective as their weakest element. Subchains may only be comprised of crews and/or links. The best result for subchain effectiveness is stored in variable EFF.

SBCOPT checks all subchain elements to determine the type of construct comprising each element. Once the construct type is found, the appropriate construct effectiveness subroutine is called (i.e. CRWEFF for crews, LNKEFF for links, etc.). If there are no weak links, the subchain is 100% effective and no optimization is required. In this case, EFF is set to 1.0 and program control is returned to the calling program. NOTE: If the link cannot be improved, the original effectiveness value is returned.

However, if there is a weak link, SBCOPT will try to improve the subchain element by calling the appropriate optimization subroutine (i.e. CRWOPT for crews, LNKOPT for links). If the weak element cannot be improved, the subchain cannot be improved. This is considered an optimization failure and program control is returned to the calling subroutine.

In the case where this subchain is part of a compound link, there is a unique check which must be made. If the subchain effectiveness is being reduced significantly by the weak link, an additional attempt is made to improve the weak link. For example, in the case of a subchain with 3 links having effectiveness' of 0.85, 0.75, and 0.3, the link having 0.3 effectiveness is really wasting the high effectiveness of the other 2 links. Note: If the weakest link is less than 75% of the average effectiveness, an attempt will always be made to improve the weak link. If the weak link was improved, the subchain optimization is considered successful and the subchain data is passed back to the calling subroutine.

(5) **Optimization of Crews.** Subroutine CRWOPT is the AURA crew optimization algorithm. Crew constructs can only be made of links. Each crew type (i.e. Tank, Gun, etc.) has a set of corresponding parameters which are described in the IDP report.¹⁸

CRWOPT attempts to optimize each crew member (link) by calling subroutine LNKOPT. To be considered improved, the crew must be improved to an effectiveness greater than the goal effectiveness. Subroutine LNKOPT processes the links in the crew in an attempt to meet (or exceed) the goal effectiveness. If a crew link can be improved in the crew's goal effectiveness, CRWOPT returns as a success and the improved effectiveness becomes the link's current effectiveness. Otherwise, the link is considered a weak link, a "choke point" that cannot be improved.

To find the overall effectiveness of the crew, subroutine CRWEFF is called. If the effectiveness of the crew has improved by .1%, the crew is considered successfully optimized. If the crew could not be improved, the weak links are placed in the weak link array (IWEAK) and program control is returned to the calling subroutine via the failure return.

(6) **Optimization of Links.** This subroutine constitutes AURA's link optimization algorithm. LNKOPT attempts to improve the effectiveness of a link via reassignment or reallocation of assets. The optimum effectiveness value attained by LNKOPT is returned to the calling subroutine as variable EFF.

Subroutine LNKOPT is called from the AURA construct optimization algorithms (CPLOPT, ORLOPT, etc.) to improve the effectiveness of the fundamental AURA construct type, the link. The link effectiveness to be improved is

18. op cit.

passed into LNKOPT as parameter EFF. The link optimization algorithm tries to improve EFF by determining the best (most effective) substitute which can perform the job of the weak link. (The pool of assets which are available (i.e. not currently assigned to a job) for substitution were stored in the array POOL by OPTMIZ).

The link optimization algorithm, also referred to as the AURA asset allocation algorithm, follows a set of decision rules which governs the process of how assets are assigned to links. These decision rules were outlined previously in section C.

The general processing sequence of the asset allocation algorithm within LNKOPT is described in the following steps:

- 1) Attempt to use the link's homelink asset. A homelink asset is an asset which has the same name as the link that it fills. A homelink asset is immediately available for its "homelink" task and contributes to the link's accomplishment at 100% effectiveness unless degraded otherwise.
- 2) If the homelink asset is not available, the link optimization algorithm will attempt to locate an asset which has been assigned to substitute in this link.
- 3) If there is no homelink asset or assigned substitute available for the link, the algorithm will try to borrow a substitute from another job.
- 4) If a substitute is not yet found, the link optimization algorithm will try to borrow a homelink asset from another job.
- 5) In the event there is no substitute available for link (JXX), the link optimization algorithm will determine if the link can be improved through repair of some available-but-damaged assets. LNKOPT does not, however, cause a repair to commence: it merely reports the existence of a repair option.
- 6) If a substitute is found that can improve the link effectiveness, LNKOPT assigns the substitute to job JXX and calculates the improved link effectiveness. Variable EFF is assigned the improved effectiveness and program control is returned to the calling subroutine.
- 7) If no improvement can be made to the link, JXX is the weak link. The link retains its original EFF value and is passed back to the calling subroutine. If the OPTIMIZE output option has been specified, informative diagnostics indicating the weak link number are printed

and LNKOPT returns to the calling subroutine.

A comprehensive description and derivation of link effectiveness is discussed in the following section.

(7) Calculation of Compound Link Effectiveness. Subroutine CPLEFF is called from subroutine OPTMIZ to calculate the effectiveness of a compound link. The compound link effectiveness is evaluated by determining the effectiveness of each of the compound link parts. Each compound link part has an associated user defined weighting factor which specifies the amount that the compound link part contributes to the compound link. The final effectiveness of the compound link is the weighted sum of all of the compound link parts.

CPLEFF determines the type of the compound link part and calls the appropriate subroutine to calculate it's effectiveness. Table 6 illustrates the subroutines called for each compound link part. CPLEFF then establishes the array IPROM to store the "promise" value of each part of the compound link. The promise vector is an array of the values indicating the ability of a compound link part to be filled. The promise value of a compound link part is defined as:

$(1 - \text{effectiveness of the part}) * \text{compound link part multiplier}.$

Therefore, the higher the effectiveness of the part, the lower the promise value.

Finally, array IPROM is sorted in descending order. This sorting allows subroutine OPTMIZ to sequentially access the compound link parts beginning with those with the greatest promise for improvement.

(8) Calculation of Orlink Effectiveness. Subroutine ORLEFF may be called by subroutines CPLEFF and OPTMIZ to calculate the effectiveness of an ORLINK. The ORLINK effectiveness is defined as the value of the strongest (most effective) orlink branch. The general process followed by ORLEFF is described as follows.

Subroutine ORLEFF determines the orlink branch which is currently being evaluated, the components which comprise the branch, and calculates the effectiveness of each branch component. The components of an ORLINK branch may be any combination of SUBCHAINS, CREWs, and LINKs. Table 6 illustrates the subroutines called to calculate the effectiveness for each ORLINK branch. ORLEFF processes all orlink branches and returns the effectiveness value of the strongest branch to the calling subroutine.

(9) **Calculation of Subchain Effectiveness.** Subroutine SBCEFF may be called by subroutines CPLEFF, OPTMIZ, or ORLEFF to calculate the effectiveness of a SUBCHAIN. A SUBCHAIN is similar to a CHAIN in that the effectiveness of the SUBCHAIN is only as strong as its weakest element. The elements of a SUBCHAIN may be any combination of CREWs and/or LINKs.

SBCEFF determines the number and type of components which comprise the SUBCHAIN. The effectiveness of each element is then calculated by a call to the appropriate subroutine (i.e. CRWEFF for CREWs and LNKEFF for LINKs). The effectiveness value for each SUBCHAIN element is returned and SBCEFF retains the value of the least effective element as the SUBCHAIN effectiveness.

(10) **Calculation of Crew Effectiveness.** Subroutine CRWEFF may be called by subroutines CPLEFF, OPTMIZ, ORLEFF, SBCEFF, and SBLOPT to evaluate the effectiveness of a CREW. The CREW effectiveness is based upon the user defined crew parameters and crew member effectiveness. All CREW members are LINKs.

The mathematical model for AURA's CREW was derived from a crew performance analysis¹⁹ by the Pacific-Sierra Research Corporation in conjunction with a study sponsored by the Defense Nuclear Agency.

In general terms, the AURA crew model is described by the following formula:

$$\text{Crew Effectiveness} = \frac{K}{\sum_{i=1}^n \left(\frac{a_i}{e_i} \right)^{b_i}}$$

where: n - number of crew members
 e_i - effectiveness of the i_{th} crew member
 and K , a_i , and b_i are constants (defined in reference cited)

19. op cit.

Subroutine CRWEFF computes the effectiveness of each crew member based on the above formula and returns the overall crew effectiveness value to the calling subroutine.

(11) Calculation of Link Effectiveness. Subroutine LNKEFF calculates a LINK's effective capability based upon the number and quality of assets currently assigned to the LINK. The LINK effectiveness is based upon the user defined LINK parameters specified in the runstream.

For each LINK, the user inputs such parameters as the number of assets needed for maximum link effectiveness (CAP100), the maximum effectiveness attainable by the link (CAPMAX), the assets which may substitute into the LINK (including their substitution time and effectiveness), the number of assets required for minimum link effectiveness (CAP0), and the minimum effectiveness attainable by the link (CAPMIN). [These parameters comprise the information necessary to evaluate the effective capability of the link based upon the link effectiveness curve shown in Figure 11.] Section IV.E.2 of the AURA input manual ²⁰ contains a complete description of all of the LINK input parameters.

Since LINKs are the fundamental part of all AURA constructs, LNKEFF may be called from any of the other construct effectiveness subroutines. The parameters passed to LNKEFF are: the link name (J), the variable CAP, (used to store and return the final link effectiveness), and the number of effective assets which are currently assigned to the link (REM).

All of the LINK parameters are then used to interpolate the effective capability of the LINK based upon the general link effectiveness curve illustrated in Figure 11. Note: A general description and derivation of the link effectiveness curve was described in Section 5.a.

Subroutine LNKSET, which sets the default parameters for each link, establishes the link capability factors for each link based upon the user inputs. The link capability factors are defined in Figure 12. The general processing sequence and calculation of link effectiveness is described as follows:

- 1) If the number of effective assets currently in the link (REM) is greater than the number of assets needed for maximum link effectiveness (CAP100), the capability of the link is assigned to its maximum effectiveness (CAPMAX).

20. op cit.

SUBTASK EFFECTIVENESS

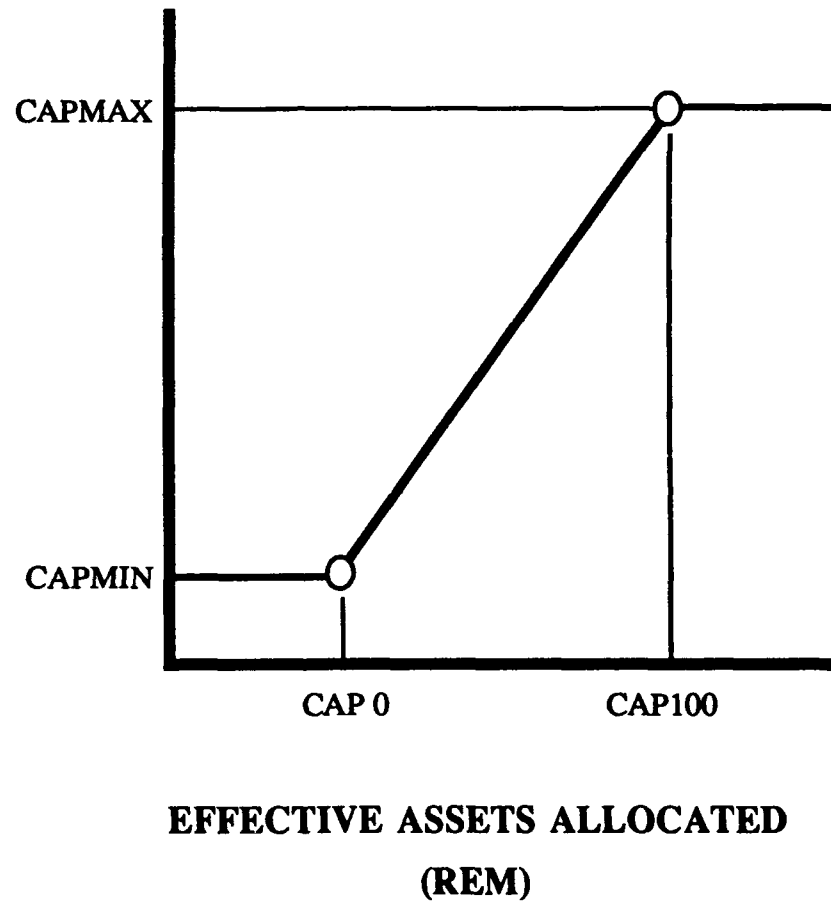


**Figure 11. General Form of a Link Effectiveness Curve
Illustrating the Link Parameters**

- 2) If the number of effective assets currently in the link (REM) is less than the number of assets needed for minimum link effectiveness (CAP0), the capability of the link is assigned to its minimum effectiveness (CAPMIN).
- 3) Otherwise, the link's capability is based upon the general link effectiveness curve. The link effectiveness calculation is derived from the formula for the link effectiveness calculation as illustrated in Figure 12.

Finally, LNKEFF returns the new link capability value (CAP) to the calling subroutine. Figure 13 describes the effectiveness derivation and description for all of the AURA constructs.

d. Application of the Asset Allocation Algorithm to the Hypothetical Case. It is helpful to apply the methodology of the Asset Allocation Algorithm to the ASP, our hypothetical supply unit. A possible mission of the ASP unit is to load trucks on order at a certain ratio. Two weight classes of items, heavy and light, are to be loaded. The heavy items, which comprise 25 percent of each load, must be loaded with a crane; the light items can be loaded by hand or by forklift. The order to fill the trucks is received by radio or telephone. Personnel are required to receive the order, man the forklift and crane teams, drive the truck, and handload if required. Handloading, however, can never accomplish more than 80 percent of the required rate, and requires more than one person. There is also a loadmaster, who supervises the operation. However, the unit has functioned together long enough to work at 60 percent of the required rate even if the loadmaster's job is not done.

The links required (and their associated effectiveness curves) to describe this unit are shown in Figure 14. Note that most jobs in this simple example are of the form (1., 100; 0., 0). (e.g. 1 asset for 100 percent effectiveness; 0 assets for 0 percent effectiveness). Exceptions to this are the LOADMASTER (1., 100; 0., 60) and the handloading MEN (5., 80; 1., 0).

For this example, only one mission was given. As described above, the mission requires receipt of the message, a radio or telephone, 0.75 light and 0.25 heavy load capability and a truck. The LOADMASTER is also ANDed into the CHAIN. However, referring to Figure 14, one notes that the absence of a loadmaster asset reduces the value of the LOADMASTER LINK only down to 0.6, thus limiting his effect on the unit.

Figure 15 graphically depicts the segments (representing subtasks) which comprise the jobs required to perform the mission. These subtasks are then combined to form the CHAIN representing the overall mission. Included in this figure are the naming conventions used for each segment as described in the AURA Input Manual.²¹ In short, the naming conventions require that the segment name

The LINK effectiveness calculation is based upon the user defined LINK parameters and is described by the following formula(s).

$$\text{Link Capability} = \text{Link Capability Factor \#1} + (\text{REM} \times \text{Link Capability Factor \#2})$$

where:

$$\text{Link Capability Factor \#1} = \text{CAPMIN}(J) - (\text{CAP0}(J) \times \text{Link Capability Factor \#2})$$

$$\text{Link Capability Factor \#2} = \frac{\text{CAPMAX}(J) - \text{CAPMIN}(J)}{\text{CAP100}(J) - \text{CAP0}(J)}$$

J - Link identifier

REM - Number of effective assets currently assigned to LINK(J).

NOTE:

If REM $\geq$ CAP100, the link capability is assigned as CAPMAX.

If REM $\leq$ CAP0, the link capability is assigned as CAPMIN.

(CAP0 < REM < CAP100)

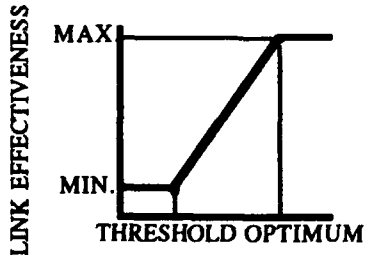
CAPMAX - Maximum effectiveness attainable by LINK(J)

CAPMIN - Minimum effectiveness attainable by LINK(J)

CAP100 - Number of assets required for maximum effectiveness

CAP0 - Number of assets required for minimum effectiveness

Figure 12. Link Effectiveness Formulation

Structure	Effectiveness Derivation	Description
Mission	$\text{Eff.} = \max \{ \text{chain1}, \text{chain2}, \dots \}$	As strong as strongest chain.
Chain	$\text{Eff.} = \min \{ e_1, e_2, \dots \}$	As strong as weakest segment.
Compound Link	$\text{Eff.} = \sum_{i=1}^n f_i e_i ; \quad \left(\sum_{i=1}^n f_i = 1 \right)$	As strong as weighted sum of parts. (f_i are the weighting factors)
Orlink	$\text{Eff.} = \max \{ e_1, e_2, \dots \}$	As strong as strongest branch.
Subchain	$\text{Eff.} = \min \{ e_1, e_2, \dots \}$	As strong as weakest element.
Crew	$\text{Eff.} = \frac{K}{\sum_{i=1}^n \left(\frac{a_i}{e_i} \right)^{b_i}}$ where: n - number of crew members e_i - effectiveness of the i^{th} crew member and K, a_i, and b_i are constants	Strength reflects the ability of the crew members to 'cover' for the weakest member. However, the crew cannot compensate for the total loss of a member.
Link	$\text{Eff.} =$ 	Fundamental 'building block'. Effectiveness depends upon number and quality of assets assigned.

Components of any structure may be any combination of lower lying structures

Figure 13 Effectiveness Derivations of AURA Constructs.

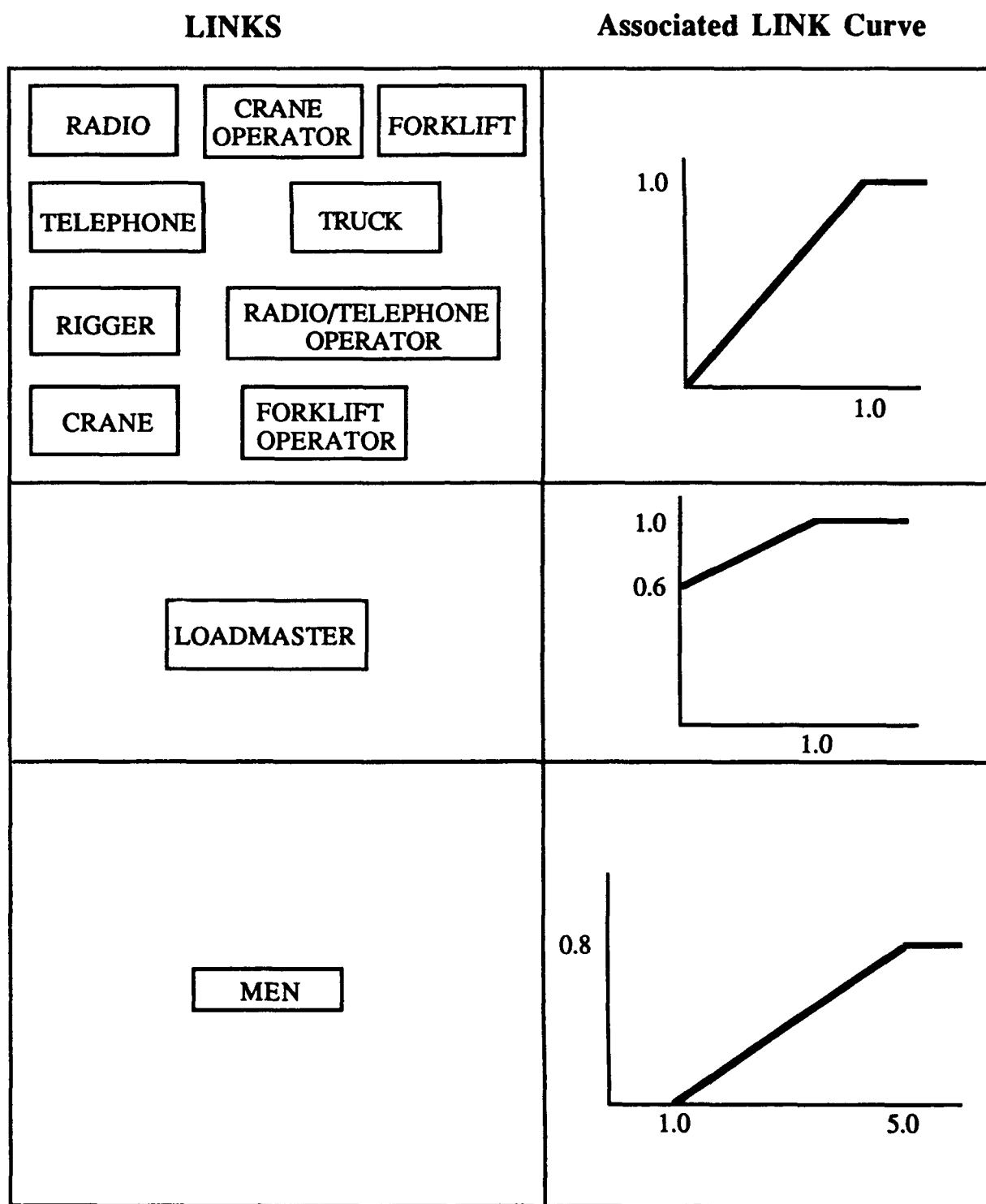


Figure 14. Links and Associated Link Effectiveness Curves for Unit Assets

Segment Type	Task Represented	Graphical Depiction
CREW	=Crane Crew	
SUBCHAIN	*Forklift Team	
SUBCHAIN	*Crane Team	
ORLINK	+Communication Device	
ORLINK	+Light Load	
COMPOUND LINK	!Loading Technique	

Figure 15 Segments Representing the Tasks of Example Unit

for CREW, SUBCHAIN, ORLINK, and COMPOUND LINK begin with =, *, +, and !, respectively. Figure 16 illustrates the complete combination of segments (CHAIN) used to represent the mission of the example unit.

CHAIN Representing Unit Mission

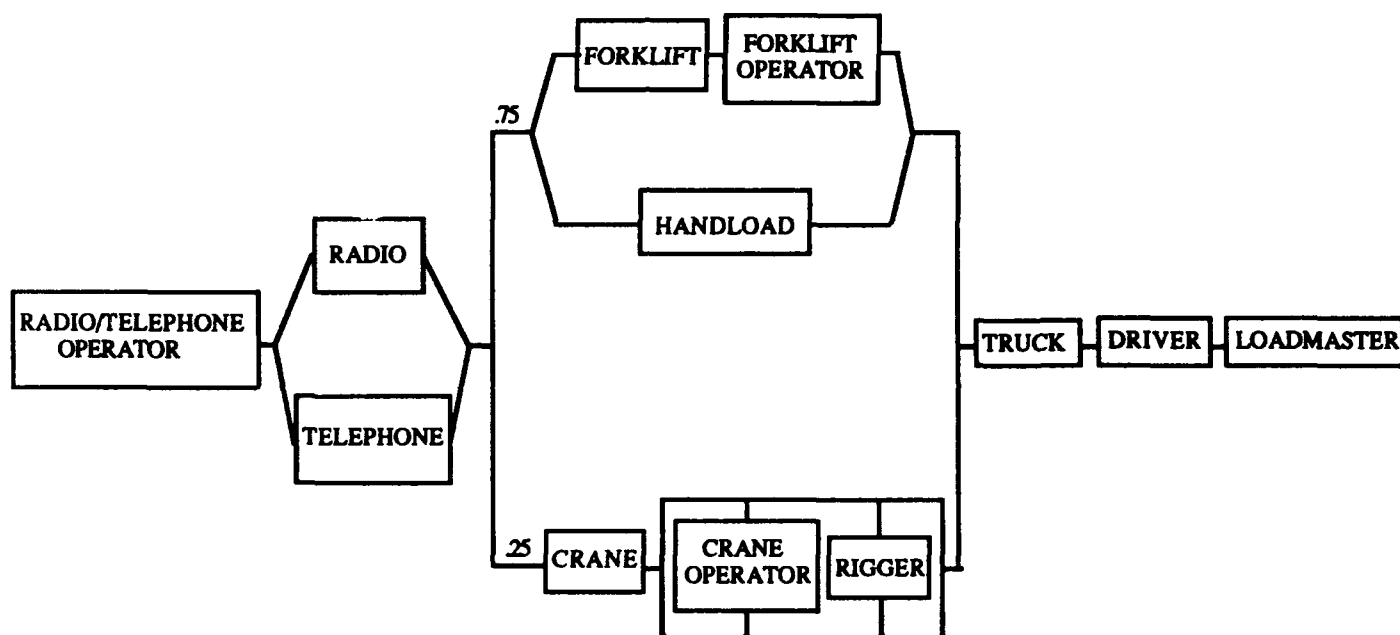


Figure 16. Chain Structure for the Example Unit

When augmented by a decision rule, the figure also depicts the value function upon which the AURA optimization process is based. That decision rule is:

The EFFECTIVENESS of a "CHAIN" is EQUAL to the EFFECTIVENESS of the WEAKEST SEGMENT.

When applied to the example depicted in Figure 16, this decision rule implies that a unit which had only one-third of its trucks would only fulfill one-third of its mission, as long as all other capabilities were greater than one-third. In particular, even if the ability to receive messages was degraded to one-half the prescribed rate, the unit would only be able to load the available one-third prescribed number of trucks.

2. The AURA Repair/Return Methodology

a. **Repairs.** AURA contains a detailed and complex model to simulate the failure of equipment and the repair/return of damaged equipment. Underlying this model are certain factors which lay the groundwork for the repair/return algorithm. They are:

- Repair capability is needed only if repairably damaged items are present;
- Repair activity may compete with other activities for unit assets;
- A commander may decide not to repair except on an "as-available" basis;
- The decision to use otherwise needed assets in repair roles depends on the criticality of the item being repaired;
- Repair completions are time distributed; i.e., a 2 hour repair may take more or less than 2 hours, with varying probabilities;
- Items being repaired can be further damaged by subsequent incoming fire. Similarly, personnel and equipment performing the repair can become casualties;
- Repair activity ceases during incoming fire;
- An ongoing repair job can be pre-empted by the need to repair a higher priority item.

AURA treats a repair as an unneeded capability until ordered. At that point, the repairing personnel and equipment are redeployed to the repair location, with transit and "get-up-to-speed" time accounted, and repair commences.

b. Ordering and Completion of Repairs. Repairs can be ordered in two ways. First, the AURA optimization routine keys on the weakest link (limiting capability). When the limiting capability is vested in an item which is repairably damaged, the routine attempts to fix the item. If such a fix can be made without detracting from the residual capability of the unit to do its mission (e.g., if the repair can be done by non-mission-essential elements), the repair will be ordered in that way. If, on the other hand, an additional decrement in unit performance must be taken in order to commence the repair of a limiting item, the decrement is compared against the item's user-selected penalty parameter (discussed in the next section). In this way, a user can choose to accept a temporary penalty in order to realize future gain, as an actual commander would do.

The second way to order repairs is on an "as-available" basis. After the mission (or repair-augmented mission) requirements have been allocated, the commander assigns any remaining repair assets to do any repairs that can be done. These repairs are considered in numerical order by asset ID number; however, the top priority repair level is considered for every asset before considering lower levels.

Repairs can be specified on three levels: 0 = contaminated; 1 = light repair needed; and 2 = medium repair needed. It is assumed that light or medium repair can be applied to the specified item regardless of the source of need, i.e. either failure or combat damage. NOTE: repairable combat damage requires both that the item appear as a repairable under the REPAIR mnemonic and that the item has exactly three kill criteria specified in the conventional lethality file. Repair of contaminated items is specified in the same way as those of level 1 and 2. It is assumed that the unit has the solutions and equipment required to effect complete decontamination unless the required solutions/equipment are specifically listed under the "EXPENDABLE" option (see next section).

The completion of a repair is treated probabilistically. It is assumed that actual repair times will be distributed normally, with user-input means and standard deviations for both light and medium damage. This implies an increasing probability that the repair will be completed. AURA implements the normal distribution at each update of the repair clock. Elapsed time is fitted against the cumulative normal curve, and corresponding probabilistic increments are credited to the repair. Upper end cut-offs are implemented to preclude dedication of assets to vanishing returns.

Repair capability is treated as a link through normal link input routines. Links needed for a given repair are input through the standard mnemonic input processor. The different levels of damage (light, medium, or heavy (irreparable)) are treated as different kill criteria in the standard conventional lethality files.

c. **Repair Mnemonics.** The "REPAIR" mnemonic inputs data relative to the repair of damaged or failed equipment. Included in the inputs are the sub-tasks (LINKs or SUBCHAINs) needed for repair, the location of the repair, the mean time and standard deviation in the mean time for repair, and the penalty (0. - 1.) that the commander would be willing to take in his immediate mission in order to fix the item if the need for the item were the choke point in the mission. AURA also uses the penalty value to help prioritize possible repairs at various levels. Repairs are input as follows:

REPAIR

asset name

\$cp0, lvl0, pnlt0, mt0, sd0, xrp0, yrp0

\$cp1, lvl1, pnlt1, mt1, sd1, xrp1, yrp1

\$cp2, lvl2, pnlt2, mt2, sd2, xrp2, yrp2

where cpI is the LINK or SUBCHAIN needed to perform level I repair on the named asset;
lvlI is the level of repair being described;
pnltI is the acceptable mission penalty for level I repair;
mtI is the mean time to accomplish level I repair;
sdI is the standard deviation in mtI;
xrpI, yrpI are the coordinates of the location at which repair of this item at this level would take place.

One, two, or three levels may be specified.

d. **Other Repair Related Mnemonics.** A number of repair related options exist which permit the modeling of various repair phenomena. These will be discussed briefly in this section.

(1) **Failures.** This option specifies the rate at which assets undergo spontaneous (reliability-type) failures. Equipment can fail so as to require light, medium, or irreparable repairs. AURA handles failures exponentially. That is, the user specifies the mean time between failures (MTBF) for each item that can fail. At each event time, a Monte Carlo technique is used to identify specific failures, where each item's failure probability is a function of the ratio of elapsed time increment to the item's MTBF. Failures are input in the following manner:

FAILURE RATE

asset name, mtbf, fl, fm

where:

mtbf is the mean time between failures;

fl is the fraction of failures requiring light repair;

fm is the fraction requiring medium repair.

If fl and fm are not specified, the failures are assumed to be dead (irreparable) failures.

A related option is the PREFAIL option which allows some light and medium repairs to be prestarted, simulating an ongoing process at the initial time of the run. This option avoids having too many items available at time zero, too many failing afterwards, and a delay in the repair return rate.

(2) **Expendables.** AURA contains an option which allows the user to identify assets that are expended as they are used. Items may be expended in two ways: by time and by repair completion. For those items expendable by time, the amount that is assessed as expended at any time point depends upon the amount of mission time spent since the previous update and the effectiveness of the unit during that time. If an item is described as expendable during repair or decontamination activities, it must have HOMELINKS which appear in the consuming repair SUBCHAINS. Expendables are input in the following manner:

EXPENDABLE

asset name, rate of usage by time

or

EXPENDABLE

\$asset name

where the latter form is used for items expended during repair.

(3) **General Repair.** The General Repair option allows the specification of general repair links or subchains. These links or subchains represent capabilities which must be satisfied only once, regardless of the number of repairs ongoing. An example may be a supervisor or a spare parts van. This option is specified in the runstream as follows:

GENERAL REPAIR

\$gnrl cp, lvl

where:

gnrl cp is the LINK or SUBCHAIN needed for any repair at level lvl;
lvl is the level for which gnrl cp applies.

(4) **Maximum Number.** To limit the number of repairs that may be ongoing at any time, the MAXIMUM NUMBER option is used. This option specifies the maximum number of repairs which may be ongoing simultaneously. The default value is 50 repairs. To change the default, the new maximum value is input as:

MAXIMUM NUMBER, maxrp

where maxrp is the maximum number of repairs that can be ongoing at any one time.

(5) **No Repair.** To specify those chains in which no repair or decontamination is allowed, the NO REPAIR option is used. To use this option, the CHAIN input must precede the REPAIR input in the runstream. This option is modeled as follows:

NO REPAIR, nch1, nch2, nch3,....

where nchI are the CHAINS which do not allow repair or decontamination.

e. **The Repair Subroutines.** A number of subroutines exist within AURA which model the playing of repairs and failures. Each will be described, in detail, in this section.

(1) **Determination of Reliability-Type Failures.** Subroutine FAIL is called by EVNTDO to identify reliability-type failures by target point. Reliability failures may occur at three levels of repairability: light, medium, or dead failures. Although repairability levels may relate to any user determined definition, light normally refers to crew level repair, medium to organizational level repair, and dead to no repair, no return to the unit.

This subroutine analyzes probability of failure for each asset type by target point. An item deployed at a target point may either be an asset or a link. Target points that are set up for assets but not jobs are termed "NOLNK" target points. FAIL keeps a list for each asset of all links into which it may substitute. This list is an array whose number of positions equals the number of links into which the asset may fill. Probability of failure is determined for asset type, and then analyzed by target points. When the code encounters a NOLNK target point that contains the asset currently being analyzed, a random number draw is performed and it is determined whether or not the item failed. When the code encounters a target point containing a link, it first searches the array list for the current asset type to see if it can fill into the job. If it can and it is currently filling the job, a random number draw is performed which determines the failure or survival of the asset. FAIL moves on to the next target point if the asset is not currently filling the job.

After determining the probability of failure for a specific asset type and then locating the asset type by target point, FAIL calls the uniform random number generator to determine if the failure actually occurred. If it has, FAIL

determines at what level it failed, i.e. light, medium, or dead. Items which are contaminated with chemical agent but still being used are also considered for reliability-type failures. Once failed, the item must be subtracted from the lists of usable assets and, if contaminated, removed from the evaporation returners so that it will not be returned to duty once the chemical agent has evaporated.

Once all asset types and target points have been analyzed, FAIL resets the variable TFAIL to the current event time. This will be checked at the beginning of FAIL the next time it is called. (TFAIL is used to calculate DELT which is the elapsed time since the last failure update.)

(2) Verification of Repair/Failure Inputs. Subroutine FAICHK is called by ENCCHK to check for correct reliability failure modeling formats. It also sets DELT0, the time used to determine failures and repairs before the onset of the scenario, when the PREFAIL option is turned on. DELT0 is also used in subroutine FAIL to determine the expected number of assets at the start of the scenario.

FAICHK first checks for items that have been modeled to fail at a repairable level but are not modeled for repair. If it finds such an asset, FAICHK will print a warning message that repair will not be performed on this particular asset.

FAICHK next checks for items that have been modeled to "prefail". Recall, prefail assumes that these items have been around for some time and that there is a probability that some of them may have failed and are not available for use at the start of the scenario. (When prefail is set, unit effectiveness can be degraded at the beginning of the scenario due to these reliability failures.) FAICHK checks for items that can fail but are not repairable and prints a warning message to the output file. This check is made to prevent large numbers of unrepairable assets from failing before the onset of the scenario.

The variable DELT0 is also calculated by FAICHK. DELT0 is the time before the onset of the scenario for which failures and repairs are being calculated. It is subsequently used in subroutine FAIL to determine the expected number of assets available at the start of the scenario. One would expect, given enough time, that the number of usable items would level off; that is, as it approaches infinity, $N(t) = (NT) \frac{(R)}{(R+F)}$ where NT is the number of total items to start with, R is the repair rate and F is the failure rate. Given the probability of survival, Ps, and the mean time between failures (MTBF), one may solve for DELT0 using the following relationship:

$$P_s = \frac{(NT)(R)}{R+F} = 2 \frac{-DELT0}{MTBF}$$

$$\frac{DELTO}{2^{MTBF}} = \frac{R}{R+F}$$

$$DELTO = (-MTBF) (\log_2 \frac{R}{R+F}).$$

$$\log_2 \frac{R}{R+F} = (1.4427) (\ln \frac{R}{R+F})$$

$$DELTO = (-1.4427)(MTBF) (\ln \frac{R}{R+F}).$$

A weighted average repair time is actually calculated and used in the above algorithm for repair rate. This average repair time (AVETIM) is calculated using the mean time to perform repairs of light level failures (TIMMU (I,1)), the percent of light failures to repair (PCTLIT) and the mean time to perform repairs of medium level failures (TIMMU(I,2)) and the percent of medium level failures (PCTMED):

$$AVETIM = (TIMMU(I,1) (PCTLIT(I)) + \frac{(TIMMU(I,2)) (PCTMED(I))}{PCTLIT(I) + PCTMED(I)})$$

Finally, FAICHK checks the failure times against the event times. If the MTBF is small compared to the maximum time between reconstitution events, too many failures could occur and sit idle before repairs are processed. To prevent this type of error, the code prints a warning message suggesting the use of additional reconstitution times.

(3) Calculation of Reliability Type Failures. Subroutine PREFL can be called by either REPSET or RPORDR, with two calling parameters: IFG, functional group or asset and EUP, number of assets at the target point. PREFL also contains one entry, PREFLR.

(4) Main PREFL Algorithm. PREFL calculates the probability of failure before the onset of the scenario. It determines what type of failure has occurred, that is, light, medium, or dead failure. (Typically, light failures are considered crew level repair items and medium failures are organizational level repair items.) Repair is modeled for light and medium failures, but not for dead failures, which are considered irreparable for the purposes of AURA modeling. This subroutine allows a scenario to start at less than 100 percent asset availability. Particularly if reliability failures are being modeled, one would assume that unit equipment has already been around awhile before the start of the scenario and that some equipment might have failed and be in need of repairs. Similarly, one would expect that if failures had already occurred that repairs would have been initiated.

By definition of MTBF, the probability of failure, Pf, is modeled such that the probability of survival, Ps, is cut in half at each mean time between failure (MTBF) interval. That is:

$$P_s = 2^{\frac{-\text{time}}{MTBF}}$$

where time is time over which failures are being analyzed
before onset of the scenario;
MTBF is mean time between failures.

Time over which failures are calculated before the onset of the scenario is called DELT0. Pf, then, is set to $(1 - P_s)$ which equals $(1 - 2^{\frac{-DELTO}{MTBF}})$. One may check that as the time over which failures are calculated increases, Pf approaches 1. To compensate for the fact that repairs are also ongoing before the onset of the scenario, repair rate is also addressed.

Subroutine PREFL initially sets Pf for the random preset option (indicated by MTBF input as negative). Recall, if MTBF is negative then the "PREFAIL,ON" option must also be used. Using a negative MTBF sets $P_f = -DELTO$ and indicates that the value of MTBF is taken to be the probability of the item not being present at time 0. Once PREFL checks for this special case, it continues with a calculation of the probability of failures. Pf is calculated as described above, that is:

$$P_f = 1 - 2^{\frac{-DELTO}{MTBF}}$$

Failures are input in terms of a single MTBF. Additionally, percent light and percent medium failures may also be specified. If not, all failures are considered to be "dead" failures (not repairable). Percent of light failures (PLITE) is calculated as:

$$PLITE = \frac{PCTLIT(IFG)}{PCTLIT(IFG) + PCTMED(IFG)}$$

where PCTLIT is the percent of light failures for
functional group IFG;
PCTMED is the percent of medium failures for
functional group IFG.

Probability of failure is calculated item by item for each target point. A uniform random number generator is used to Monte Carlo failures. After calculating the probability of failure, a random number draw is made. If the random number (RN) is less than or equal to the probability of failure, the failure occurs; if RN is greater than Pf, failure does not occur. Once a failure occurs, it is subtracted

from the number of assets available at that target point. Random number draws are made for each item at the target point.

If a failure has occurred, PREFL checks if it was a light, medium or dead failure. If it was a light or medium, this item is added to the array of equipment which need fixing (FIXJNK).

(a) **Calculation of Prefailed Assets.** This entry is called for RPORDR if the PREFAIL option is on and if the number of internal reconstitution times is one. Given that prefail is on, repairs must be made before the onset of the scenario to avoid having too many failed items at the beginning of the scenario. This entry randomizes progress that is accomplished on ongoing repairs. First, the time to complete the repair is found in terms of the mean and the standard deviation; these values are user inputs. The time to complete repair is determined to be either two times the mean time to repair or two times the standard deviation plus the mean, whichever is smaller.

For repairs that are described with a large standard deviation, this calculation will arrive at a completion time of two times the mean. For repairs that are described with a small standard deviation (that is, a repair where the mean is greater than two times the standard deviation), this calculation will arrive at a completion time of the mean plus two times the standard deviation.

After the completion time is determined, a uniform random number draw is made to determine the actual time spent making the repair (TMSPNT):

$$TMSPNT(IJ) = (CC)(RN(1))$$

where RN(1) is the random number.

(5) **Removal of Jobs from Repair Shop.** RMVRPR is called from RPRAVL, RPRLTH, or RPRRTN to remove repair jobs from the shop and place them in the junkyard. The parameters JMRV and REASON are passed into this subroutine. JMRV indexes the array ITBRPR which means that it is used to number the items to be repaired; REASON is used to define the reason why items are being removed from the repair shop. The term "junkyard" is used to refer to the list of damaged equipment that is neither being used by the unit nor repaired. Specifically, it is used to refer to the array, FIXJNK. FIXJNK is a list of assets which cannot be used until repaired or decontaminated.

RMVRPR is called from RPRLTH if an item currently in the shop has undergone further damage such that it is no longer repairable. It is called from RPRAVL in order to clear out repairs from the repair shop if there are no available assets to perform the repair. It is called from RPRRTN in order to resequence repair activities from the highest level of order to the lowest.

(6) Initiation of New Repairs and Calculation of Repair Effectiveness. Subroutine RPORDR can be called by either RPRAVL or OPTMIZ to initiate new repairs, restart old ones and calculate repair effectiveness for new or ongoing jobs. This subroutine is called with the following parameters: IFG is the asset number to be repaired; NL is the level of damage; * indicates an alternate return; ITRB flags needed repairs; RPREFF is the repair effectiveness; and, AVLRP is the amount of asset available for repair.

RPORDR has three sections with different purposes. The main routine determines the status of the job. Entry RPRORD deals with new repair jobs while entry REORDR calculates repair effectiveness for either new or ongoing repair jobs.

RPORDR skips to entry REORDR if the job is ongoing and the effectiveness has not been calculated. If the job is new or if the ongoing job has already been staffed and a new job is to be started, RPORDR goes to entry RPRORD. RPRORD assigns values such as the job number, number of repair jobs, mean time for repair, standard deviation for repair, the X and Y coordinates for repair location, number of repairs requested, and the amount of asset available to repair. If the repair level indicates the equipment is contaminated, a call to entry DCRES0 removes the asset from the "clean by evaporation" group to avoid returning the asset by way of both repair and evaporation. If the PRE-FAIL option is specified, subroutine PREFL is called to process the prefailure equipment. Once the job is initialized, entry REORDR will calculate the new job repair effectiveness.

(7) Ordering of Repairs. Subroutine RPRAVL is called by RECEVN to fill any additional repair links with available assets. These additional repair links do not compete with other unit links for unit assets. They are filled only by non-essential (leftover) assets.

The following factors form the basis of the repair methodology for non-competing repairs and are modeled by RPRAVL and those subroutines which it calls:

- Repair capability is needed only if repairably damaged items are present, i.e., repair links are filled only if needed;
- Repair completions are time distributed;
- Certain personnel or items of equipment may be required for the supervision or initiation of a repair activity. If required, these assets are described using the "general repair links".

- Repairs may be performed by several teams of personnel, i.e., the repair links may be filled as many times as allowed by the maximum number of repair personnel specified;
- Repair parts may be expended proportionally to the number of repairs being completed.

In the completion of repair on an as-available basis, the first requirement is to satisfy the general repair link, if any. Following this, the specific repair links are filled; in this subroutine, repair links may be filled more than once.

Repair tasks may be described such that they consist of only a simple link or a more complex subchain of links. The repair process may also be described such that a general repair link needs to be satisfied before the specific links can be filled. If the general repair link is not filled, then the specific repair links will not be filled. The general repair link may be used to model the essential need for a repair shop or supervisor, for example.

RPRAVL first determines the number of damaged assets needing repair, the type of repair required, and the links needed for the repair. If there are no damaged items, the subroutine returns to its calling routine. If there are damaged items, RPRAVL checks to see if there are enough repair assets to fill the repair link and then searches for the appropriate substitutes.

For the modeling of as-available repairs, a link may be used more than once. For example, assume a repair link requires two assets for 100 percent link effectiveness and five assets are available. AURA will not put all five assets in one link but rather set up three links, two of which are completely filled and operating at 100 percent and one which is only half-filled and, therefore, operating at 50 percent.

When links cannot be filled with their homelink asset, RPRAVL searches the list of substitutes by their sequentially increasing versatility numbers (see Asset Allocation Algorithm section for discussion of versatility); the least versatile asset will be used first. Once the least versatile asset is found, then substitution time is analyzed. If this asset cannot fill the link before the next event, then RPRAVL searches for another asset to fill the link. There is an internal limit of five asset types which may work on any one repair. If more than five asset types are available to fill into the repair link, RPRAVL will look for the most effective asset types to complete the repair link.

Once the repair links are filled, subroutine LNKEFF is called to determine repair link effectiveness. After this is done, the repair will be started through a call to RPRORD if it is a new repair or REORDR if this is a repair that has been previously started. These subroutines track what is currently in the repair shop. If the repair shop is full at the time RPRAVL orders a repair, the repair cannot

be made. Finally, a call to RMVRPR will clear out any repair jobs from the shop which are not staffed.

Note, assets to perform repairs are drawn from a pool of survivors that were not used by subroutine OPTMIZ. After RPRAVL has assigned all the assets needed to perform repairs, it places the remaining personnel (in the survivor pool) into an array that keeps track of them for purposes of fatigue modeling.

(8) Determination of Further Damage to Ongoing Repairs. RPRLTH analyzes the further damage to items of equipment which are currently being repaired. Damage may be the result of either a conventional or nuclear attack; chemical "damage" is not considered here since it is primarily contamination.

This subroutine first determines whether or not the incoming round is within range of the repair shop (the location where repairs are taking place). If it is, the asset number and posture of the item to be repaired are determined as well as its X and Y location. RPRLTH then calls either the CNVDMG subroutine, if it is a conventional round, or NUCDMG, if it is a nuclear round, to determine the item's probability of kill (see the sections on Conventional Modeling or Nuclear Vulnerability Modeling for discussions of these subroutines).

If the incoming round is conventional, RPRLTH also determines whether there is one or three levels of combat damage. If three levels exist (light, medium, and non-repairable), three different PKs are determined for each level. There is only one level of kill for nuclear damage; therefore, the three PKs are set to the same PK.

After the PK is determined, RPRLTH determines what level of damage has occurred. If total kill occurred, the item is no longer repairable and is removed from the shop. If medium damage has occurred, the item is placed into the junkyard for medium repair. If light damage occurred, light repair parameters are reset and the repair is restarted.

(9) Return Repaired Assets to Survivor Pool. RPRRTN is called by EVNTDO at each event time so that repairs of failed, damaged, or contaminated equipment may be updated. The number of repairs completed is determined based on the amount of time which has elapsed since the last update and the user specified repair inputs such as mean time for repair of the item, the associated standard deviation, and the code-determined repair effectiveness. Repair effectiveness is used to modify the rate of repair over the elapsed time. It is also possible to model a repair such that each time a repair is completed, a certain amount of a specified asset is expended. This is the "EXPENDABLE" option and it identifies the assets that are expended as repairs are made and the rate of expenditure. RPRRTN takes care of reducing the available assets for expenditure as a function of repair completion. If the repair was decontamination, RPRRTN

reevaluates the time at which personnel may reduce their level of mission oriented protective posture (MOPP).

Repairs are modeled deterministically. At each event, RPRRTN determines the amount of time that has been spent making the repair and the fraction of repairs (a normalized value) that have been accomplished. It then determines the total number of repairs made and returns these items to the deployment and puts them back into the proper links. As repairs are made, if items are expended, the subroutine subtracts them from the list of available assets.

Repairs are calculated to occur based on a normal distribution. Repairs are cumulative and since a cumulative normal curve never quite reaches 1.0, where 1.0 indicates the repair is completed, the code determines a cut-off point at which it returns the item to duty. The cut-off is determined to be the mean time to repair plus the smallest of either the mean or two times the standard deviation. This value is then compared to the amount of time spent on the repair so far; if the amount of time so far is greater than the cut-off, the repair is completed.

RPRRTN next calls subroutine CUMNRN to update repairs for the current time period. Once the number of completed repairs is calculated, RPRRTN adds these items back to the running total of assets and records the repairs made. The return to duty is completed by replacing the repaired items in the appropriate deployment points; similarly, expendable items are subtracted from the deployment points.

(10) Determination of Repairable Assets and Repair Priority.

This subroutine, called by ENCSET, clears repair arrays that keep track of repair requests and repairs recorded. RPRSET prioritizes a list of repair levels in order of increasing penalty value (the penalty a unit is willing to accept to unit effectiveness in order to repair the asset) for each asset. The repair levels are: 0 = decontamination; 1 = light; and, 2 = medium. If the penalty is -1, there is no penalty data (the penalty ranges from 0 to 1). The list of penalties is packed into a word, for each asset, in array KRP and used for later processing.

f. Application of Repair to the Hypothetical Case. The repair algorithm was used in the original Ammunition Supply Point (ASP) study and its use in that study will be discussed here.²² The repair option was used in regard to certain, mission essential pieces of equipment contained in the ASP, namely, fork-lifts, cranes, drop-sided trucks, tractors, and a wrecker.

22. Stark, M.M., Klopac, J.T., "The Resiliency of an Ammunition Supply Point to Combat Damage(U)", USA Ballistic Research Laboratory Technical Report BRL-TR-2614, December 1984, (SECRET).

Lethality data was determined for each piece of equipment and input in terms of light, medium, and heavy damage. In this analysis, heavy damage was defined as damage requiring at least eight hours of repair. It was assumed any piece of equipment requiring this much time to repair would require repair beyond the unit's capability and thus would be lost to the ASP. Medium damage required a minimum of 90 minutes to repair, while light damage required a minimum of 30 minutes. These times were used as a basis for estimating the mean time for repair and the corresponding standard deviation.

The ASP study made use of both the repair links and the general repair links options. It was assumed that the maintenance leader and the auto technician were needed to supervise or provide guidance for any ordered repair. These personnel, along with the necessary tractors and trailers needed to transport the damaged equipment, were considered general repair links. Also needed for any specific repair were the corresponding repair links, known in this study as repair teams. Each team, consisting of three personnel, was assigned to one damaged item at a time.

3. The Fatigue/Sleep Deprivation Model

a. **Overview of the Fatigue Model.** The AURA code attempts to quantify the state of restedness and the benefits of sleep for unit assets. To do this, a unit of sleep, called the SLUNIT, was invented. Although the SLUNIT is an abstract concept whose value can be adjusted to best fit the existing data, a SLUNIT can be roughly equated to the recuperative value of one minute of efficient sleep. Thus, accumulated SLUNITs are a measure of an individual's restedness. SLUNITs can be accumulated by an individual up to a maximum (CEILING), which corresponds to being fully rested. That maximum may, in fact, be a highly individual quantity; AURA, therefore, provides for user inputs for CEILING values for each individual in the simulation. As a default value, AURA currently uses a CEILING of 1520 SLUNITs for everyone. Figure 17 depicts the SLUNIT accumulation curve. In the interest of brevity, the reader is referred to a BRL report for more information on both the fatigue model and the derivation of the default CEILING value.²³

(1) **SLUNIT Expenditure.** SLUNITs are expended by activity. The rate of SLUNIT use, therefore, is job dependent. Studies have consistently shown that cognitive jobs are more tiring than rote, physical jobs. Scientists at the Walter Reed Army Institute for Research (WRAIR) have quantified the loss of

23. Klopacic, J.T., "The AURA Fatigue and Heat Stress Algorithms", USA Ballistic Research Laboratory Memorandum Report No. BRL-MR-3802, December 1989, UNCLASSIFIED.

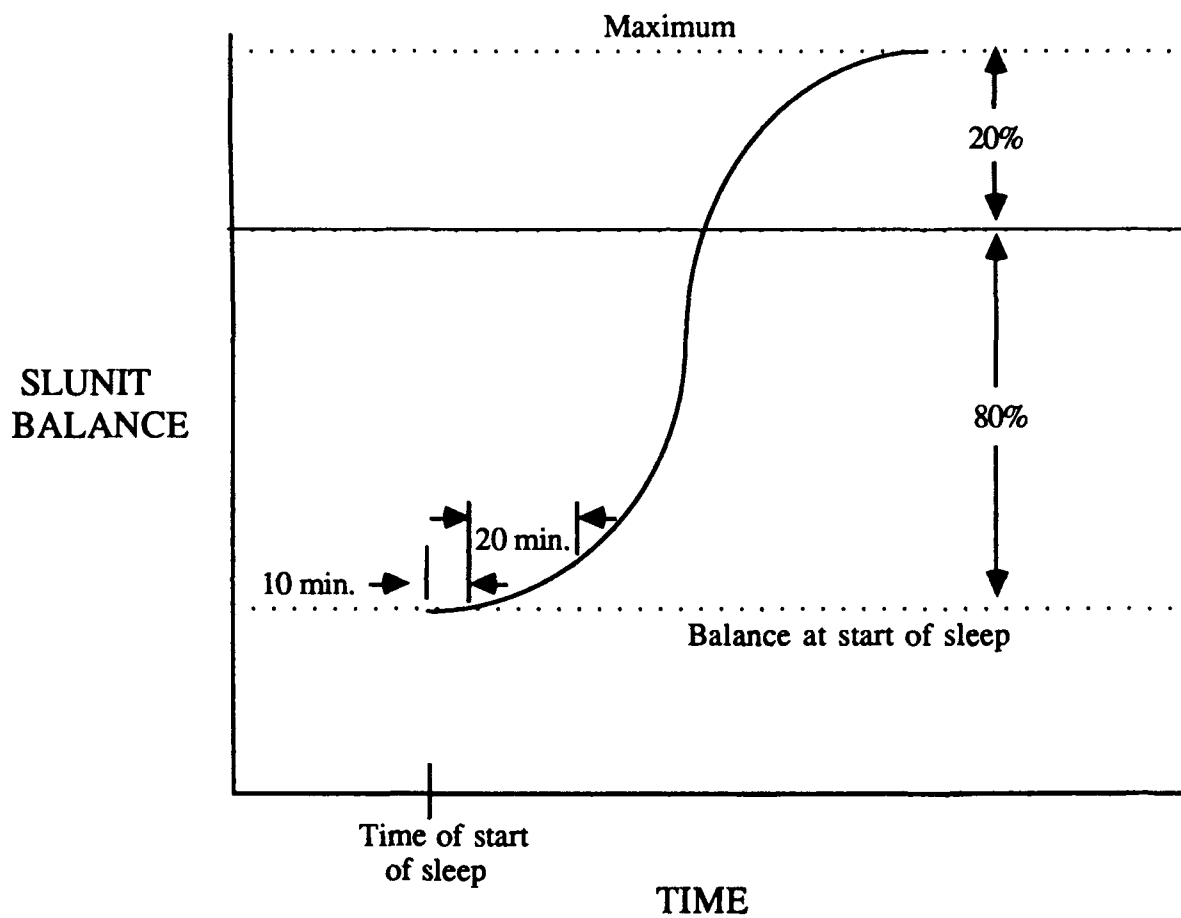


Figure 17. SLUNIT Accumulation (Sleep Efficiency) Curve

acuity due to prolonged employment at a difficult cognitive job at approximately 25% per day. In terms of the AURA fatigue algorithm, this corresponds to an expenditure of 380 SLUNITs per day or 0.264 SLUNITs/minute as the "fatigue rate" for the default case. However, in AURA, the user has the ability to describe the fatigue rate, in SLUNITs/minute, to be associated with any job. In the course of a run, the SLUNIT balance is maintained for each individual, increasing when he is sleeping and decreasing at the job-associated rate when he is working. Figure 18 shows the SLUNIT expenditure and recovery curve.

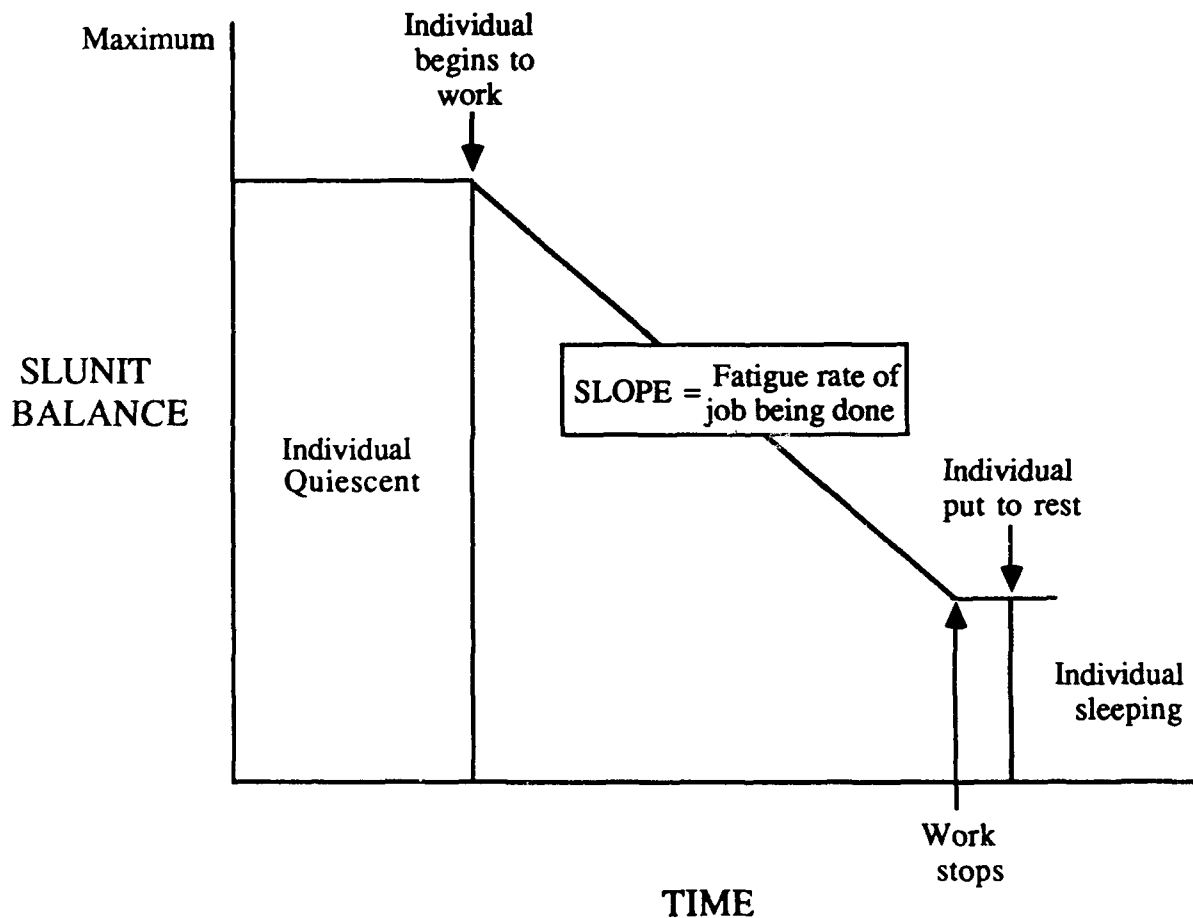


Figure 18. SLUNIT Expenditure and Recovery

(2) **Job Degradation.** Job performance is degraded as a function of the SLUNIT balance of the individual performing the job. This is modeled by establishing, for every job, a SLUNIT demand level. An individual whose SLUNIT balance exceeds the demand level for a particular job suffers no degradation of job performance due to fatigue; below the demand level, job performance goes to zero with the SLUNIT balance. This can be seen in Figure 19.

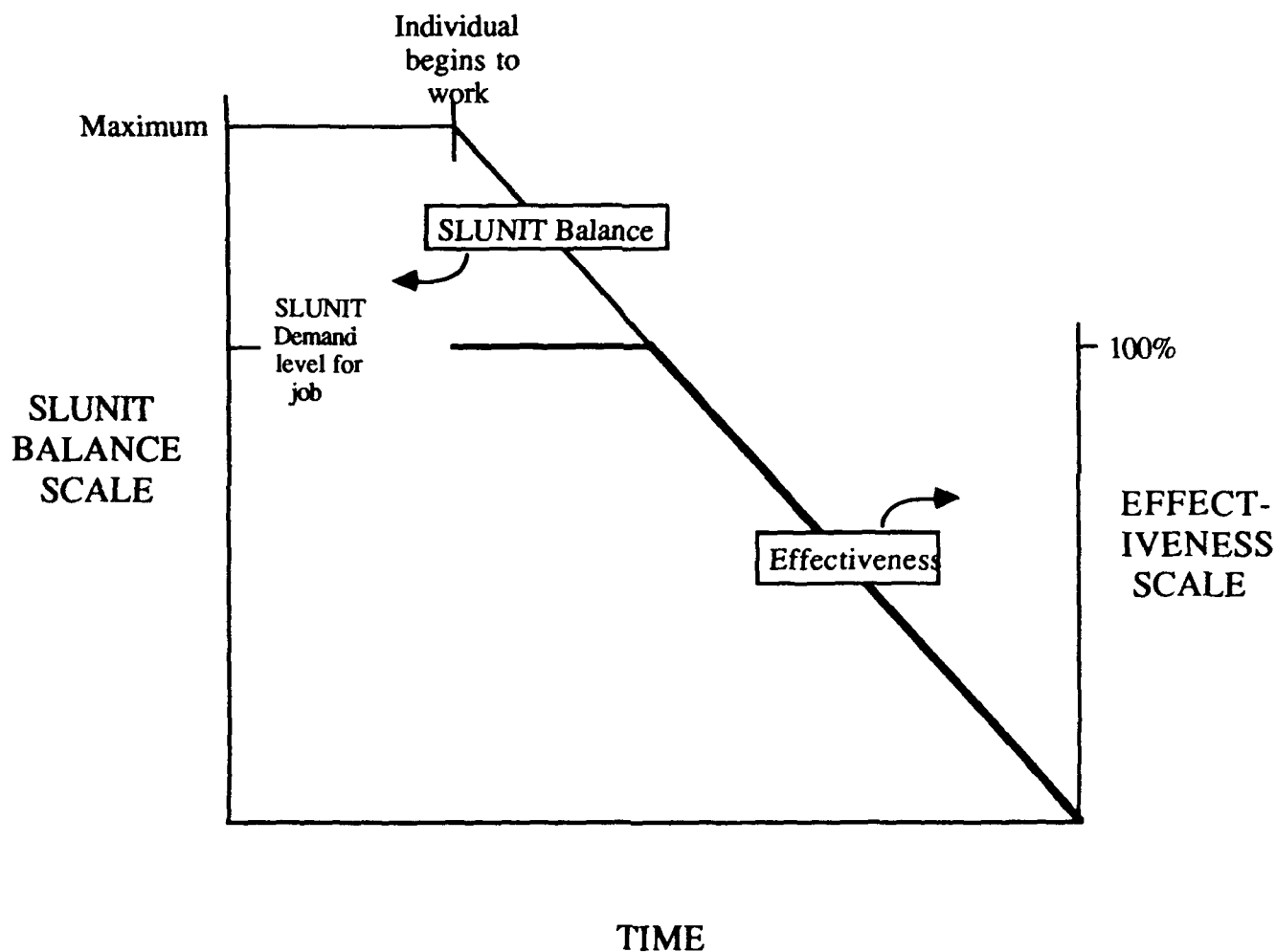


Figure 19. Relationship between SLUNIT Balance and Job Performance

The degradation of job performance due to fatigue is multiplicatively combined as an independent factor with other job performance degradation factors in order to evaluate the value of a personnel asset assigned to a job. This allows fatigue to be fully incorporated in the AURA process, including the logical decision process which leads to the disposition of assets. Four controls are made available to the AURA user which allow the user to influence the decision process as related to fatigue. Three particular controls are SLUMIN, SLUMAX, and SLPMIN. These are shown in Figure 20. The fourth control is a general control on job assignment. SLUMIN is a specified minimum SLUNIT level for an asset. Once an individual reaches this level, he may not be assigned to any task; the individual will be unassigned at the end of the asset allocation process and will be put to rest on an "as-available" basis. SLUMAX is the SLUNIT balance above which the individual may not be put to rest. This forces availability of individuals until they have worked for a certain period. SLPMIN is the shortest sleep interval for an individual. Once put to rest, the individual cannot be disturbed, except by an incoming round, until SLPMIN minutes have elapsed. The TIREDNESS command gives the user the ability to determine a minimum level of individual effectiveness, below which an asset will not be assigned to a task.

b. **The Fatigue Subroutines.** There are three main fatigue subroutines: FATIGU, WAKEUP, and SLEPE. In addition, there are three other subroutines that can contribute to the overall modeling of fatigue: OPTMIZ, LNKOPT, and RPRAVL.

As discussed earlier, the main sleep unit is the SLUNIT. There are three issues around the SLUNIT that form the core of the fatigue modeling: 1) how to accumulate SLUNITs; 2) how to use up SLUNITs; and, 3) what effect do SLUNITs have.

(1) **SLUNIT Build-Up.** The main subroutine for accumulating SLUNITs is FATIGU. At every event, FATIGU checks everyone that is working, the job he is doing and the fatigue rate associated with that job. The code also knows how much time has transpired since the previous update. FATIGU multiplies these two pieces of data to determine how many SLUNITs have been used.

In order to accumulate SLUNITs, the asset must be put to rest. When put to rest, the asset is placed in a sleep bin that allows him to accumulate SLUNITs at a certain rate. The array SLPDOG keeps track of those personnel who are to be put to rest. The actual placing into sleep bins is done by subroutine SLEPE.

(2) **Designating Potential Sleepers.** Any time there is an attack, everyone is awakened by subroutine WAKEUP. At the beginning of every reconstitution, the code reconsiders every single person as a potential worker, whether or not the asset was a sleeper coming into the reconstitution. Subroutines AWAKEN, OPTMIZ and RPRAVL are the three places where the asset can be designated as a potential sleeper and placed in the SLPDOG array which

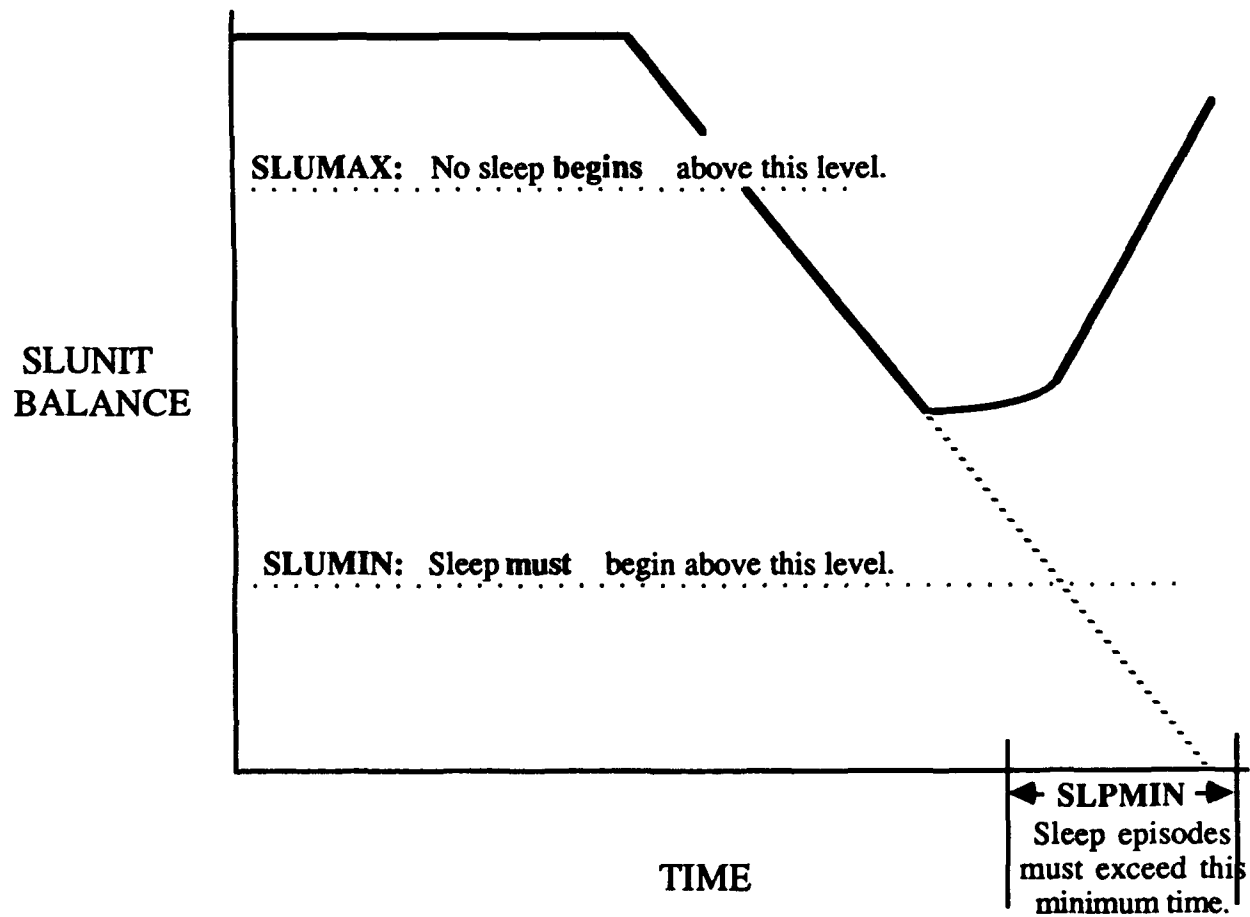


Figure 20 . SLUMIN, SLUMAX, and SLPMIN

indicates the asset is going to be a sleeper. At the beginning of each reconstitution, SLPDOG is zeroed out and the three routines decide if they want to put someone into the SLPDOG array. AWAKEN takes a look at those already sleeping. If they have not stayed asleep the minimum amount of time, AWAKEN places them in SLPDOG (Note: these people are then not available in the pool of assets for OPTMIZ). In addition, those assets whose SLUNIT level has dropped below the minimum SLUNIT level are also placed in SLPDOG by AWAKEN. OPTMIZ can only put one asset to rest. If the weak link is a personnel asset and if the commander can improve unit performance by putting someone to rest, the code will consider this option and see what penalty will be taken by the unit. If the penalty is not too great, OPTMIZ will put that asset in the SLPDOG array. Finally, RPRAVL looks at what repairs can be done with the assets at hand. After all jobs have been assigned, if personnel are left who are fatigued, RPRAVL will put them into the SLPDOG array.

Subroutine SLEPE takes the people in SLPDOG and puts them into the sleep bins. Recall that all personnel were made available at the beginning of a reconstitution. SLEPE looks at the assets in the SLPDOG array and determines if they were removed from the sleep bins at the beginning of the reconstitution. If so, SLEPE tries to replace them in their original bin. This reflects the fact that these assets would not be starting at zero again in the rest cycle but rather would continue from the point at which they were at the beginning of the reconstitution. If they are new assets to be put to rest, SLEPE puts them in an empty bin or creates a new one if no empty bins are available.

(3) **Effects of SLUNITs.** Subroutine LNKOPT quantifies the effects of SLUNITs. LNKOPT determines the minimum amount of SLUNITs needed before a job will become degraded due to lack of SLUNITs. (Recall that each job has a minimum level associated with it.) The asset assigned to the job can do the job with no degradation until the minimum SLUNIT level is reached. Below this minimum level, job performance goes to zero with the SLUNIT balance.

4. Methodology for Degradation Due to Heat Stress

Concurrent with the publication of this report, the methodology used by AURA to determine personnel degradation due to heat stress is undergoing an extensive upgrade. The primary focus of the new methodology (known as the Goldman-Givoni model) is to provide a more accurate heat stress model by considering such factors as: meteorological conditions, acclimatization, and rate of dehydration. This methodology is being documented and will be the subject of an addendum to this report.

The current methodology used by AURA to evaluate the degradation of personnel due to heat stress is described in the reference cited.²⁴

V. Modeling of Lethality Events and Effects

1. Targeting Methodology

AURA's targeting and munition delivery methodology describes the way in which rounds are dispersed across a target in terms of attack time, number and type of rounds, and designated and actual ground zero of incoming rounds. While there are major differences between conventional, nuclear and chemical munition systems, the procedure used to distribute the rounds on the target is the same. This section discusses the types of attacks that can be modeled in AURA, the applicable mnemonic cards and pertinent subroutines.

a. Overview of Targeting and Munition Delivery Methodology.

The method by which AURA models an attack, or a series of attacks, on a unit is simple. Appropriate data are input under the applicable mnemonic headings such as WEAPONS, TLE or CEP TLE, DELIVERY ERROR or CEP, VOLLEY or ROUND, AGENT TYPE or YIELD, FIRE DIRECTION and others. Several defaults do exist in the code for some of these parameters and will be employed if data are not entered.

Conventional, chemical, combined conventional/chemical and nuclear attacks can be modeled in AURA. The types of conventional high explosive (HE) munitions that traditionally have been modeled using AURA are artillery rounds, mortars, improved conventional munitions (ICMs), bombs and rockets. Point detonating or proximity fuzed munitions may be modeled. The differences between the various conventional systems are portrayed by the lethality data, delivery errors and weapon reliability inputs.

Artillery rounds, bombs and rockets with chemical fill have also been modeled. Typically nerve agents such as GB and VX are modeled with AURA. Defaults are included in the code for these two agents. The defaults include personnel performance curves which describe personnel performance as a function of time and sublethal dosages; and probit slopes for the sampling dose-response. (This is further discussed in section V.3.B.)

²⁴. op cit.

Modeling of nuclear attacks in AURA is limited to low altitude bursts. The different nuclear weapons are characterized by their delivery errors and the size of the nuclear yield. The typical nuclear yield modeled within AURA falls between 0.01 and 500 kilotons where the smaller yields are delivered by artillery and the larger yields are delivered by missiles. The yield modeled is limited only by applicability of the nuclear vulnerability equations used to model its effects.

b. **Targeting Mnemonics.** This section discusses each of the mnemonics that can be used to model an attack in AURA. Several examples are included to illustrate how combinations of these mnemonics affect the overall scenario.

(1) **Weapons.** In order to initiate an attack in AURA, a WEAPONS mnemonic must follow the REPERTOIRE or NAMES card in the runstream. The threat systems are given user specified names which are followed by the weapon type: conventional, chemical, nuclear, conventional/chemical, or secondary explosion.

Secondary explosions refer to the explosion of ammunition stacks, fuel tanks or other assets which have the potential to detonate as a result of being hit by incoming rounds. Since secondary explosions represent a threat to unit assets similar to a conventional munition's blast, the lethality of secondary explosions is modeled like that of a conventional munition. (See the conventional lethality section for more details.) Secondary explosions must be given a threat name and listed under both ASSETS and WEAPONS; however, no other targeting mnemonic is required for this type of threat.

(2) **Agent.** If the runstream contains weapons that are designated as chemical or toxic threats, the AGENT card must also be employed. This card specifies the type of agent employed by the weapon. The default type is a non-persistent vapor agent (G). Other agent types currently available are a persistent, percutaneous agent (V) and a blister agent (H). Designation of agent type specifies which set of performance degradation curves the code will use to analyze the performance of personnel who accumulate less than lethal dosages.

(3) **Yield.** The YIELD card is used when nuclear weapons are specified under the WEAPONS card. This mnemonic specifies the yield in kilotons of the nuclear warhead and is used in the nuclear vulnerability calculations.

(4) **Acquisition Probability.** The first consideration in targeting a unit is acquiring it. The mnemonic ACQUISITION PROBABILITY is used to model the probability of acquiring a target. The code performs a random number draw from a uniform distribution to determine whether or not the target was acquired. (This draw is made in subroutine REPSET at the beginning of each replication and upon every change in event in OTHEVN.) If the target is not acquired, the current lethality event is skipped. If this mnemonic is not utilized, the default acquisition probability is 1.0.

(5) **Target Location Error.** Errors associated with the location of a target are called target location errors (TLE). TLE may be changed as a function of time to model changes in the method used to determine target location. After TLE is calculated by the code, it is used in combination with the weapon delivery system and munition errors to determine the actual ground zero of incoming rounds. See subroutines REPSET, OTHEVN and AGZCLC.

There are two mnemonic used to input target location errors in AURA, TLE or CEP TLE. If neither mnemonic is utilized, the code assumes an error of zero. The TLE mnemonic expects an input for both the range (direction from the weapon system to the target) and the deflection (direction perpendicular to the range). The CEP TLE mnemonic accepts target location errors in terms of a circular error probable, the distance from the aim point that the target is expected to be fifty percent of the time.

There are two ways of implementing the TLE mnemonic, that is, to model either a standard normal distribution or a uniform distribution. If the values input under TLE are positive, the code draws from a normal distribution. A positive value input under TLE, therefore, represents the value of a standard deviation. The value of the standard deviation represents the distance from the aim point within which the target is expected to lie approximately one-third of the time. The random number draw from the normal distribution produces both positive and negative numbers and therefore over a sufficient number of replications, the sample of aim points will be distributed normally about the target. Negative values input under the TLE mnemonic represent the radius of a uniform distribution. A negative input value causes AURA to sample from a uniform distribution. The magnitude of these values specify the distance from the aim point that the target is expected to lie.

If a CEP TLE is included in the input runstream, AURA converts it to a standard deviation with which to make the calculation of actual ground zero. This conversion is accomplished in subroutine TLEIN and the calculation of actual ground zero is performed in AGZCLC. In standard practice, when utilizing standard deviations, the normal distribution should be implemented. As with the TLE mnemonic, the code will draw from the normal distribution when the value of the input is positive and the uniform distribution if the value is negative.

(6) **Reliability of Weapon.** An incoming round or volley of rounds may fail to detonate upon arrival at the target. In order to sample the probability of munition functioning, the mnemonic RELIABILITY OF WEAPON is available. If not employed, weapon reliability is assumed to be 1.0. Weapon reliability inputs are read by subroutine RELIIN. Two type of reliability inputs are allowed for the RELIABILITY OF WEAPON mnemonic: single round reliability or volley reliability. Volley reliability is applied to a set of rounds input under the VOLLEY card while round reliability is assessed for each round. Weapon reliability is determined in subroutine LETHAL.

(7) **Delivery Errors.** Weapon delivery errors may be input in terms of standard deviations using the DELIVERY ERROR mnemonic or in terms of a CEP using the mnemonic CEP ERROR. There are four basic configurations that may be implemented when modeling the distribution of rounds. These configurations which apply to both the MPI errors and the round to round errors are illustrated in Figure 21.

The first is the use of the DELIVERY ERROR mnemonic with a uniform distribution. When values of the input under DELIVERY ERROR are negative, the code will sample the errors from a uniform distribution. The upper left hand block in Figure 21 illustrates the area within which the round or volley of rounds would fall given the input rx and ry , where rx is input value for range, ry is the input value for deflection and the center of the coordinate system represents the aim point. The inputs values rx and ry simply represent the distance from the aim point that the round or volley of rounds may fall.

The second configuration is the use of the DELIVERY ERROR mnemonic with a normal distribution. When values of the DELIVERY ERROR input are positive, the code will sample the errors from a normal distribution. The upper right hand block in Figure 21 illustrates the distribution where σ_r is the input value for the range direction and σ_d is the input value for deflection, the center of the coordinate system represents the aim point and the vertical axis represents the probability of a round landing at any coordinate (x,y) . The inputs values σ_r and σ_d represent the standard deviations, that is, the distance from the aimpoint in the x and y directions, respectively, within which one-third of the rounds are expected to fall.

The third configuration is the use of the CEP ERROR mnemonic with a uniform distribution. Negative CEP ERROR inputs are interpreted as the radius about the aim point within which the round or volley of rounds will fall. Negative values are sampled from the uniform distribution. The lower left hand block of Figure 21 illustrates the area within which the round or volley of rounds will fall given CEP ERROR input, R .

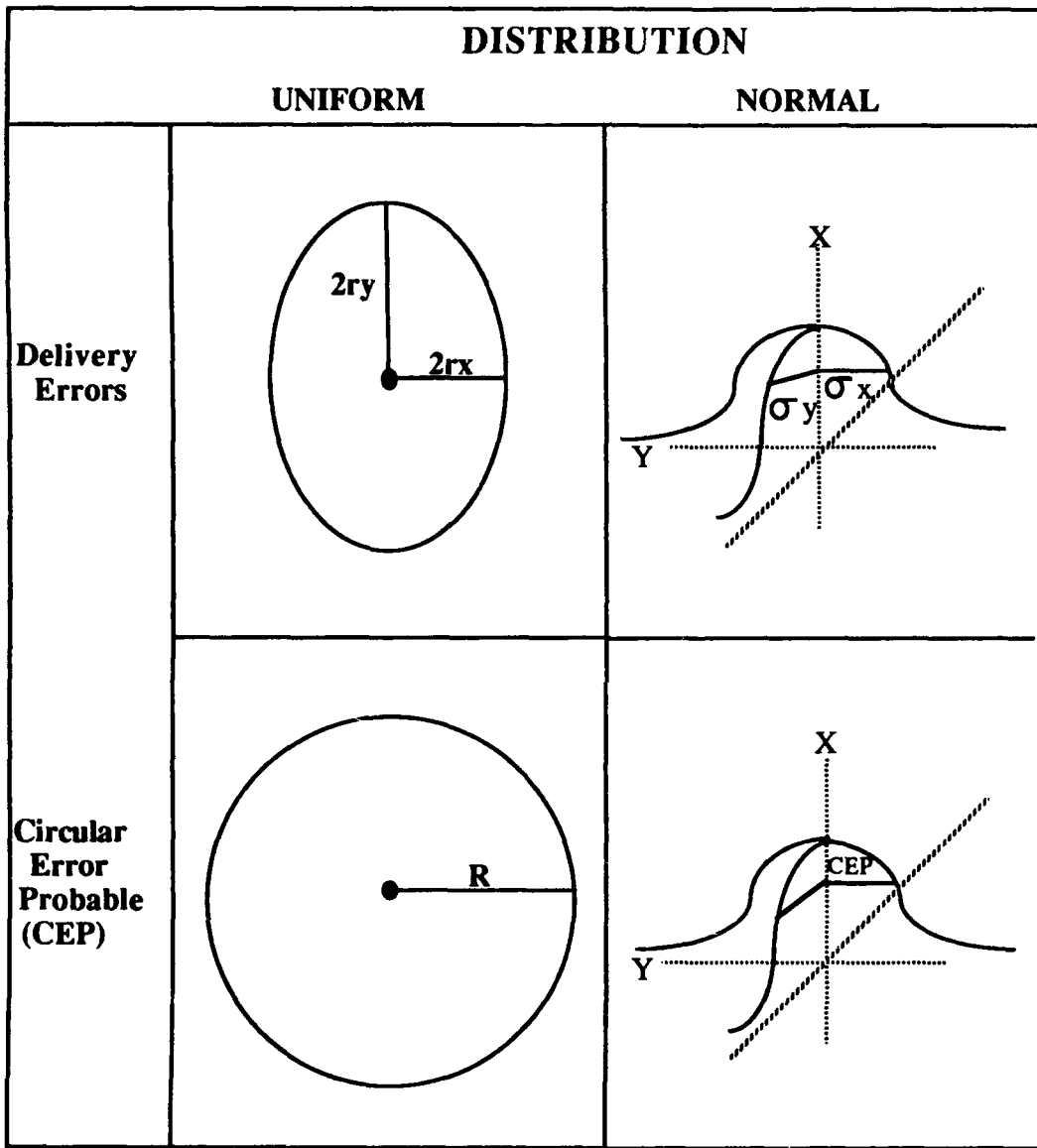


Figure 21. Graphical Descriptions of AURA's Delivery Errors

The fourth configuration is the use of the CEP ERROR mnemonic with a normal distribution. Positive CEP ERROR inputs are interpreted as the distance from the aim point within which fifty percent of the rounds are expected to fall. CEP ERROR inputs described with a normal distribution are converted to a standard deviation within the code:

$$\sigma = 0.85 R,$$

where sigma is the standard deviation and R is the input value of the CEP ERROR. This relationship is derived from the definition of the circular normal density function which is described in detail in section V.1.d.3 of this report.

Height of burst errors may also be input under either the DELIVERY ERROR or CEP ERROR mnemonic. The use of HOB errors is limited to applications of conventional munitions modeling where several sets of lethality data specifying different HOBs have been included in the conventional lethality files. The HOB resulting from the random draw is used to determine which set of lethality data is applicable for that incoming round. HOB errors are not utilized in conjunction with chemical or nuclear attacks.

Both the DELIVERY ERROR and CEP ERROR mnemonics allow for changes in errors as a function of time to simulate changes that may occur on the battlefield over time. Each input mnemonic, DELIVERY ERROR and CEP ERROR, may be used to describe errors in the mean point of impact of an incoming volley of rounds (MPI errors, also called volley-correlated errors) as well as errors in the precision of the independent rounds (call precision errors or round-to-round dispersion errors).

(8) Miss Distance Option. The MISS DISTANCE option is included to allow the analyst to systematically vary the actual ground zero about the target at a specified radius. This mnemonic typically would be used in place of the DELIVERY ERROR or the CEP ERROR mnemonic. Burst points are selected uniformly along a circle of given radius about the target. Height of burst errors may also be varied under this mnemonic; these errors may be selected from either the normal or uniform distribution. MISS DISTANCE errors are sampled in subroutine AGZCLC.

(9) Round and Volley Inputs. In order to set up an attack in AURA, as currently configured, one must specify the aim point of each incoming round using the ROUND mnemonic or specify the midpoint of a set (or volley) of incoming rounds along with a description in terms of line length and volley angle. Line length is the distance over which the rounds are spread, and volley angle is the angle that the line length makes with the direction of incoming fire. MPI delivery errors are drawn on the midpoint of the volley and precision errors are drawn for each of the rounds in the volley. A cluster munition may be modeled by using a line length of zero and delivery errors such that all rounds are

dispersed about the volley midpoint. An ICM may be modeled by using a uniformly distributed range error that spreads the errors about the same midpoint of the volley over a distance equal to the line length.

Aim points are designated using the ROUND or the VOLLEY mnemonic. The ROUND mnemonic is used when only specifying aiming information for one round. Inputs are weapon name, the time of round functioning and the designated ground zero (the intended x and y coordinates and the intended height of burst.) A round to round error and MPI error are drawn for each new line of input under the ROUND mnemonic.

The VOLLEY mnemonic is used to input the parameters of a volley of multiple rounds (of the same weapon name). Inputs are the weapon name, the time that the rounds function (assumed to occur simultaneously), the intended midpoint of the volley, height of burst, the number of rounds in the volley, the angle that the line length makes with the fire direction and the distance over which the round will be dispersed (line length). One MPI error is drawn for each line of input under the VOLLEY mnemonic. A round to round error will be drawn for each round on every line of input.

Currently the analyst must determine attack aim points to input under the ROUND and VOLLEY cards. One option that is available is the barrage option under the VOLLEY card. This option creates multiple volleys, each one stepped in a specified direction by a specified distance. Future improvements to the codes targeting module include the addition options which result in the internal calculation of the aim points and number of rounds fired based on the Battery Computer System (BCS) method of aiming as well as the Fenderkov method of aiming.

(10) **Incoming Fire Direction.** The INCOMING FIRE DIRECTION option allows the analyst to orient the direction of incoming rounds with respect to the targets' coordinate system. Incoming fire direction may be changed as a function of time to model various placements of the threat on the battlefield. Additionally, fire direction may be randomly selected from a uniform distribution. Incoming fire direction calculations are made in subroutines REPSFT and OTHEVN.

When modeling chemical munitions, it is important to remember that the relative angle between the direction of fire and the direction of the wind may have an impact on the shape of the resultant chemical cloud and dissemination pattern. In general, it is appropriate to use the fire direction option in conjunction with the modeling of a point source chemical munition. However, when modeling fire direction changes with a line source chemical munition, the angle between the fire direction and wind direction must be considered and appropriately described by the NUSSE inputs used to produce the chemical patterns.

(11) **Wind Direction.** The WIND DIRECTION option allows the direction of the wind relative to the unit coordinate system, to be set to a constant other than the default angle of zero or to be varied as a function of time or over a range of angles. Calculations are made in subroutines REPSET before each replication and OTHEVN at each change as a function of time. As currently configured, the impact of varying the wind direction is limited to rotating the chemical pattern lay down about its point of function. This is applicable for the modeling of point-source chemical munitions, however, should not be applied to line source chemical munitions. In the default case, the wind and fire direction are both parallel to the positive x-axis.

If modeling changes to the wind and or fire direction where a line source chemical munition is being employed, care should be taken to use an appropriate wind angle variation in the NUSSE description to show the relative angle of the wind to the direction of fire.

c. **Targeting Methodology Algorithms.** The first step in processing targeting information is a call to DEFALT from MAIN. DEFALT sets up important default values and initializes input parameters. Important defaults for targeting purposes include: chemical agent type defaults to a "G" agent, probability of target acquisition defaults to 1.0, round and volley reliability defaults to 1.0, incoming fire and wind angle coincide and default to 0° (along the +x-axis), all delivery and target location errors default to zero and maximum range defaults to 1×10^{36} in the positive and negative directions. Next, MAIN calls ENCIN to input information for the AURA encounter. ENCIN calls each of the targeting input subroutines: RNDIN, VLLYIN, DLVRIN, TLEIN, ACQRIN, RELIIN, and MISDIN.

After inputs are read and stored, MAIN calls REPSET to reset and perform sampling of specific parameters at the beginning of each replication. To do this REPSET calls on the two random number generators that AURA utilizes: UNIFORM, the uniform random number generator, and NORMAL, the normal random number generator. Additionally, REPSET calls COORDT which performs a coordinate transformation of the target location error from range and deflection axes to the unit coordinate system, x,y axes. REPSET is called once at the beginning of each replication.

Once all necessary information is stored and initialized, MAIN calls EVNTDO which processes the events specified in the AURA runstreams. Three types of events may be processed: reconstitution events, lethality events and other events. Lethality events and other events are those types of events included in the targeting methodology algorithms. A lethality event is an incoming round or volley. Other events include a change as a function of time to delivery errors, target location errors, incoming fire direction, wind direction and probability of acquisition. (Other events also include changes to fatigue parameters, described in the Fatigue section of this report).

EVNTDO first processes other events through a call to subroutine OTHEVN and then calls LETHAL to distribute the round or volley on the target. In order to perform its designated functions, LETHAL calls DONDXS to perform a binary search on the ordered y coordinate list to determine the subgroups of targets within range. Once the AGZ coordinate of the round and the x,y deployment coordinate of the target points to be analyzed are determined, LETHAL calls the appropriate lethality subroutines to assess probability of kill and to estimate casualties.

d. Reading and Storing Targeting Information. The subroutine ENCIN calls each of the following subroutines in the order in which the associated mnemonics are encountered in the runstream: RNDIN, VLLYIN, DLVRIN, TLEIN, ACQRIN, RELIIN, and MISDIN. The order in which these subroutines are called is not important unless the CULL mode option is specified. The CULL function, performed in VLLYIN and RNDIN, is a test to determine if each round is within range of target(s). If it is not, it is no longer considered. If using the CULL mode, the weapon characteristic data and deployment data used to perform the test must be read in before volley and round information. That is, the DEPLOYMENT, DELIVERY ERROR or CEP ERROR, TLE, CEP TLE, MISS DISTANCE, CONVENTIONAL, NUCLEAR, and TOXIC mnemonics must precede VOLLEY and ROUND in the runstream. The code will print a diagnostic error message and end the run if the mnemonics are not in the required order for the CULL mode.

(1) Round Inputs and Calculation of Maximum Range. The subroutine RNDIN is called by ENCIN to input the single round attack parameters. RNDIN reads input from the runstream, culls out rounds that are out of range of the target and stores round parameters which are within range under appropriate variable names.

Values read in include: the weapon names, time of arrival, designated ground zero along the x-coordinate, designated ground zero along the y-coordinate and designated height of burst.

The "CULL" function is a procedure by which each incoming round is assessed to be either within range or out of range. If the round is out of range of the target, it is not processed as a lethality event and the next round is then assessed. In order to make this determination, the maximum values of the target location error, delivery error and lethality radii must be available from other subroutines (TLEIN, DLVRIN, MISDIN, CONVIN, TOXIN). These maximum values are added together to determine the maximum range of effects of the weapon.

(2) Volley Inputs and Calculation of Maximum Range. The subroutine VLLYIN is called by ENCIN to input the multiple round attack parameters and multiple volley parameters. VLLYIN reads input from the runstream, culls out volleys that are out of range of the target and stores remaining

volley parameters which are within range under appropriate variable names.

The parameters that are read in include: weapon name, time of arrival, the designated x-coordinate of the pattern midpoint, the designated y-coordinate of the pattern midpoint, the intended height of burst, the number of rounds in the volley, the angle of the pattern line with respect to the incoming fire angle and the width of the pattern line (which specifies the maximum distance over which the rounds of the volley will be spread).

(3) Target Location Error Input and Conversion of CEP TLE to Sigmas. The subroutine TLEIN is called by ENCIN to read the target location error input. It converts CEP to a standard deviation based upon the familiar Gaussian to circular error probable relationship. Finally, it calculates the maximum TLE value for use in CULL option calculations in subroutines VLLYIN and RNDIN. Random number sampling on TLE is subsequently performed in subroutine REPSET before each replication and in OTHEVN when TLE is changed at a specified time, as input under the TLE mnemonic.

The sigma is calculated to be:

$$\sigma \approx 0.85 \text{ CEP},$$

where sigma is the standard deviation. This relationship is derived as follows²⁵: The circular normal density function expresses the density of impacts on a circle of radius r center at the origin.

$$\delta(r) = (r/\sigma^2) \exp(-r^2/2\sigma^2)$$

Integrate to determine the cumulative circular normal distribution:

$$P = 1 - \exp(-R^2/2\sigma^2)$$

where P is the probability that impact is within a radius of R of the aim point. The CEP of a circular normal distribution is defined as that value of R for which the cumulative circular normal distribution has a value of 0.5, therefore,

$$.5 = 1 - \exp(-\text{CEP}^2/2\sigma^2)$$

$$-\text{CEP}^2/2\sigma^2 = \log_e(.5)$$

25. Groves, Art, "Handbook On The Use of The Bivariate Normal Distribution In Describing Weapon Accuracy(U)", Memorandum Report No. 1372, USA Ballistic Research Laboratory, September 1961, (UNCLASSIFIED)

$$CEP^2/2 \sigma^2 = \log_e(2)$$

$$CEP^2 = 2\sigma^2 \log_e(2)$$

$$CEP = \sigma \sqrt{2 \log_e(2)}$$

$$CEP = 1.1774 \sigma$$

$$\sigma = 0.85 CEP$$

(4) **Delivery Error Input and Conversion of CEP Errors to Sigmas.** The subroutine DLVRIN is called from ENCIN to read the delivery error input. The parameters read by DLVRIN include: weapon name, time of error change (if any), the precision and MPI errors in the range and the precision and MPI errors in deflection. If the CEP ERROR mnemonic was specified, the code sets the range error equal to the deflection error and then converts to a standard deviation based upon the Gaussian to circular error probable relationship previously described. Finally, it calculates the maximum delivery error value for range and deflection for use in CULL option calculations in subroutines VLLYIN and RNDIN. Random number sampling on delivery errors is subsequently performed in subroutine AGZCLC.

(5) **Probability of Acquisition Input.** Subroutine ACQRIN's function is to read the probability of target acquisition input. Determination of target acquisition is made in subroutine REPSET. If data is not provided for this mnemonic, the code assumes that the probability of acquisition is 1.00.

(6) **Reliability of Weapon Input.** Subroutine RELIIN's function is to read weapon reliability input which includes a parameter specifying individual round reliability as well as volley reliability. If only one input is given, the code assumes it is single round reliability. In this case, volley reliability defaults to a probability of 1.0. Sampling on weapon reliability is performed in subroutine LETHAL.

(7) **Miss Distance Input.** Subroutine MISDIN reads in the weapon name, radius, and height of burst error (if there is one). If two are read, the second is assumed to be a height of burst variable. Height of burst is used to determine which set of conventional lethality data to access given that there is height of burst variability.

e. **Sampling Before Each Replication.** Subroutine REPSET is called from MAIN, before the start of each replication, to reset the variables holding values for incoming fire direction, the wind direction, the probability of target acquisition, the target location errors and the delivery errors. If wind or fire direction is sampled over a range, the new angles are calculated in REPSET. Sampling on TLE is also performed in REPSET. The subroutine COORDT is

called from REPSET to transform the TLE coordinates into the xy-coordinate system of the unit deployment.

The fire direction is reset for each replication if the random fire direction option is implemented. This option is used to specify a range over which the fire direction will be uniformly varied. Inputs include the time at which the change in fire direction occurs and the two angles, $\theta 1$ and $\theta 2$, from which the new fire direction ($\theta 3$) can be calculated. The code first draws a random number, X , from the uniform random number generator such that X is between 0 and 1 inclusive, and the new angle, $\theta 3$ is calculated as follows:

$$\delta = \theta 2 - \theta 1 ;$$

$$\theta 3 = \theta 1 + X * \delta ;$$

Therefore, the new angle of incoming fire, $\theta 3$, always falls between the two inputs $\theta 1$ and $\theta 2$.

The wind direction is reset in the same manner as the new fire direction if the random wind direction option is implemented. This option is used to specify a range over which the wind direction will be uniformly varied. Inputs include the time at which the change in wind direction occurs and the two angles, $\theta 1$ and $\theta 2$ between which the new wind angle, $\theta 3$ will always fall.

At the beginning of each replication the probability of target acquisition is assessed. A uniformly distributed random number, RX , is drawn and then compared to the acquisition probability, P_a . If $RX > P_a$, then the target is not acquired and the lethality event is skipped. However, if $RX \leq P_a$, then the target was acquired and the lethality event is assessed.

Target location errors are sampled at the beginning of each replication. If the value of the error in the range, $xerror$, is negative then both the range and deflection TLEs are drawn from a uniform distribution. Given a uniform distribution, the TLE input is defined as the maximum distance from the target in either the positive or negative direction within which the aim point may lie. The actual target location errors, $atlex$ and $atley$ in the x and y directions, respectively are calculated as follows when uniformly distributed:

$$atlex = tlex - 2 * R1 * tlex ;$$

$$atley = tley - 2 * R2 * tley .$$

The uniform random numbers, $R1$ and $R2$, have values between zero and one, and $tlex$ and $tley$ are the values input for the target location error in range and deflection, respectively. These calculations produce samples in the TLE along both the the positive and negative range and deflection axes.

If the value of the error in the range is not negative, then both the range and deflection TLEs are sampled from a normal distribution. The normally distributed random numbers, N1 and N2, have no bounds in either the positive or negative direction, however two-thirds of all numbers sampled will lie on the interval of 2 standard deviations (between -1.0 and 1.0). The TLE input for both range and deflection is defined as one standard deviation from the target. The actual target location, when the error is assumed to be normally distributed is calculated as follows:

$$\text{atlex} = \text{N1} * \text{tlex}$$

$$\text{atley} = \text{N2} * \text{tley}$$

where tlex and tley are the values of the standard deviations in the x and y location of the target, respectively.

After determining the actual target location error for range and the actual target location error for deflection as described above, the code calls subroutine COORDT to perform a coordinate transformation.

The final targeting function that is performed in REPSET is the resetting of height of burst, round-to-round and correlated errors to the values specified on the input stream. These errors are subsequently sampled in subroutine AGZCLC.

f. Coordinate Transformation Subroutine. Recall that the target location errors are defined for the range (the direction of fire) and for deflection (the direction perpendicular to the range) and that the direction of fire may change. Since the fire direction is specified in terms of an angle which is relative to the x-axis of the unit deployment system (xy-coordinate system), the target location errors must be translated such that they are described relative to the unit deployment coordinate system. Figure 22 illustrates the two sets of axes; the xy-coordinate system describes the deployment of the unit, and the x'y' coordinate system describes the incoming attack where ϕ is the angle between the x-axis and the direction of incoming fire.

From this illustration one may see how the formulas for rotation of axes through angle ϕ are implemented. Let the angle ϕ be the angle that the incoming fire direction (x') makes with the positive x-axis, and the angle θ be the angle that the positive x' axis makes with the line P, as shown in Figure 22. For the purpose of illustrating the coordinate system transformation, let the point x',y' be the resultant aim point after TLE calculations. The problem now is to determine the aim point x,y in terms of the unit deployment system. By definition of the sine(sin) and cosine(cos) trigonometric functions:

$$x' = p * \cos \theta ;$$

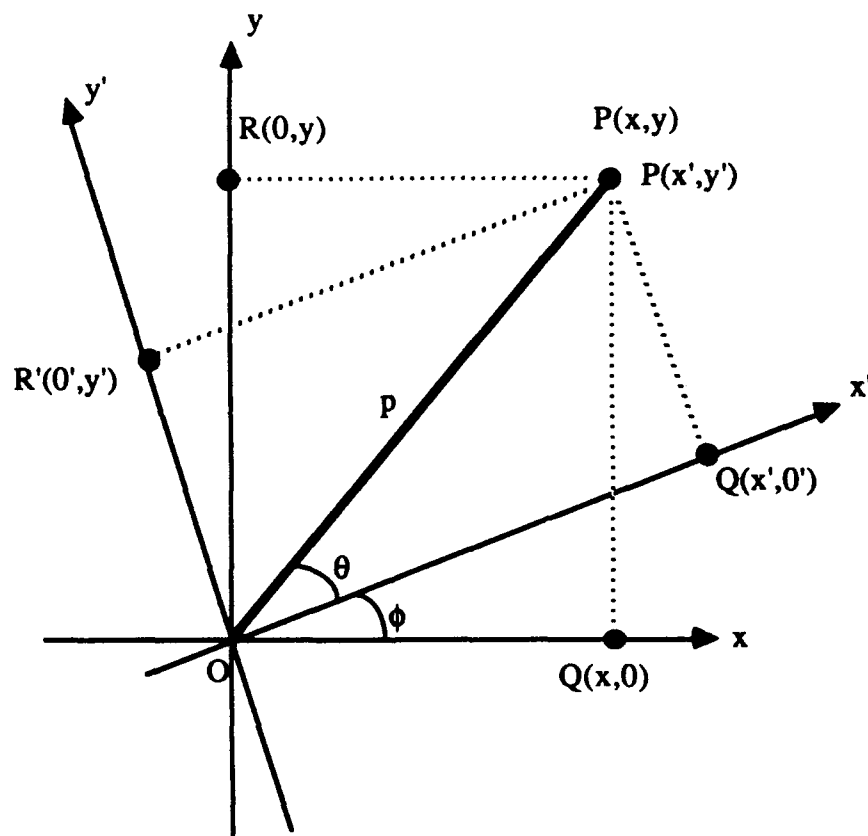


Figure 22. The Unit Deployment (xy - coordinate) and the Incoming Attack ($x'y'$ coordinate) systems.

$$y' = p * \sin \theta ;$$

$$x = p * \cos(\theta + \phi) ;$$

$$y = p * \sin(\theta + \phi) .$$

By trigonometric identity formulas:

$$x = p * (\cos\theta\cos\phi) - p * (\sin\theta\sin\phi) ;$$

$$y = p * (\sin\theta\cos\phi) + p * (\cos\theta\sin\phi) .$$

Solving for x and y by substitution:

$$x = x' * \cos\phi - y' * \sin\phi ;$$

$$y = y' * \cos\phi + x' * \sin\phi .$$

g. Event Processing. Once all necessary information is stored and initialized, MAIN calls EVNTDO which processes the events specified in the AURA run-streams. Three types of events may be processed: reconstitution events, lethality events and other events. Lethality events and other events are those types of events included in the targeting methodology algorithms. A lethality event is an incoming round or volley. Other events include a change to delivery errors, target location errors, incoming fire direction, wind direction and probability of acquisition which have been specified to occur at a given time. (Other events also includes changes to fatigue parameters, described in the Fatigue modeling section of this report).

EVNTDO first processes other events through a call to subroutine OTHEVN. OTHEVN uses the two random number generators UNIFORM and NORMAL, as well as the coordinate system transformation subroutine, COORDT. After the required parameters have been updated as a function of time, EVNTDO checks the logical variable LPACQR to determine whether or not the target was acquired. If the target was not acquired, the lethality event is bypassed. If the target was acquired, then EVNTDO calls LETHAL to distribute the round or volley on the target.

LETHAL performs several targeting functions including the test for weapon reliability (using the uniform random number generated.) Weapon reliability is determined first so that the remaining calculations may be by-passed if the weapon fails. If the weapon does not fail, LETHAL calls AGZCLC to calculate the actual ground zero for the incoming round or volley. AGZCLC uses the random number generators UNIFORM and NORMAL as well as the coordinate transformation subroutine COORDT. Once the AGZ is determined, LETHAL calls the appropriate lethality subroutines depending on the type of event

(conventional, chemical, nuclear or conventional/chemical).

(1) Processing of Other Event Types that Change with Time.

The subroutine OTHEVN is called by EVNTDO to process events other than reconstitution or lethality events. These events include changes to: delivery errors, target locations errors, incoming fire direction, wind direction and probability acquisition at specified times throughout the scenario. These other events are triggered by the inclusion of time changes to these parameters under the mnemonics. The random number draw and recalculation of TLE is performed according to the standard procedure discussed in TLEIN. The new wind and/or new fire direction calculations are performed according to the procedure discussed in REPSET. The probability of acquisition is redetermined as shown in REPSET. OTHEVN is also called when there is a change to delivery errors. The random number draw on delivery error is made in either REPSET or AGZCLC.

(2) Processing Lethality Events.

LETHAL performs several targeting functions including the test for weapon reliability (using the uniform random number generated.) Weapon reliability is determined first so that the remaining calculations may be by-passed if the weapon fails. If the weapon does not fail, LETHAL calls AGZCLC to calculate the actual ground zero for the incoming round or volley and DONDXS to perform a binary search on the ordered y coordinates. LETHAL then calls the appropriate lethality subroutines to assess probability of kill and estimate casualties.

If a volley of rounds is being dispersed, the code calculates the placement interval between rounds, dx, along the x-axis and dy, along the y axis:

$$dx = L * \cos(\theta)/(N-1);$$

$$dy = L * \sin(\theta)/(N-1);$$

where θ is the angle between the positive x axis and the volley direction as shown in Figure 23, N is the number of rounds in volley, and L is the distance over which rounds in volley will be dispersed.

Each round is processed individually in LETHAL. The designated ground zero coordinate, DGZ(X,Y) of the first of N rounds to be processed is located at one extreme end of the volley:

$$DGZ(X_1) = X_0 - 0.5 * L * \cos(\theta);$$

$$DGZ(Y_1) = Y_0 - 0.5 * L * \sin(\theta).$$

where (X₀,Y₀) are the coordinates of the volley midpoint (the aim point of the volley). The DGZ of each round in the volley is subsequently found by adding in the value of the distance dx and dy;

$$DGZ(X_{i+1}) = DGZ(X_i) + dx,$$

$$DGZ(Y_{i+1}) = DGZ(Y_i) + dy; \text{ for } i=1, \dots, N.$$

The $DGZ(X,Y)$ is then passed into subroutine AGZCLC to determine actual ground zero coordinates based on target location and delivery errors.

Figure 23 illustrates an example where the rounds of the volley are to be dispersed along a line perpendicular to the direction of incoming fire and the direction of incoming fire makes an angle of $\alpha=45^\circ$ with the $+x$ -axis.

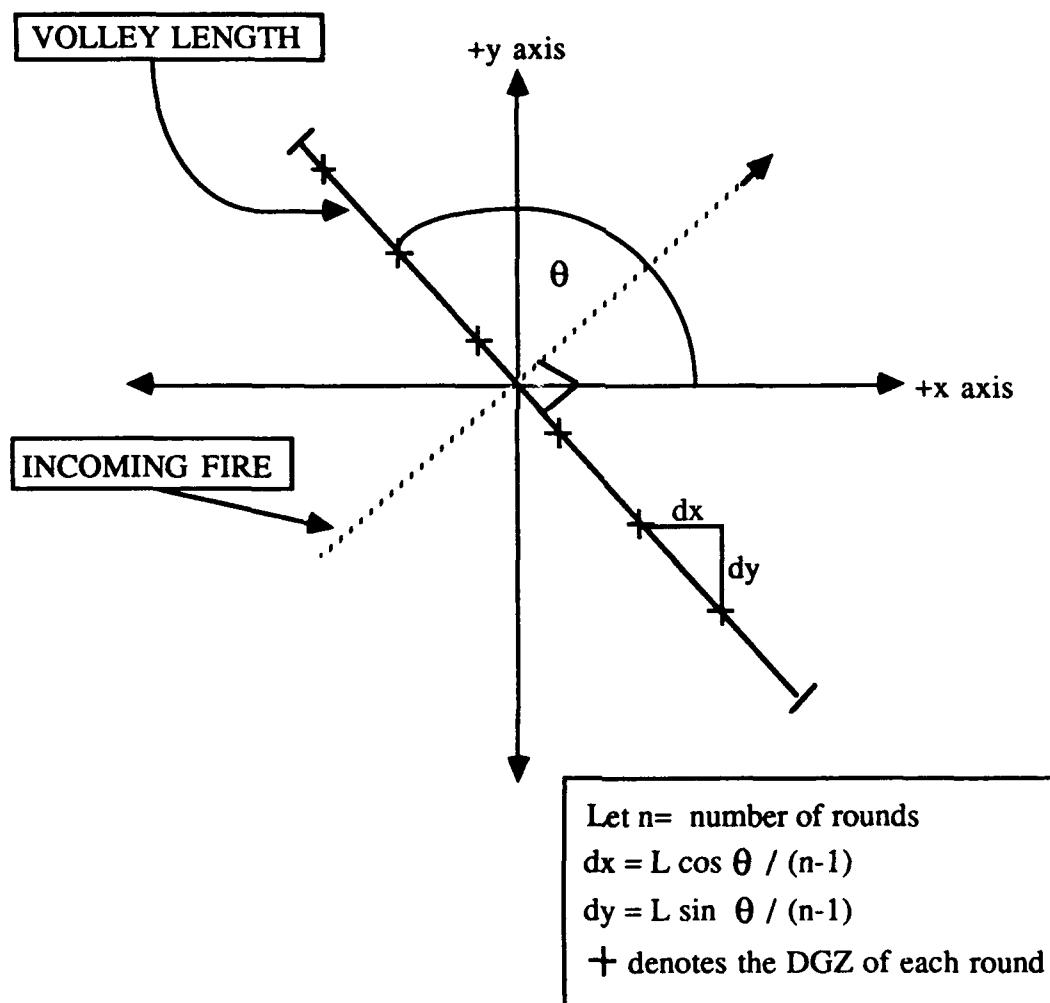


Figure 23. Dispersal of Rounds in a Volley

The values of the AGZ and DGZ coordinates are summed in subroutine LETHAL, and the mean of the AGZ, the mean of DGZ and the standard deviation of the AGZ are calculated in ENDOUT and then written to the standard output file. The means are calculated for each value as follows:

$$\mu = \left\{ \sum_{i=1}^N X_i \right\} / N$$

where X is either the x or y coordinate of either the AGZ or DGZ for the appropriate calculation. The standard deviations of the actual ground zero coordinates (σ_a) are each calculated as follows:

$$\sigma_a = \left(\frac{\sum_{i=1}^N X^2 - \mu \sum_{i=1}^N X}{N-1} \right)^{\frac{1}{2}}$$

(3) Calculating Actual Ground Zero. This routine performs sampling of delivery errors and then combines the delivery errors with target location errors in order to determine the actual ground zero coordinates. First the code draws from either the uniform or normal random number generator to determine the delivery error based on input through one of the delivery error mnemonics, MISS DISTANCE, DELIVERY ERROR or CEP ERROR (where CEP inputs have been converted to sigmas). The delivery errors, S(1), S(2), S(4) and S(5), are calculated differently depending upon the mnemonic used to input the error estimate and the distribution from which they are drawn.

The mnemonic MISS DISTANCE may be used to vary the burst point of the munition about the target at a specified radius. To distribute the round about a circle, a uniformly drawn random number is used to determine the angle between the x-axes and the burst point. For conventional and nuclear events, the calculation is as follows:

$$\begin{aligned} S(1) &= \text{rnd} * \cos(U(1) * 2 * \pi), \\ S(2) &= \text{rnd} * \sin(U(1) * 2 * \pi), \end{aligned}$$

where rnd, the round to round error input value, is the desired miss distance and U(1) is the random number drawn from a uniform distribution; S(1) and S(2) are the resultant x and y burst points based on this type of delivery error description.

If the lethality event type is either chemical or chemical/conventional, the code limits the burst point to the upwind direction. For this case S(1) and S(2) are determined to be:

$$\begin{aligned} S(1) &= -\text{rnd} * \cos(\theta), \\ S(2) &= -\text{rnd} * \sin(\theta), \end{aligned}$$

where θ is the angle between the wind and the x-axis. In order to include a randomness in this determination, one must also include the WIND DIRECTION option where the angle will be uniformly varied between two specified angles such as 0° and 360° . The correlated errors, S(4) and S(5), are set to zero when the MISS DISTANCE option is used.

The more standard approach to delivery error modeling is accomplished through use of the input mnemonics DELIVERY ERROR and CEP ERROR. If a CEP ERROR is used, inputs are converted to sigmas in subroutine DLVRIN.

Given that the normal distribution has been selected, the range and deflection round to round errors, rerr(1) and rerr(2), and the range and deflection correlated errors, cerr(1) and cerr(2) are multiplied by the random numbers R(1), R(2), R(4), and R(5). The equations producing normally distributed errors are as follows:

$$\begin{aligned}RR(1) &= R(1) * rerr(1), \\RR(2) &= R(2) * rerr(2), \\RR(4) &= R(4) * cerr(1), \\RR(5) &= R(5) * cerr(2).\end{aligned}$$

Because these errors are in terms of the weapon delivery system, range and deflection coordinate system, the subroutine COORDT must be called to perform a transformation to the unit deployment system, the x,y coordinate system. This transformation produces the equivalent error estimates with variable names S(1), S(2), S(4) and S(5).

If the round to round errors, rerr(1) and rerr(2), are to be drawn from a uniform distribution and the resultant calculations produce an elliptical distribution such as might be applied to a typical cluster munition, then:

$$\begin{aligned}RR(1) &= (U(1)^{1/2} * rerr(1) * \cos(2 * \pi * U(2))) \\RR(2) &= (U(1)^{1/2} * rerr(2) * \sin(2 * \pi * U(2)))\end{aligned}$$

where U(1) and U(2) are random numbers drawn from a uniform distribution and RR(1) and RR(2) are the resultant coordinates in range and deflection, respectively. This is derived from the trigonometric relationships: $x = r * \cos(\theta)$ and $y = r * \sin(\theta)$. In the above set of equations, θ is the random variable: $2 * \pi * U(2)$, and r is the random variable: $U(1)^{1/2} * rerr(i)$ for $i=1,2$; where rerr(1) is the range error and rerr(2) is the deflection error. The resulting values of RR(1) and RR(2) are subsequently transformed to the equivalent values in the unit deployment x,y coordinate system through a call to subroutine COORDT.

If both range and deflection errors are to be drawn from the uniform distribution, AURA will use the above equations to produce the elliptical distribution for the cluster munition. If the range or deflection round to round errors are

specified to be drawn from a different distribution, then the following calculation is used for which ever one is being drawn from the uniform distribution:

either $RR(1) = (2 * U(1) - 1) * rerr(1),$
or $RR(2) = (2 * U(2) - 1) * rerr(2),$

where $U(1)$ and $U(2)$ are random numbers and $rerr(1)$ and $rerr(2)$ are the range and deflection round to round errors, respectively. The remaining round to round error is calculated in accordance with the equations given for normally distributed errors. Values are transformed to $S(1)$ and $S(2)$ equivalents for use in the unit deployment (x,y) coordinate system through a call to COORDT.

If the correlated errors are to be uniformly distributed, the error estimates for $RR(4)$ and $RR(5)$ are calculated:

$RR(4) = (2 * U(4) - 1) * cerr(1),$
 $RR(5) = (2 * U(5) - 1) * cerr(2),$

where $U(4)$ and $U(5)$ are the random numbers and $cerr(1)$ and $cerr(2)$ are the range and deflection correlated error inputs, respectively.

The height of burst error, $herr$, is calculated in the same manner for all three delivery error input options:

either $herrz = R(3) * herr,$
or $herrz = (2 * U(3) - 1) * herr,$

where $herr$ is input value of height of burst, $U(3)$ is the uniform random number and $R(3)$ is the normal random number. No transformation is required on the height of burst error estimate since this coordinate axis remains the same between both coordinate systems.

After delivery error sampling, actual ground zero calculations for the X and Y burst point on the ground and the height of burst, Z, are made as follows:

$X = xaim + tlex + RR(1) + RR(4)$
 $Y = yaim + tley + RR(2) + RR(5)$
 $Z = zaim + herrz$

$xaim$, $yaim$, $zaim$ are the aim points along each axis, $tlex$ and $tley$ are the actual target location errors along each axis, $RR(1)$ and $RR(2)$ are the round to round errors, and $RR(4)$ and $RR(5)$ are the correlated errors along the respective axes.

Control of delivery error calculations is determined by the mnemonic used to input aiming information. Each incoming round input under the ROUND card will have a uniquely sampled round to round and MPI error per replication.

Each incoming round input under the VOLLEY card will also have uniquely sampled round to round error per replication. All rounds input on the same line of input under VOLLEY card will have the same MPI error per replication. MPI errors are drawn once per replication for each line of input under the VOLLEY card. For example, if 100 rounds are specified on one line of input under VOLLEY, then actual ground zero for these 100 rounds will be based on 100 different round to round errors and 1 MPI error.

h. Application of Targeting Methodology to Hypothetical Case. As a summary of the targeting methodology, five examples of possible weapon lay downs are described; single incoming rounds, a cluster munition with normally distributed rounds, a cluster munition with uniformly distributed rounds, a rectangular sheaf and a rolling barrage.

The examples presented in this section are intended to provide a small sample of the ways in which the targeting mnemonics may be used. Because of the flexibility of the code, there are many other ways in which to simulate standard doctrinal methods of firing as well as to approximate special uniform or normal distributions of rounds on target.

(1) Single Incoming Rounds. Single incoming rounds are simulated using the ROUND mnemonic. The code will draw a round to round and MPI error for each round. To illustrate the use of the ROUND mnemonic, suppose it is desired to drop 4 bombs with the name BOMB at the designated ground zeros: (0.,0.), (0.,25.), (25.,25.) and (25.,0.) such that the bombs are detonated at one minute intervals beginning at time 5. The following lines of input could be used:

```
ROUND
BOMB,5.,0.,0.,0.
BOMB,6.,0.,25.,0.
BOMB,7.,25.,25.,0.
BOMB,8.,25.,0.,0.
END
```

The first alphanumeric input indicates the name of the round. The first real number indicates the time after the onset of the scenario that the bomb detonation will occur. The second, third and fourth real numbers indicate the x,y,z coordinates for the designated ground zero. In this example, no errors were inputs. Therefore, the rounds will function exactly at their aim points.

(2) Cluster Munitions. A convenient way to model a munition containing submunitions in AURA is as a volley of submunitions. For example, if the attack consisted of the employment of one bomb containing 1000 submunitions, the following lines of input could be used to described it:

```

VOLLEY
BOMB,60.,0.,0.,0.,1000,0.,0.
END
CEP ERRORS
BOMB,-150.,0.,0.
END

```

These lines of input describe an attack that occurs 60 minutes after the beginning of the scenario. The weapon named BOMB is aimed at the (x,y,z) coordinates (0.,0.,0.). At burst, 1000 submunitions will be dispersed about the aim point in a pattern described by the CEP ERROR input. Note that the line length of the volley was set to zero for this type of munition. The angle of the pattern relative to the incoming fire direction may be set according to scenario assumptions. In this example, the CEP ERROR input describes the round-to-round dispersion such that rounds will be uniformly dispersed across the radius of 150 meters from the aim point. The MPI error is set to zero as is the height of burst error. An MPI error greater than zero would act as an error in the location of the pattern center. Since MPI errors were set to zero in this case, the center of the pattern will lie on (0.,0.) as aimed.

Consider the above set of inputs again with a positive value of the CEP ERROR inputs:

```

VOLLEY
BOMB,60.,0.,0.,0.,1000,0.,0.
END
CEP ERRORS
BOMB,150.,0.,0.
END

```

Because the positive value of the inputs triggers the code to draw from a normal distribution, CEP ERROR input will be converted to a standard deviation. The result is a normally distributed round pattern about the aim point.

(3) **Rectangular Sheaf.** Suppose the target is considered a rectangular target of 200 by 100 meters and it is desired to fire a rectangular sheaf of eight rounds on the target. The target is deployed such that the lower left hand corner of its rectangular area has the coordinate (0.,0.) and the center of the target is located at coordinate (100.,50.). To disperse eight rounds, called SHELL in this example, on the target as per an eight-tube artillery battery, the following lines of input could be used:

```

VOLLEY
SHELL,60.,25.,50.,0.,4,90.,200.
SHELL,60.,75.,50.,0.,4,90.,200.
END
DELIVERY ERROR
SHELL,10.,40.,8.,55.
END

```

The MPI error, $\sigma_x = 40$ and $\sigma_y = 55$, would be sampled once per replication for the volley aimed at the coordinate (25.,50.) and once for the volley aimed at the coordinate (75.,50.). Once the mean point of impact for a volley is determined, the rounds will be evenly distributed about the aim point across the specified length of 200 meters at the specified angle to the range direction (in this case, 90 degrees). Furthermore, the round to round errors, $\sigma_x = 10$ and $\sigma_y = 8$, are sampled once for every round per replication.

(4) **Rolling Barrage.** Suppose it is desired to fire over a large area of terrain. The multiple volley option in AURA may be used to replicate volleys of similar number and dispersion. Consider, for example, a volley that consists of eight rounds equally spaced over a line length of 100 meters which is situated at a 45° angle to the direction of range. Each volley in the barrage is to be separated by 150 meters and fired at 1 minute intervals. A total of 6 volleys is desired. The movement of the intended pattern midpoint from one volley to the next is at a 90 degree angle (measured counter-clockwise from the x direction). This could be modeled with the following lines of input:

```

VOLLEY
SHELL,1.,0.,0.,0.,8,45.,100.
$6,1.,90.,150.
END

```

In fact the resultant round pattern is no different than what would be produced by:

```

VOLLEY
SHELL,1.,0.,0.,0.,8,45.,100.
SHELL,1.,0.,150.,0.,8,45.,100.
SHELL,1.,0.,300.,0.,8,45.,100.
SHELL,1.,0.,450.,0.,8,45.,100.
SHELL,1.,0.,600.,0.,8,45.,100.
SHELL,1.,0.,750.,0.,8,45.,100.
END

```

This set of inputs would result in the placement of 48 rounds across an area of approximately 100 by 750 meters. One MPI error would be drawn for each of the 6 volleys, with round to round errors being drawn individually for each round.

2. Conventional Lethality Modeling

The conventional lethality model is a procedure by which the number of kills resulting from a conventional attack on a unit is estimated. (The term "kills" is used here to cover the range of damage and casualty criteria.) The availability of unit assets can be one of the most important factors in determining unit effectiveness. Therefore it is important to have a sound representation of the munition lethality and unit vulnerability.

a. **Overview of the Conventional Lethality Model.** In AURA, the lethality of indirect, conventional fire munitions is defined by "lethal area". Lethal area is a measure of the warhead's ability to incapacitate a target. Given the probability of incapacitation of the target at the point (x,y) from one round is $P_k(x,y)$, the lethal area can be defined as:²⁶

$$A_l = \int_{-i\infty}^{\infty} \int_{-i\infty}^{\infty} P_k(x,y) dx dy .$$

Lethal area is used to estimate the probability of kill as a function of the distance from the round to the target. The Survival Rule states that the probability of surviving equals one minus the probability of kill, the survivor rule is then used to determine the probability of surviving a multiple round attack: The joint probability of events equals the product of the probabilities of the separate occurrences.

Lethal areas have been calculated for many targets against a wide range of conventional munitions. Lethal areas vary as a function of fall angle, type of fuze (point detonating or proximity), target posture and kill criteria. In order to model conventional lethality, the weapon name, fall angle and fuze type for each different munition type must be defined. Secondly, conventional kill criteria and protective postures for each target must be defined. While the conventional kill criteria and conventional target postures are designated in the Deployment section of the AURA runstream file, these physical attributes are described by the data in the conventional lethality file.

26. Myers, K.A., "Lethal Area Description", BRL Technical Note No. 1510, July 1963, (UNCLASSIFIED)

One source of lethal area data for artillery munitions is the Army Materiel Systems Analysis Activity's "Handbook Series G, No. 5, Performance Characteristics of Artillery and Mortar Systems".²⁷ Other sources are the Joint Munitions Effectiveness Manuals published by the Joint Technical Coordinating Group for Munitions Effectiveness.

(1) **Conventional Kill Criteria.** In order to estimate the number of kills resulting from a conventional attack, the vulnerability of personnel, equipment and supplies to the incoming munition must be known. First, one must define the meaning of vulnerability in terms of a resultant level of incapacitation given a hit, that is, a kill or casualty criterion. (The phrase "damage criteria" is usually associated with equipment and "casualty criteria" with personnel). This criterion is generally chosen based on the objective of the analysis. If the analysis is concerned with estimating the reduction in movement rate of a howitzer battery across the battlefield, one may choose to model only mobility kill and ignore fire power kill. In a more comprehensive study, both damage criteria may be modeled. A damage criterion that is of specific interest in discussing the following hypothetical case is the detonation of a secondary explosion source such as a stack of ammunition. Damage criteria may also be chosen such that they represent various levels of repairable combat damage.

(2) **Application to Hypothetical Case.** Manned with 215 personnel, the ASP maintains a total of approximately 3200 short tons of various kinds of ammunition. It is the primary source of conventional ammunition in the division sector. A major vulnerability of the ASP is the risk of secondary explosions of ammunition stacks as a result of a conventional munitions attack. In order to perform its mission, the ASP must have available forklifts and cranes to move the palletized ammunition from flatbed trailers to the 5 ton trucks used to deliver the ammunition to the user units. A major concern in the analysis of an ASP is that of the availability of the ammunition stacks and the materials handling equipment used to move it.

A simple application of the conventional lethality model is to model one level of incapacitation, total kill for example. Once this state of incapacitation is achieved, the item (a forklift, for example) can no longer be used. This may have a degrading effect on unit effectiveness by reducing the rate at which the ammunition stacks are received, stored and issued.

27. "Performance Characteristics of Artillery and Mortar Systems", Handbook Series G, No. 5, Army Materiel Systems Analysis Activity, 1985, (SECRET-NOFORN).

Perhaps it is known that there are certain types of damage that prevent the continued operation of the forklift until simple repairs are made. Damages of this type constitute a level of damage less severe than total kill, but may also degrade the unit by temporarily holding up the mission while repairs are accomplished. Modeling only the one level of repairable combat damage instead of total kill would result in a larger casualty estimate, and furthermore, no distinction between repairable damages and total kill would be made. This could result in the overestimation of unit degradation. However, one may specify both kill criteria by including the two sets of lethality data in the conventional lethality file. Comparing the lethal areas associated with two levels of damage, one would find that the lethal area associated with repairable combat damage is larger than the lethal area associated with total kill.

As shown in this example, in order to optimize the operations at the ASP it could be important to distinguish between the two levels of damage: repairable and total kill. Up to three levels of kill criteria may be modeled, two of which are repairable. The only requirement is that the lethal areas associated with the two repairable damage levels must be greater than the lethal area associated with non-repairable combat damage given the same probability of damage.

A limitation that one may find in trying to implement the use to three levels of combat damage is that of data availability. One technique that has successfully been employed to develop lethal areas corresponding to different levels of repair is to separate the system components into three groups.²⁸ These three groups represent those components which, when damaged, can be replaced or repaired by 1) the crew, 2) organizational assets or 3) direct support elements. These levels of repair are typically called light, medium and heavy repair. Although damage and repair levels are generic and may be user specified, the terms light, medium and heavy are used through out the conventional lethality subroutines to refer to the three levels where light refers to the largest lethal area and heavy to the smallest for equal P_d s. Items damaged such that they require repair by direct support elements are considered as total kill since they will not be returned to duty in a reasonably short period of time. A single lethal area can be derived for each repairable combat damage level based on the associated lethal areas of the individual components.

Another example of the use of the conventional kill criteria is the way in which the vulnerability of ammunition stacks are modeled. Two kill criteria which may be of interest include the detonation of a stack or fragmentation and

28. Juarascio, S.S., "Evaluation of an 8-GUN M109A2 Artillery Battery with Replacement of Combat Damaged Mission Essential Components(U)", BRL-TR-2682, October 1985, US Army Ballistic Research Laboratory, Aberdeen Proving Ground, Md., (SECRET).

blast damage which renders it unusable. The detonation of a stack is called a secondary explosion.

Once a secondary explosion is initiated, this reaction becomes a threat to adjacent ammunition stacks as well as nearby personnel and equipment. The secondary explosion threat is described in the AURA runstream as a weapon with a corresponding set of lethality data which describes its incapacitating or lethal effects. Simulated in this way, there is a probability that one secondary explosion may set off another and may even result in a chain reaction.

The conventional kill criteria described by the lethality data may be used to describe effects other than kills and damages. The conventional kill criteria is a convenient way to set up 'denial areas' around potential hazards such as secondary explosion sources. For example, the ASP has been modeled such that the area around a detonated stack is restricted from use for safety reasons. Equipment and supplies falling within this area are no longer usable unit assets. In order to free the area and equipment for use, a fire fighting task is initiated at the sight of the explosion through the use of the REPAIR mnemonic. Once the "repair" is completed, previously restricted ammunition supplies and equipment are returned to duty.

(3) Target Posture. Target posture refers to the positioning of each asset at the target. Each asset may be assigned its own unique posture such as specifying a person in the open, in a foxhole, in a forest or perhaps located inside of an armored vehicle. Posture changes may also be initiated at the time that an attack is modeled. These inputs describe some of the details of the deployment of the target and are input under the DEPLOYMENT mnemonic.

(4) Lethality Data File. The conventional lethality data file must contain data describing the lethality of each weapon system defined by the WEAPONS mnemonic against each asset type. The lethality description must include data for each kill criteria and each posture defined by the target description under the DEPLOYMENT mnemonic. Sources used to obtain lethal area data for conventional munitions typically provide lethal area data as a function of both kill criteria and posture.

(a) Application of Lethality Data File to Hypothetical Case. The ASP's 215 personnel are deployed in the open and in a temperate forest. In the event of incoming rounds, personnel change to prone position. These postures and posture changes are defined in the Deployment section of the runstream. However, the vulnerability of unit personnel in the various postures is described through variations in the lethal area and corresponding probabilities of kill used in the conventional lethality file. Likewise, equipment and supplies may also be deployed in the open and in the forest with the differences being described through variations in the corresponding lethal areas. One conventional posture change may be specified for each item in the DEPLOYMENT section of the

runstream. This change occurs at a stochastically determined time after an incoming attack and triggers AURA to look for the associated lethality data set in the conventional lethality file.

b. Triggering of Conventional Scenario. An AURA conventional lethality scenario can only be initiated by including the CONVENTIONAL LETHALITY DATA card in the AURA runstream file. The CONVENTIONAL LETHALITY DATA card causes AURA (via subroutine CONVIN) to open and read a conventional lethality data file named as FORTRAN unit 2.

The conventional lethality file inputs a weapon name as it appears in the WEAPON list under the NAMES mnemonic in the runstream file. Following the weapon name are a series of asset names from the ASSET list under NAMES. Under each asset name is a list of conditions to include: weapon heights of burst, asset postures, and kill criteria, followed by the parameters that give the vulnerability of the asset to the weapon under these conditions. The kinds of conventional weapons that have typically been modeled using AURA are high explosives (HE) rounds, bombs and artillery rounds such as mortars, and howitzer and multiple rocket launcher munitions. Improved conventional munitions may be modeled by treating each submunition as an individual round. Most conventional, indirect fire munitions are appropriate for use with AURA. Not compatible with the AURA methodology, as currently configured, would be weapons systems such as smart munitions, guided munitions or direct fire munitions. (Note: Smart munitions are currently being incorporated into the AURA methodology and will be documented in an addendum to this report.)

A conventional lethality scenario is a time dependent event in which each incoming round is analyzed by target point to determine the fractional damage incurred against all unit assets. The probability of kill is determined for each item at each target point for each incoming round. The survivor rule is then used to sum these probabilities to determine the items probability of surviving the current lethality event being analyzed. This information is stored and used in other subroutines in order to determine asset allocation, link effectiveness and, ultimately, unit effectiveness as a function of time.

c. Modeling Kills Deterministically Versus Stochastically. AURA analyses can be run in either of two modes: deterministic or stochastic. In this context, "stochastic" refers to analyses in which probabilistic phenomena are handled by making binary choices (0 or 1, killed or not killed), in such a way that, after many runs the average of the choices reflects the underlying probability distribution. Stochastic models generally involve the use of random number (Monte Carlo) techniques. (see section II.3 for a more detailed discussion of deterministic and stochastic modeling)

The original, and currently default, mode for AURA is deterministic. Actually, even the deterministic mode involves some stochastic features. For example, the selection of actual weapon burst points and the time each individual changes postures are stochastically selected.

(1) **Deterministic Mode.** In assessing kills deterministically, if a weapon hits a distance from point A such that the probability of killing a particular item is 0.6 then, upon the proper functioning of such a weapon, 60 percent of all such items located at point A will be killed.

Note that the fractional procedures involved in deterministic modeling can be compounded. Thus, if 0.4 of an asset is assigned to a job which had four possible locations (0.25 each) and a weapon causes a 0.3 probability of kill at one point, the computed loss at the point would be as follows:

$$(0.40)(0.25)(0.30) = 0.03$$

Although clearly involving significant assumptions, the deterministic mode has the outstanding advantage of requiring fewer replications for convergence.

(2) **Stochastic Mode.** Stochastic modeling of asset kill or damage may be implemented through the use of the MODE (STOCHASTIC, ON) option. As indicated above, the stochastic mode does not resort to fractional assets to account for various probabilities. Rather, assets are treated as entities and probabilities are used as probabilities. As such, the stochastic mode is more realistic. It is also more "instantaneous", in that it makes specific decisions (e.g. exactly where is the asset at this instant) rather than using average values.

In the area of asset kills, the stochastic mode proceeds as follows: When a round has hit, the probability of kill (P_k) for each asset is determined by the lethality routines (as in the deterministic mode). However, the whole asset is then either killed or not killed depending upon the value of a random number drawn from a uniform distribution in the range 0 to 1: if the random number is less than P_k , the item is dead, otherwise, it survives.

d. **Overview of the Conventional Lethality Methodology.** The input routines providing information to the conventional lethality subroutines may be separated into three categories: weapons inputs, unit deployment information and conventional lethality data. The handling of the conventional lethality data and the methods involved to determine the probability of kill of an unit asset is the subject of this discussion.

(1) **Weapons and Unit Deployment Information Needed by the Lethality Model.** In order to analyze the impact of a conventional lethality event on the target, the conventional lethality subroutines must know the burst point or actual ground zero of the incoming rounds and the location, conventional

posture and kill criteria of the assets being analyzed. This information is passed to the conventional lethality subroutines from the weapon and unit deployment subroutines.

Weapons information includes the actual ground zero (which has been calculated using the designated ground zero and any target location and delivery errors) and weapon name being analyzed.

Unit deployment information includes the x and y target location being analyzed; the conventional posture (in a foxhole, in an armored personnel carrier or any other quantifiable protective posture); and the conventional kill criteria (up to three levels may be specified). AURA assesses the impact of the conventional lethality event against each target point in the unit. The posture and kill criteria associated with each asset may vary depending upon its deployment. The vulnerability of the asset will vary depending upon its posture and the kill criteria of interest. Therefore, posture and kill criteria for each asset must be defined for all target points. The analyst is not limited to a specified list of postures or kill criteria in AURA. The only limitation is that imposed by the availability of data.

(2) Conventional Lethality Subroutines. The conventional lethality subroutines perform a variety of functions including the storage and retrieval of conventional lethality input data, the calculation of P_k at item level for each incoming round, and the calculation of P_k over all incoming rounds using the survivor rule, as well as a detailed accounting of all unit assets that are damaged and the type of damage. An account of the fraction of available assets is required at each user specified reconstitution event or lethality event. These data are used so that the commander may make decisions as to how to reallocate unit assets to optimize the unit mission.

Currently, in AURA, there are two kinds of functions which use the lethal area and determine probability of kill: the Carleton Von-Neumann and the cookie-cutter functions. Figure 24 illustrates an example of a symmetric cookie-cutter and a Carleton.

The cookie-cutter, sometimes called the step function, is simply made up of concentric ellipses, which may be either symmetric or asymmetric. The probability of kill is defined at specific distances along the range and deflection axes from the target point. Within the area specified by these semi-axes, the probability of kill is uniform. Up to three ellipses may be defined for each target type against each munition type.

The Carleton Von-Neumann function is an exponential function describing the change in the probability of damage as a function of distance from the burst point to the target center. This function may also be either symmetrical or asymmetrical, depending upon user inputs.

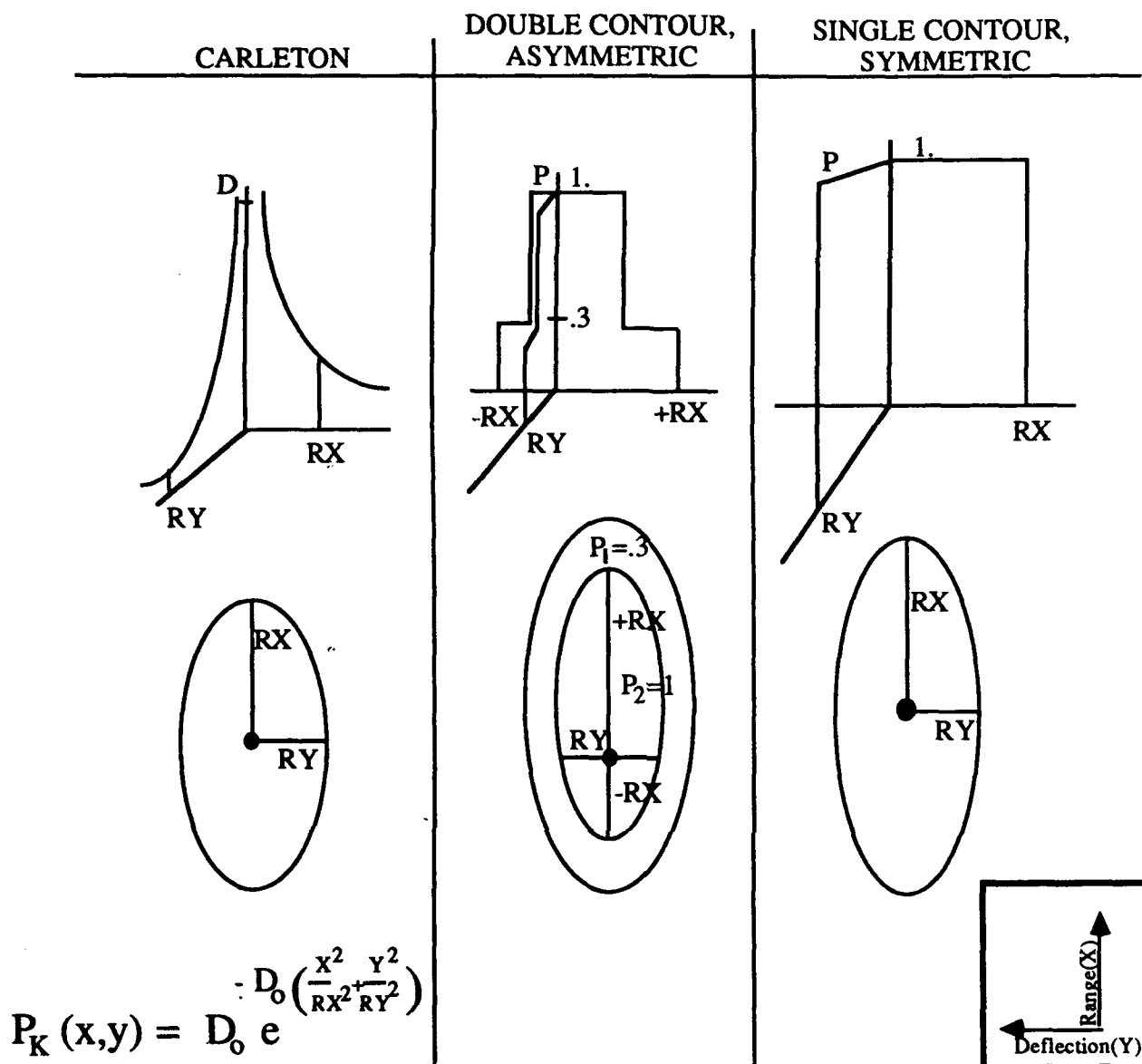


Figure 24. Examples of Conventional Lethality Footprints

In the conventional lethality file, each unit asset is described independently; it is possible to use a mix of the cookie-cutter and Carleton Von-Neumann lethality types discussed above. There are a total of eight possible data types:

- 1) symmetrical Carleton Von-Neumann;
- 2) asymmetrical Carleton Von-Neumann;
- 3) single symmetrical cookie-cutter;
- 4) single asymmetrical cookie-cutter;
- 5) double symmetrical cookie-cutter;
- 6) double asymmetrical cookie-cutter;
- 7) triple symmetrical cookie-cutter;
- 8) triple asymmetrical cookie-cutter.

Either one or three user specified lethal areas for user specified damage levels may be modeled in AURA. Items modeled with only one level of damage are not repairable. If damaged, the code removes these item from the commanders list of useable assets. Damage described with only one lethal area is called either heavy damage or total kill for the purposes of discussion here and in the subroutines themselves. However, it is really determined by the data that are used to describe it. The only modeling criterion that exists is that items which receive heavy damage are not returned to duty.

There are two criteria for items described with three levels of damage. Each level of damage describes a lethal area of increasing size, that is, the lethal areas are mutually inclusive. Damage can be repaired for only two of those levels (the two largest). The smallest lethal area describes heavy damage for which repair can not be modeled by AURA. For the purpose of discussion, these damage levels are referred to as light, medium and heavy where light corresponds to the largest lethal area, medium to the intermediate area and heavy to the smallest lethal area. However, the definition of these levels is dictated only by the data chosen to describe them.

The modeling of repairable combat damage significantly complicates the record keeping that takes place in CNVLTH. Probabilities of kill, corresponding to the three levels of damage, are modeled such that they are mutually inclusive. That is, the probability of light damage (P_l) means at least light damage has occurred and potentially medium or heavy.

e. Conventional Lethality Subroutines and Their Functions. There are four subroutines that make up the conventional lethality model: CNVLTH, CNVDMG, PKVNC and PKHECC. Subroutine MAIN calls EVNTDO which calls LETHAL which then calls CNVLTH. The subroutines discussed in detail in this section include CNVLTH, CNVDMG, PKVNC and PKHECC. These four subroutines handle the incoming lethality information, process it to determine P_k and report the fractional number of survivors. Unit assets are addressed item by item for each target point, for each incoming round at each user specified

reconstitution and lethality event.

(1) **Calculation of Conventional Losses/Damage.** CNVLTH is the first of the four lethality subroutines. It is called by LETHAL which passes three parameters: I, the identification of the asset whose damage is being analyzed; K, the target point of this asset; NN, the address in primary storage that tells how many of these assets are located at K. LETHAL calls CNVLTH from within nested do-loops which analyze each asset type at each target point for each incoming round in the lethality event being processed.

CNVLTH calls upon CNVDMG to provide the P_k calculations for the asset located at the given target point for the incoming round currently being analyzed. If the stochastic mode is turned on, it uses this P_k to determine whether or not ($P_k = 0.0$ or $P_k = 1.0$) the item was killed.

The bulk of the coding in this subroutine, however, is devoted to record keeping. CNVLTH performs a massive amount of record keeping to track the number of available items and damaged items at the target point. It stores target posture and kill criteria of the current asset being analyzed and stores the conventional lethality results. The code accounts for all items receiving light and medium damage for processing in the repair subroutines. Before returning to the calling routine, CNVLTH uses the "survivor rule" to determine the asset's probability of surviving all rounds processed to that point in the the lethality event.

Since three levels of damage may be modeled, the code must assess each one independently. First, it determines whether or not light damage has occurred (P_l). Recall that the lethal area assigned to the level of damage called 'light' is the largest of the three damage levels. Therefore, if light damage has not occurred, there is no reason to further analyze the probability of medium (P_m) or heavy (P_h). If light damage has occurred, the code must continue to analyze the probability of medium damage. If medium damage has occurred, the code must also analyze the probability of heavy damage. These probabilities are defined such that:

$$\begin{aligned}P_l &= P_l - P_m - P_h \\P_m &= P_l - P_m \\P_h &= P_h\end{aligned}$$

The bookkeeping that takes place in CNVLTH also identifies items that are secondary explosion sources. Secondary explosion sources must also be analyzed as a threat against all other unit assets. When CNVLTH encounters an asset that has been identified as a "secondary", it simply adds the name of this asset to an array of weapons to be processed through the conventional lethality subroutines. Conventional lethality data describing the effects of a secondary explosion on unit assets are input into the conventional lethality input file in the same manner as other weapons effects data. The stochastic mode of modeling is also available for the simulation of secondary explosions.

To calculate the conventional losses or damage for a specific asset at a specific target point, CNVLTH calls the subroutine CNVDMG which determines the P_k . Using the current P_k information and P_k calculations stored from previous calls to the conventional lethality subroutines for asset I, CNVLTH calculates asset I's probability of surviving (P_S) over all incoming rounds for the lethality event being processed.

The survivor rule is as follows:

$$P_S(N) = \prod_{i=1}^N (1.0 - P_k(i))$$

where

- N total number of incoming rounds.
- $P_S(N)$ probability of survival against N rounds.
- $P_k(i)$ probability of kill from individual round
 currently being analyzed.

The survivor rule is derived from the concept of statistical independence. An independent probability function is basically defined as a function having the property that the joint probability of events equals the product of the probabilities of the separate occurrences. As applied to the problem of determining the survival of an asset to a conventional lethality event, the joint probability of events, $P_S(N)$, is simply the product of the probabilities of surviving the individual incoming rounds.

(2) Calculation of Probability of Kill. CNVDMG is called by CNVLTH which passes two parameters: INDXWF, which is a parameter containing information on the size of the vulnerability data array and INDWWF, which is a pointer into the vulnerability data storage array. CNVDMG uses INDWWF to read the conventional lethality data, determine the data type (one of the eight types discussed above), and the kill criteria and posture of asset I in order to determine the P_k of the item being analyzed.

If the proper data is not located an error message is flagged and the AURA run is terminated. Once the vulnerability data is read from the primary storage array, CNVDMG calls one of two subroutines to make the actual P_k calculations. PKHECC is called if the conventional data type is a cookie-cutter. PKVNC is called if data type is in Carleton Von-Neumann format.

(3) The Carleton Von Neumann Algorithm. The primary function of the PKVNC subroutine is to calculate the P_k for asset I based on the Carleton Von-Neumann function, a bi-variate Gaussian distribution:

$$P_k(X, Y) = D_0 e^{-D_0 \left(\frac{X^2}{RX^2} + \frac{Y^2}{RY^2} \right)}$$

where $X = X_T - X_B$, $Y = Y_T - Y_B$,

D_o = maximum P_k at center of target ;

X_T = X coordinate of the target location ;

X_B = X coordinate of the burst point ;

Y_T = Y coordinate of the target location ;

Y_B = Y coordinate of the burst point ;

RX = parameter defining extent of ellipse in the x direction ;

RY = parameter defining extent of ellipse in the y direction .

The values D_o , RX and RY are user inputs which define the shape of the probability damage function. These inputs may be determined from the lethal area (A_L) and the ratio RY/RX . The ratio RY/RX may be determined from the curve given in Figure 25 (this figure was extracted from 61 JTCG/ME-81-8) for a specific munition fall angle. Values for RY and RX may be calculated for a specific lethal area using the relationship, $A_L = (\pi)(RX)(RY)$.

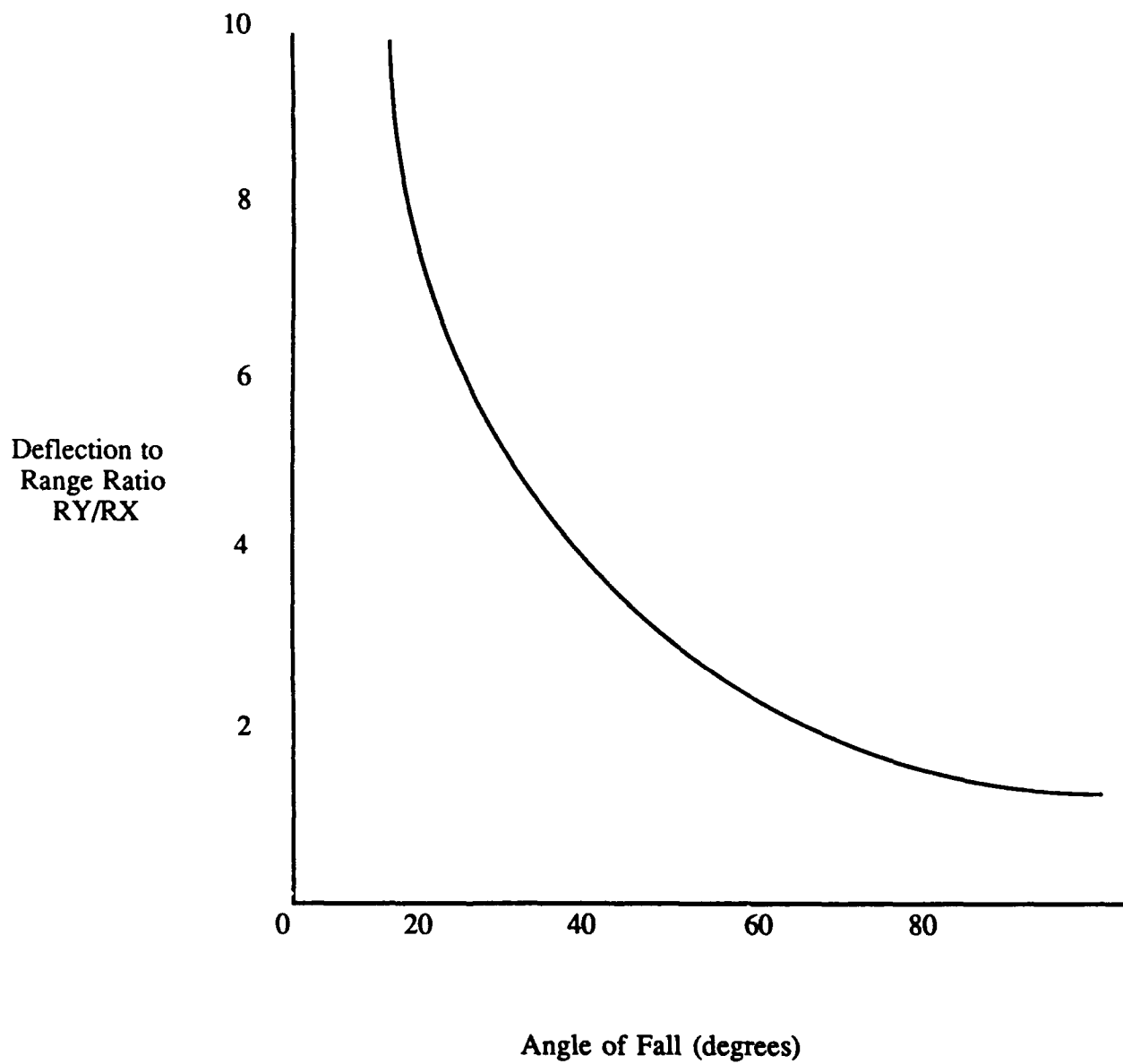


Figure 25. Deflection to Range Ratio (R_Y/R_X) versus Fall Angle for HE Munitions

Typical values for the parameter D_0 are 0.20 for the proximity fuze high explosive (HE) round, 0.25 for a point detonating HE round and 1.0 for an improved conventional munition (ICM).²⁹ Note that ongoing and future analyses may provide new estimates.

(4) **The Cookie-Cutter Algorithm.** The primary function of the PKHECC subroutine is to calculate the P_k for asset I based on the cookie-cutter or step function methodology. The cookie-cutter may be symmetrical or asymmetrical and contain up to three contours.

In the cookie-cutter methodology, the distance from the burst point to the target location is rotated into the incoming weapon (range-deflection) coordinate system. The rotated values, along with the vulnerability data, are substituted into the elliptical equations to determine the probability of kill or damage. The rotation is made as follows:

$$X = (XT - XB)\cos(\text{inc}) + (YT - YB)\sin(\text{inc}) ;$$

$$Y = (YT - YB)\cos(\text{inc}) + (XT - XB)\sin(\text{inc}) ;$$

where

inc = angle of incoming fire ;

XT = x coordinate of target point ;

XB = x coordinate of burst point ;

YT = y coordinate of target point ;

YB = y coordinate of burst point .

For the case of an asymmetric contour, the code must check the position of the round against the contour to decide whether or not the round landed in front, or in back, of the target to determine which value of RXX to use in the next equation. The position of the burst point in relation to the target is determined via the following formula:

$$\frac{X^2}{RXX} + \frac{Y^2}{RAY} > 1.0,$$

If this condition is true, the round landed outside the contour. (RXX and RAY

29. "Simplified Artillery Projective Effectiveness Model - Artquick Computer Program, User Manual", 61 JTCG/ME-81-8, Oct 1980, Unclassified.

are the elliptical axes of range and deflection which have been determined by on lethal area data.) Otherwise, the round landed inside the contour and the probability of kill is set to the P_k defined for the contour.

The cookie cutter methodology may be used to provide a gross estimate of the fractional damage to unit assets when insufficient lethality/vulnerability data are available. The cookie-cutter is typically used to provide a good estimate of the probability of kill due to blast when blast is the predominant effect of the munition. The Carleton function would typically be used when trying to model the far-reaching effects of the fragmentation of a munition. The predictions of the cookie-cutter most closely match those of the Carleton against hard targets when delivery errors are high.

The JTCG Full Spray³⁰ model which provides a map of P_k s in a grid formation around the target can be used to estimate the probability of kill in concentric, cookie-cutter form. The BRL has an adaptation of this model which makes these estimates and calculates the necessary input for AURA.

The one-step, circular, cookie-cutter is the simplest mathematical form for describing the probability of kill given only one parameter, $P_k(X)$, at a given distance. This simple case assumes a constant kill probability in the circular region within the damage radius X .

f. Application of Hypothetical Case to the Conventional Model. As a summary of the conventional model consider the special case of the ASP where it is necessary to characterize the vulnerability of ammunition stacks. For the baseline characterization, let the probability of secondary explosion given a hit be 1.0 and the denial radius around an exploded stack be 500 meters. The denial of ammunition stacks resulting from the occurrence of a secondary explosion within 500 meters may be treated as a type of repairable damage using the repair methodology. Once a secondary explosion occurs, the code allocates personnel to the task of fire fighting. For this example, let the fire fighting team require 20 personnel which are chosen from the units 215 such that the degradation to the unit is minimized.

The measures of effectiveness (MOEs) for this scenario include the expected number of secondary explosions, the expected number of denied ammunition stacks, the expected unit effectiveness over time, and the identification of the weak link(s) in unit effectiveness.

30. "Computer Program for General Full Spray Materiel MAE Computations, Volumes 1 and 2", 61 JTCG/ME-79-1-1, January 1979, (UNCLASSIFIED).

Suppose that the weakest link in the unit is the availability of ammunition stacks, one might ask how reducing the size of the denial radius would reduce the expected number of denied ammunition stacks and increase unit effectiveness. To perform this sensitivity analysis, one would vary the size of the denial radius by varying the associated axial lengths of the lethality function over a set of values such as 0, 100, 200, 300, 325, 350, 375, 400 and 450. The results of these excursions may be used to show availability of ammunition stacks and unit effectiveness as a function of denial radius such as illustrated in Figure 26. Such results may indicate a sensitivity to denial radius only over a certain range, as in this hypothetical example, between 0 to approximately 700 meters.

One may also wish to examine increases in unit effectiveness as a function of reducing the vulnerability of the ASP, in particular, reducing the probability of secondary explosion given a hit. Again, a set of excursions could be made (holding baseline denial radius at 500 meters) such that the probability of detonation is varied from 0.4 to 0.9 (the baseline gives the MOEs where $P_d = 1.0$). Additionally, based on the denial radius excursions, it is agreed that denial radius should be reduced to 350 meters if feasible, but it is not yet known how this would impact upon the survivability of the unit. Then, a third set of excursions may consist of holding the denial radius at 350 meters, and again varying the probability of detonation given a hit from 0.4 to 1.0.

This is a limited example to show how the conventional lethality methodology may be applied in an analysis. Conventional lethality data are usually based upon a lethal area which is the measure of a warhead's ability to incapacitate a target. As shown in this example, the methods applied to accomplish this may also be extended to simulate such events as denial of unit assets and the ability of a stack of ammunition (or other secondary explosion source) to incapacitate another unit asset or cause another secondary explosion.

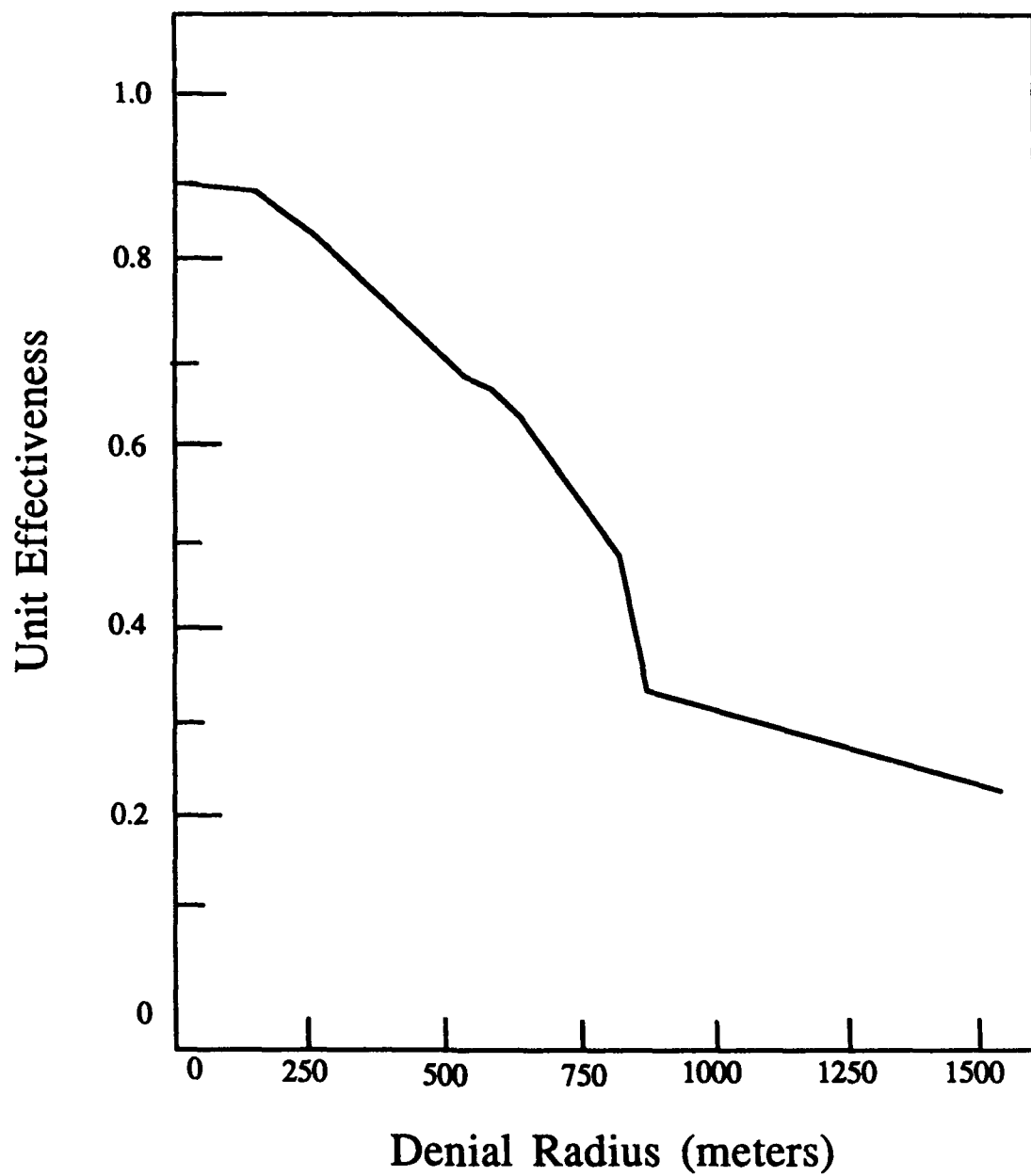


Figure 26. Example of Unit Effectiveness as a Function of Denial Radius

3. Chemical Lethality Model

a. **Triggering a Toxic Scenario.** A chemical scenario is triggered in the AURA model by the employment of a weapon identified as "TOXIC" in the NAMES input. (Note, for the purposes of this discussion, the words "toxic" and "chemical" are used synonymously.) All weapons so identified must have associated chemical dispersion data on a chemical dispersion file. The TOXIC DISPERSION DATA card causes AURA (via the subroutine TOXIN) to open and read a chemical dispersion data file via FORTRAN unit 4. The standard code used to generate these data for use in AURA is the Non Uniform Simple Surface Evaporation (NUSSE) model developed by the U.S. Army Chemical Research, Development and Engineering Center (CRDEC).^{31 32} A separate utility program, called PRETOX, was developed to extract from a NUSSE output the chemical dispersion data required by AURA. Appendix B of Volume 2 of this report discusses these codes in more detail.

b. Overview of the Chemical Lethality Model.

(1) **Inputs.** The chemical lethality model requires inputs in the following five areas: weapons, unit deployment, toxic kill criteria, dispersion file (unit #4), and degradation values for sublethal chemical doses. Another input of interest, which is optional, is chemical alarms. This section will briefly discuss these six areas and how the vulnerability of assets to chemical weapons is assessed.

(a) **Weapons.** The weapon inputs required to assess chemical lethality are the weapon name and type (i.e., "toxic"), agent type, and all user inputs. The weapon name and type are used to identify the weapon as a chemical weapon throughout the remainder of the runstream. This information is used by various chemical subroutines to determine such data as the contamination pattern and chemical agent persistence time. Agent type identifies one of four agent types: G, V, T (thickened GD), or H, with G being the default type; this, in turn, identifies which of the dose response curves will be used when assessing the effects of the agent against personnel.

31. Saucier, R., "NUSSE3 Model Description", U.S. Army Chemical Research, Development and Engineering Center, CRDEC-TR-80746, May 1987, (UNCLASSIFIED).

32. Saucier, R., "NUSSE3 User's Guide and Reference Manual", U.S. Army Chemical Research, Development and Engineering Center, CRDEC-SP-86009, March 1986, (UNCLASSIFIED).

(b) **Unit Deployment.** The x and y coordinates of the target point, asset identification, how many assets at the target point, the chemical posture and the toxic kill criteria are the unit deployment inputs of concern in the chemical arena. The x and y coordinates of the target point are used to locate the target point relative to other assets within the unit. In addition, for chemical, these coordinates are used to locate the target point relative to the chemical cloud pattern location. The toxic kill criteria code describes the job category of personnel located at that target point; it is defined under the T.K.C (toxic kill criteria) mnemonic. (This will be discussed in the next section.) Chemical posture describes the level of chemical protection assumed by the personnel asset at a target point. Default values range from 0 (no protection) to 4 (full protection). In addition, the user has the option to create as many as four additional chemical postures.

(c) **Toxic Kill Criteria (T.K.C.) Code.** T.K.C. is a job related mnemonic which associates a code number with a verbal description of the job and a chemical dose multiplier used to simulate higher (or lower) than normal ratio of dose, as would be acquired by a person whose task required a higher (or lower) than normal breathing rate. This mnemonic is normally used to associate similar jobs or tasks performed by the unit with a chemical dose multiplier as well as a MOPP degradation value. This input also allowed specifying heat stress parameters to each job category (the heat stress algorithm has subsequently been updated and will be discussed in an addendum to this report). The T.K.C. code number is used to indicate deployment points at which a difficult or easy job is being done. Any individual assigned to that deployment point inherits the chemical-related difficulties of the job as portrayed by the associated heat stress, MOPP degradation, breathing rate, etc.

(d) **Chemical Dispersion File (unit #4).** The chemical dispersion file, generated by the NUSSE and PRETOX models, contains up to four sets of data; this includes the contamination outline, the grid of contamination density, a time series of dosage grids and a concentration data grid. The number of data sets required depends upon the threat environments (contamination, percutaneous, vapor) to be included from each warhead. The first set of data, the contamination outline, gives the width, arrival and evaporation time of contamination as a function of downwind distance. The second set, also taken from the liquid phase of the dissemination calculation, is a grid of contamination density, normalized to one lethal percutaneous dose, for crosswind displacements as a function of downwind distance. The third set is a series of dosage grids, one for each selected time point, with each grid containing an entry for accumulated dosage at each crosswind displacement as a function of downwind distance. In the third set, all entries are normalized to one lethal inhalation dose. (NOTE: The normalizing values are user specified. The default normally used by BRL AURA analysts is one lethal dose; other values can be specified, such as incapacitation.) The fourth set is an X, Y grid of concentration peaks and arrival times. For each X,Y point, three values are given: the peak concentration that will occur at the

point, the time when the vapor first arrives, and the time that the peak concentration occurs.

(e) Degradation Inputs for Sublethal Chemical Doses.

AURA permits association of specified jobs (links) with particular dose-time degradation sets in order to degrade the performance of chemically dosed individuals within the unit. Degradation is modeled in AURA as a function of dose and time. Currently, six degradation matrices are built into AURA, representing three agent types (G, V, and H) and two jobs. One job describes a gunner, which is considered typical of most cognitive jobs while the second job describes an ammunition loader, typical of a physically demanding job. These matrices are set in the DGOSET subroutine with the default sets being the gunner job. The user may change the degradation of specific links from the default gunner matrices to the ammunition loader matrices or input his own. These changes are accomplished with the SUBLETHAL DOSE DEGRADATION mnemonic. New matrices are read from unit #11.

(f) Chemical Alarms. An optional input in the AURA model is chemical alarms. Alarms are used to trigger the start of the MOPP clock and the change by unit personnel into alternate MOPP postures. The default within the model is for personnel to automatically change MOPP upon any incoming round. If the user wishes to change this to MOPping only upon the activation of a chemical alarm, a number of options must be set. First, alarms must be deployed in the same manner as other unit assets. The ALARM mnemonic is then used to identify chemical alarms and to indicate the threshold value required for activation due to each chemical weapon; alarms can be activated by either vapor concentration or vapor dosage. (The calculation of alarm activation time will be discussed later in the section detailing chemical related subroutines.) Finally, ROUND, OFF must be set under the MOPP mnemonic to indicate that personnel will not automatically change postures upon the arrival of a round.

(2) Definition of the Term - Casualty. For the purposes of this discussion, the term casualty refers to an individual removed from the analysis as a result of exposure to a chemical agent(s). Internally, AURA uses the normalized dose of one as the dose at which 50% of the personnel would become casualties. Thus, the absolute amount of agent used as the 50% casualty level is specified by the user when normalizing the toxic dispersion input file (via PRETOX).

(3) Overview of the Chemical Lethality Methodology. The chemical lethality methodology is used to assess the effects of chemical agents on unit personnel and equipment. The purpose of this section is to provide the reader with an understanding of the flow of the chemical lethality methodology and how it is applied to personnel and equipment. The next section will discuss the actual subroutines in detail.

The process of calculating the effects of chemical agents begins with the arrival of one or more chemical weapons. Calculation of the dispersion of the chemical agent is done in two parts. First, upon the arrival of the round(s), data are stored which are then used in dispersion calculations. Subsequently, effects which are time dependent are reassessed at every reconstitution time or lethality event during the simulation.

It is important to note that two actions occur prior to the assessment of the effects of the chemical. The first is the start of the "MOPP clock". If the default is used for MOPP time, then the MOPP clock is started upon the arrival of any round, not just chemical. Consequently, by the time the actual chemical subroutines are called, the MOPP clock has already begun. The other way to start the MOPP clock is through the use of chemical alarms, as previously discussed.

The second action is the determination of the cloud travel time. Once the inputs have been read in and processed by the AURA input evaluation subroutines, AURA knows if a chemical attack will occur within the simulation (from the ROUND and/or VOLLEY mnemonics). As a result, subroutine CLDTIM is called by MAIN to precalculate the chemical cloud travel time. This time represents the time it takes for a vapor cloud to drift downwind a distance equal to the maximum extent of the unit area. The persistence time is calculated by adding the maximum evaporation time and the secondary vapor hazard drift time. However, since there is no secondary vapor hazard for a pure vapor agent, persistence time for such agents is merely the drift time for the primary vapor.

The ensuing discussion will deal with personnel and equipment assets separately.

(a) **Personnel.** Upon the arrival of a chemical round, LETHAL is called by EVNTDO to store preliminary chemical data. For a percutaneous hazard, LETHAL calls CNTMST to calculate the contamination arrival time and the amount of liquid for each target point. These data are stored in TGTCNT(K,1) and TGTCNT(K,2), respectively. If it is a pure vapor hazard, LETHAL simply stores the needed weapon arrival data since the chemical effects will be evaluated by the time dependent subroutines (e.g. TOXLTH).

If the next event is another lethality event (hence the arrival of another round), the above process is repeated if the round is chemical. If the next event is a reconstitution or other event, EVNTDO will first update all time dependent factors to the current event. For a percutaneous hazard, EVNTDO calls CNTM. Using the data already calculated by CNTMST, CNTM calculates the dosage based on the amount of contamination present and the personnel asset's MOPP posture and penetration factor. This value is stored in TGTDOS(K,2). Similarly, for vapor, EVNTDO calls TOXLTH which, in turn, calls DOSAGE. DOSAGE evaluates the dose received up to the current time or the associated target point MOPP time, whichever occurs first; this value is then returned to TOXLTH.

TOXLTH takes this vapor dosage value and adds it to TGTDOS(K,2) which already holds the percutaneous dose. TGTDOS(K,2) now represents the total dose, both percutaneous and vapor, received by the personnel asset at that target point. (NOTE: This portion of the methodology has been updated for route-of-entry effects and will be documented in an upcoming BRL report.)

At this point, EVNTDO calls RECEVN for the processing of the current event, if it is a reconstitution event. RECEVN first calls CUMDOS which calculates the total dose received up to that point in time by assets at each target point. This total dose represents the summation of the asset's dose from the current bin plus any new dose they have received as stored in TGTDOS(K,2). For each target point containing personnel assets, CUMDOS divides the number of personnel of that asset type present by the total number of surviving assets of the same type. This is the fraction of that asset type assigned the new dose. CUMDOS then assigns these dosed assets to bins which are indexed in ascending order with respect to dose amount. Assets receiving more than the maximum dose, indicating immediate casualties, are stored in one bin and given an arbitrarily high dose value. This high dose value will effectively remove them from any future substitution considerations.

Before attempting substitution, RECEVN calls TOXDTH to calculate any time dependent deaths. TOXDTH checks personnel assets in each dose bin. TOXDTH first determines which agent caused the dose. It then calls PRBTOX to calculate the effective dose multiplier, which uses the "probit" (inverse Bliss) slope to account for variations in sensitivity to chemicals across the population. Once the effective dose is known, PZPTOX is called to calculate the asset's time of death. If this time is less than the current time, then this personnel asset is assumed dead and the bin is flagged to indicate assets in this bin are dead; if the time to death is greater than the current time, the personnel asset is not dead and TOXDTH moves to the next bin.

Subroutine OPTMIZ is then called to perform the optimization process. During the optimization, subroutine LNKOPT processes personnel assets in the dose bins. This is accomplished through a call to DGRTOO. DGRTOO checks the bins with the least dosed personnel in an attempt to find one for a link optimization. DGRTOO stops at the first bin it encounters with a needed asset type. When DGRTOO finds an asset, DGDSTX is called to calculate the asset's effective degradation. DGDSTX calls PRBTOX to calculate the effective dose multiplier which is multiplied by the actual dose to give the effective dose. DGDSTX also finds the amount of time the asset has retained the dose. With this information, an upper and lower bound are placed on the chemical dose and time of dose and an interpolation made between these bounds to determine the degraded performance value of the asset. Finally, when a substitution decision has been made, LNKOPT calls DGRTOX to check the bins again and subtract the needed amount of the asset for link substitution.

(b) **Equipment.** Handling of contaminated equipment within AURA is a fairly straightforward matter. Upon the arrival of a chemical weapon, EVNTDO calls CNTM to determine contaminated equipment assets by target point. When CNTM finds an equipment asset which is contaminated, CNTM finds and stores the equipment's return to duty time due to natural evaporation. The contaminated equipment is then placed in the pool of available assets (if contaminated usage is allowed) or the "junkyard" with its associated return to duty time. During each subsequent lethality or reconstitution event, CNTMEV is called by SRVUPD to check evaporation times for contaminated equipment. If CNTMEV finds equipment available due to evaporation, it restores the item to the pool of available assets. (Note, items can also be returned to duty due to decontamination. This topic is discussed within the repair modeling section.)

c. **Chemical Lethality Subroutines and Their Function.** This section will discuss, in detail, the subroutines applicable to the assessment of chemical weapons and their effects.

(1) **Calculation of Alarm Activation Time.** Subroutine ALRM is called by LETHAL to find the sound-off time for alarms. ALRM first determines if the alarm is in range of the vapor concentration or vapor dosage from the incoming weapon. If the alarm is in range and is activated by vapor dosage, ALRM finds the dosage versus time at the alarm. If the dosage exceeds the activation dosage, ALRM uses interpolation to determine at what time the alarm was activated. Since this time is relative to burst time, ALRM then converts it to clock time for use in the simulation. ALRM also handles vapor concentration sensitive alarms. First, ALRM finds the vapor concentration values at X, Y points that bracket the alarm location by calling subroutine BRCKT2; these values are used to calculate the X and Y values to be used in the vapor concentration interpolation. ALRM calculates the peak (largest) vapor concentration level of the agent. If this peak value exceeds the alarm threshold, it then finds the times at which the vapor concentration is zero and the time it reaches the peak vapor concentration level. These values are used by ALRM to calculate the time at which the threshold vapor concentration level was reached, that is, the level required to activate the alarm.

(2) **Calculation of Liquid Contamination (per Target Point).** CNTM is the chemical subroutine which handles contaminated equipment and calculation of personnel percutaneous doses. In addition to the main subroutine, CNTM contains two entries, CNTMST and CNTMEV. Entry CNTMST sets the contamination arrival and evaporation time while entry CNTMEV checks the evaporation time in relation to contaminated equipment return to duty and personnel unMOPping times. CNTM is called by three subroutines: EVNTDO, LETHAL, and SRVUPD; CNTM itself calls two subroutines: BRCKT2 and UNPK6.

(a) **Main CNTM Subroutine.** CNTM loops through all target points checking for contamination of assets (ignoring protected equipment.) CNTM evaluates all target points to determine if contamination is present. For equipment, if contamination is found, it finds the reconstitution time at which the equipment will be usable due to evaporation. Recall, the evaporation time of the agent is one of the parameters input in the chemical lethality file (unit #4). This time is compared to the times specified in the simulation to determine if the item will be usable within this encounter. CNTM then checks all asset types against the homelink job being done at this target point. Three possibilities exist. One, the asset type cannot substitute in which case CNTM moves on to the next asset type. The remaining two possibilities are 1) the asset is a homelink or 2) the asset is a substitute. In either case, the asset is then checked to determine if it was contaminated earlier in the simulation (i.e. recontaminated). CNTM tallies those that have been recontaminated and subtracts those from the total in order to determine the newly contaminated items. All contaminated items, both newly contaminated and recontaminated, at this target point are stored in the junkyard with their decontamination time due to evaporation stored in the appropriate reconstitution time bin. Once the contamination data is tallied, CNTM updates equipment damage status in the FIXJNK array and stores the total of newly contaminated and recontaminated items. After this information is stored in the appropriate location, the target point is cleared of contaminated items. Evaporation time for the chemical agent is then set to the larger of either the evaporation time for contamination on the asset or the evaporation time of the chemical agent.

For personnel, CNTM checks to see if the time at which personnel will change protective postures is greater than the arrival time of the agent. (Recall, this pertains only to liquid (percutaneous) doses). If it is, personnel are assumed to be in their original MOPP posture; otherwise, personnel are assumed to have changed into their alternate posture. CNTM calls subroutine UNPK6 to find the individual's original or alternate MOPP posture and the associated penetration factor. CNTM uses this data to calculate the target point's dose and time of dose.

(b) **Calculation of Contamination Arrival and Evaporation Time.** Entry CNTMST sets the contamination arrival and evaporation times; effectively, this is the contamination lethality routine. CNTMST checks to see if the target point falls within the contamination grid by evaluating whether the contamination grid brackets the target point in both the downwind and crosswind directions. If the target point is located within the grid, CNTMST determines what is located there by evaluating the array ITAR. If $ITAR > 0$, then an asset is located at this target point. If the asset is equipment, then the persistence factor is determined. If $ITAR = 0$, it means a dummy link for personnel is at this target point and the persistence factor is set to zero as it is irrelevant in this case. Finally, if $ITAR < 0$, a dummy link for equipment is located at this point and the persistence factor is calculated in case the link is ever used. For

personnel, the current protective posture is determined as well as the penetration factor; a call to subroutine UNPK6 supplies this information. CNTMST then calculates the accumulation of contamination from this weapon, adds it to the total and stores the result in the array TGT CNT. This value is used in the main CNTM subroutine to calculate personnel dose if posture change has not taken place. Finally, CNTSMT stores the agent type in array TGT DOS.

(c) **Calculation of Evaporation of Contaminant.** Entry CNTMEV checks the evaporation time of the contaminant. It restores items placed by subroutine SRVUPD into the array FIXJNK (equipment damage status array). (SRVUPD is the subroutine responsible for putting all contaminated but usable items into the junkyard.) CNTMEV checks FIXJNK to see if all contaminated items are gone as a result of natural decontamination or deliberate decontamination; it does not check those items in repair for decontamination. Personnel are allowed to unMOPP when the last contaminated item is in repair (meaning FIXJNK is empty) or if the persistence time has elapsed.

(3) **Computation of Travel Time for a Chemical Cloud.** Subroutine CLDTIM computes the time it takes for a chemical vapor cloud to travel downwind of the unit. This time is used to model the amount of time that the unit would be required to stay in MOPP.

CLDTIM determines the length of time that the vapor hazard remains within the unit boundaries. It is assumed that the secondary vapor hazard will remain until blown downwind of the unit; this is referred to as the "secondary vapor drift time".

Drift time is equal to the maximum extent of the unit divided by the wind speed. The persistence time is calculated by adding the maximum evaporation time and the drift time for the secondary vapor hazard. However, since there is no secondary vapor hazard for a pure vapor agent, persistence time for such agents is merely the drift time of the primary vapor.

(4) **Initialization of Chemical Degradation Values.** DGOSET is called from DEFAULT to set the chemical default degradation matrices for cognitive and physically demanding tasks. There is a set for each of the three different agent types (G, V, and H as previously discussed above). Also, the time at which performance goes to zero and stays there is established in this subroutine. Finally, DGOSET checks for user supplied degradation matrices.

(5) **Evaluation of Degraded Performance Due to Chemical Dose(s).** DGDSTX is called by two other subroutines, specifically, DGRTOO and DGRTOX. Calling parameters for DGDSTX are the pointer to the dose bin being evaluated (IJ) and the link being evaluated with the degraded performance value returned to the calling subroutine. Subroutine DGDSTX evaluates the degraded performance of a job (link) as a function of chemical dose and time. This

subroutine first calculates the effective chemical dose through a call to PRBTOX. A factor which adjusts the actual dose (depending on how "chemically rugged" an individual is) to the effective dose is retrieved from the PRBTOX subroutine. This adjustment shows that individuals less "rugged" than normal will perform as though they have received a larger dose (vice versa for more "rugged" individuals). (It should be noted here that the adjustment discussed is related to the dose-response probit curves used by AURA and not the dose multiplier mentioned earlier which reflects job-dependent breathing rates.) Then, calling BRCKT2, lower and upper bounds (taken from subroutine DGOSET) are placed on the chemical dose and time values in this subroutine. Finally, a linear interpolation between these bounds is performed to determine the degraded performance value.

(6) Removal of Assets from Dose Bins. DGRTOX is called from LNKOPT as the latter tries to fill a job with a particular asset. There are six parameters for DGRTOX: functional group (IFG); the job linkname (JLK); the amount of the functional group needed (AVL); the current level of interest for operant construct (i.e., link subchain, etc.); the effective degradation of AVL of IFG; and, an alternate return. The subroutine starts with the least dosed bin (since these individuals are generally the most effective) to see if there is an available amount of the asset. DGRTOX goes through each bin, from least dosed to most dosed, subtracting the amount of the asset found, until it reaches the required amount. DGRTOX calls subroutine DGDSTX to calculate the degradation of the amount of the asset found in each bin. Once DGRTOX has reached the required amount of the asset and has determined the degradation of each fraction found in the various bins, DGRTOX calculates the effective degradation (weighted since assets from different bins would have different doses) of the required amount of the asset. The effective degradation is calculated by summing the amount of asset from each bin times its associated degradation and dividing this summation by the total amount of the asset required.

(7) Calculation of Vapor Dosage (per Target Point). Subroutine DOSAGE is called by both ALRM and TOXLTH to calculate the chemical dosage at a target point from a given weapon for a given time interval. The parameters for DOSAGE are: weapon; the displacements of the target point - crosswind and downwind; first time point; second time point; and, dosage. Dosage is determined by first calling BRCKT2 to find values which bracket the crosswind, downwind, and the two time points of interest. The data used by BRCKT2 come from the chemical lethality file and represent the cumulative dosage over time from the weapon. BRCKT2 is called three times to calculate the two values bracketing the target point in terms of crosswind, downwind, and time. Using these data, DOSAGE performs a linear interpolation to calculate the dosage received at the target point. DOSAGE checks to make sure that the values bracketing crosswind, downwind and the time interval fall within the crosswind, downwind, and times arrays from the chemical footprint. If at least one does not, the value returned by DOSAGE is set to zero as no dosage would be received. Note, the one case where this does not hold true is if the time interval

is greater than the times array; in this case, the dosage is set to the maximum dosage value.

(8) **Change of Personnel MOPP Posture.** Subroutine MOPP changes the target point's MOPP from the original to the alternate posture. In addition to the main subroutine, MOPP also contains two entries, MOPPST and UNMOPP. Entry MOPPST sets the time for the change while entry UNMOPP resets all the MOPP times and posture flags. MOPP can be called by two subroutines, EVNTDO or TOXLTH; in turn, MOPP calls three subroutines: NORMAL, UNIFRM, and UNPK6.

(a) **Main MOPP Subroutine.** MOPP first checks to see if the target point either does not change MOPP or has already changed; if either case is true, MOPP skips this point. If these states are not true, then MOPP checks the target point to determine its alternate MOPP posture. This is accomplished through a call to UNPK6 which unpacks the array containing the original and alternate posture of the target point.

(b) **Calculation of Posture Change Time.** Entry MOPPST sets the times for the chemical posture change. It checks to see if the call to change postures was a result of a chemical alarm or an incoming round. If it is an alarm, the time to start the posture change is set to the time the alarm sounds. If an incoming round signals the change, the time to change is set to the current event time. MOPPST then checks to see if the round was within range to trigger the posture change. This is done by comparing the minimum and maximum X and Y coordinates of the lethality event plus any proximity value with the X and Y coordinates of each target point. The proximity value is set through use of the PROXIMITY sub-mnemonic within the MOPP mnemonic. This value represents the distance from the warhead within which an asset will feel threatened by an incoming round and change MOPP posture; the default value is infinity. Subroutine NORMAL is called to generate a normally distributed random number which is then used in the determination of the time to change postures. Entry MOPPST retrieves, from the storage array, the kill criteria of the target point and uses this information to determine if the target point has heat stress parameters associated with it. If it did, subroutine UNIFRM is called to generate a uniformly distributed random number. This random number was drawn against the probability values to determine at what time the target point would become a heat stress casualty. (The algorithm for heat stress has been updated in the latest version of AURA and the changes will be detailed in an addendum to this report.)

(c) **Reset of Personnel to Original MOPP Posture.** Entry UNMOPP resets all the MOPP posture flags and then resets all the time related variables to infinity. This includes the time to change, the time to become a heat stress casualty and the time the alarm will sound.

(9) **Calculation of Personnel Dose Response.** PRBTOX is called by two subroutines, DGDSTX and TOXDTH, to calculate the effective chemical dose (both percutaneous and vapor) received by a personnel asset. Parameters passed into PRBTOX are the normally distributed random number (RN) and the type of chemical agent (IAG).

The effective dose (AX) is equal to $\frac{D_{50}}{D_i}$ which can be derived in the following manner:

$$\text{Given: } \log D_i = \log D_{50} + RN\sigma L$$

$$\begin{aligned}\log D_i - \log D_{50} &= RN\sigma L \\ -\log D_i + \log D_{50} &= -RN\sigma L\end{aligned}$$

$$\log \frac{D_{50}}{D_i} = -RN\sigma L$$

$$AX = \frac{D_{50}}{D_i} = 10^{(-RN\sigma L)}$$

where:

- D_i = dose at which the individual (i) would exhibit a particular symptom;
- D_{50} = dose at which 50% of the population would exhibit the same symptom;
- RN = normally distributed random number
- σL = chemical probit slope of agent (PRBSLP(IAG))

PRBTOX first calculates the exponent (EX) by multiplying the negation of the normally distributed random number by the probit slope of the chemical agent type. The effective dose multiplier (AX) is then calculated by the following equation:

$$AX = \frac{D_{50}}{D_i} = 10^{EX}$$

In AURA, the distribution of dose responses is simulated rather than solved for analytically. Each individual is assigned a sensitivity factor which is held constant throughout the replication. The sensitivity is determined through a random number draw from a normal distribution. The random number (RN) is used to depict the softness/hardness of a personnel asset with respect to a chemical dosage. A random number of 0 means that a personnel asset falls exactly into the LD50th percentile and would receive an effective dose multiplier of 1. A random number greater than 0 indicates that a personnel asset has a greater

resistance/hardness to the chemical dosage and would result in an effective dose multiplier of less than 1. Therefore, the effective dose would be less than the actual dose. Alternatively, a random number draw of less than zero would indicate that the personnel asset would have less resistance (softer) to the chemical dose and would result in an effective dose multiplier greater than 1. Consequently, the effective dose would be greater than actual dose. The methodology used by PRBTOX is derived from the chemical probit slope³³ which is shown in Figure 27. PRBTOX then terminates and the effective dose (AX) value is passed back to the calling subroutine.

(10) **Accumulation of Chemical Dosages.** For each reconstitution event, subroutine CUMDOS accumulates chemical dosages from each target point and places them in dose bins according to asset type and amount of dose. A dose bin for a specific asset (stored in array RA) contains the following information:

- RA(IJ+0) = dose in this bin
- RA(IJ+1) = pointer to next bin (-1 means no next bin)
- RA(IJ+2) = time of earliest dose
- RA(IJ+3) = type of degradation
 - 0 - demanding task
 - 1 - undemanding task
 - 999 - no degradation
 - 1 - all personnel in this bin are unusable
(based on kill criteria of interest
i.e., incapacitation, lethality, etc.)
- RA(IJ+4) = agent type
- RA(IJ+5) = personal hardness value
- RA(IJ+IBNL) = number of assets in the bin
- RA(IJ+IBNL+1 to 6) = working storage

A set of dose bins, which are stored in ascending order according to amount of dose, are created for each asset type or group. The personal hardness value is used for both chemical and nuclear and determines how susceptible an individual is to a chemical or nuclear dose.

Subsequent dosage at a target point is assessed against personnel assets located there in the following manner. Depending on the dose received, CUMDOS either moves personnel to a higher dose bin or creates a new bin. Due to substitutions (asset allocation) from job to job (i.e., target point to target point), which individual of a particular asset group has what dose or is deployed at what

33. Litchfield, J.T. and Fertig, J.W., "On a Graphical Solution of the Dosage-Effect Curve", Bulletin: Johns Hopkins Hospital, Volume 69, pp. 276-286, 1941, (UNCLASSIFIED).

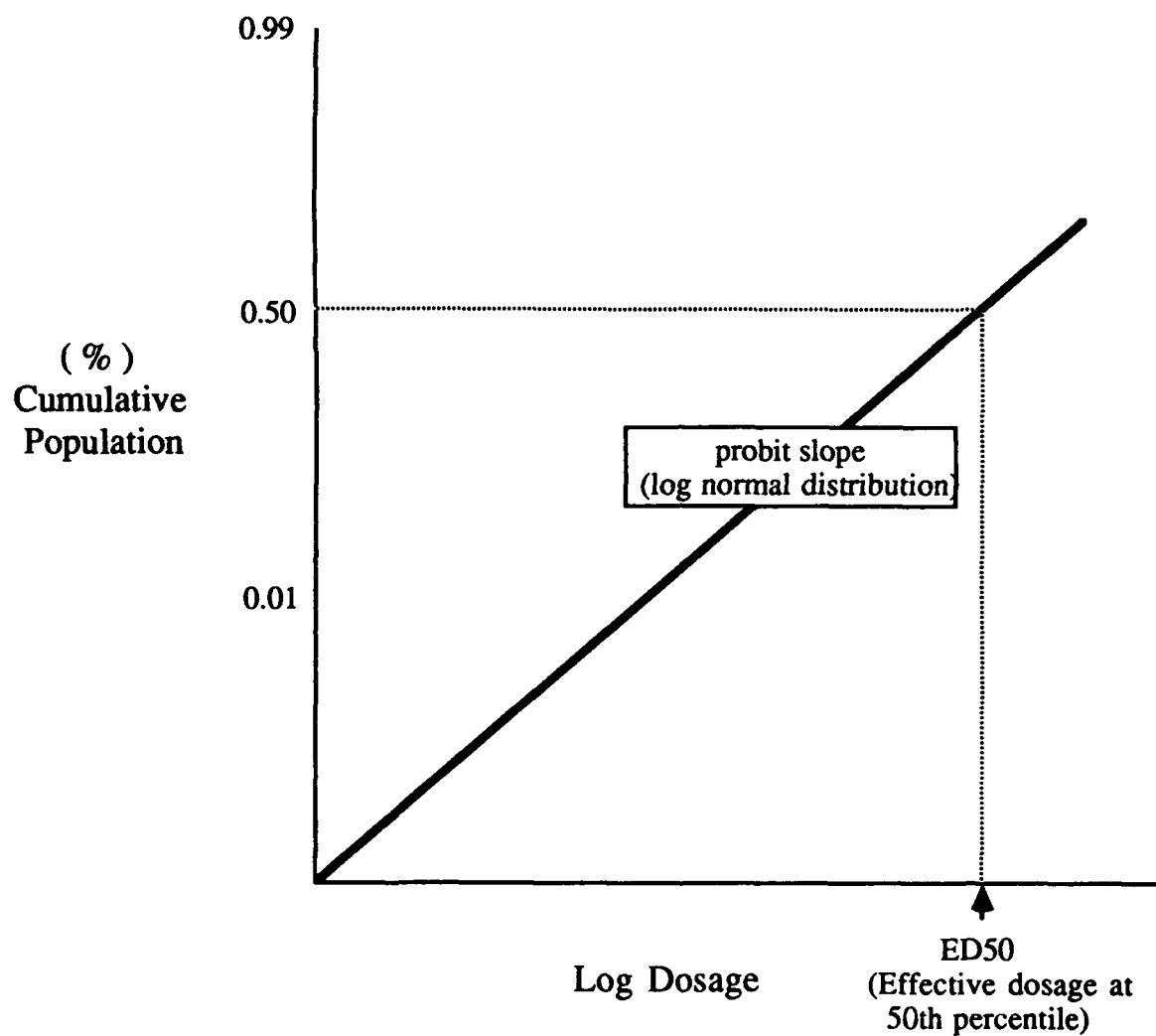


Figure 27. Illustration of Chemical Probit Slope

target point is not specifically known. Since the personnel asset at the target point of interest represents some fraction of the asset group, CUMDOS assumes that this fraction of the asset group in each dose bin receives the new dose.

Once CUMDOS determines that a new dose has been received, the bins have to be updated. As a result, the fraction of personnel assets receiving the new dose are moved to a bin of appropriate dose and matching characteristics. For personnel assets to match a bin, the difference between hardness values can be no greater than $1.E-5$. Doses are differentiated to the nearest $1/2\%$ of a lethal dose. If no such bin exists, a new one is created. This is done for all personnel assets.

To illustrate this process, a simple example follows: Two personnel from the same asset group are located at two different target points (A and B) but can perform the same job at either location. Suppose the following doses are experienced at the two target points during two distinct time intervals:

Dose	
Target Point A	Target Point B
Initial : 0.5 lethal dose	0.0 lethal dose
Later : 0.3 lethal dose	0.0 lethal dose

Initially, the first asset receives 0.5 lethal dose and the second 0.0 lethal dose. At some later reconstitution time, one receives an additional 0.3 lethal dose. Since substitutions (possibly fractions) may have been made, and AURA does not differentiate between members of the same asset group, determining which asset gets the additional 0.3 lethal dose is not straightforward. CUMDOS handles this situation in the following manner:

No. of Personnel (at Target Points A and B)	Dose
0.5	0.8
0.5	0.5
0.5	0.3
0.5	0.0

CUMDOS divides the dose among the two assets, assigning 0.3 of a lethal dose to 0.5 of the first asset and 0.5 of the second asset. In this example, then, 0.5 of each asset has the original doses (0.5 and 0.0) while the other 0.5 of each asset has the original dose plus the new dose (0.8 and 0.3). CUMDOS then moves those fractions which received additional doses (0.5) to new bins (see bins above). As these bins indicate, the assets have a 50% probability of accumulating one of the four dose levels.

(11) Calculation of Personnel Permanent Zero Performance Time. Subroutine PZPTOX is called by TOXDTH with two parameters, amount of chemical received by an individual (DOSE) and chemical agent (IAGENT), and returns the permanent zero performance (PZP) value. This subroutine checks the chemical dose array, which contains seven default values set in subroutine DGOSET, to determine at what time an individual goes to permanent zero performance. If the dose is less than the first four values in the array, then the variable PZP is set to infinity as individuals with that dose will not go to zero. If the dose is greater than or equal to the last three dose values in the array, PZPTOX linearly interpolates between the times given for those doses and the actual dose received to determine when the individual will go to permanent zero performance.

(12) Calculation of Chemical Casualties. TOXDTH is called by RECEVN to calculate personnel casualties for each reconstitution. TOXDTH, in turn, calls subroutine PZPTOX to determine the time at which an individual with a dose goes to permanent zero performance. The time of permanent zero performance is then compared to the elapsed time since dose was received (ETIME) to determine if the individual is a casualty. TOXDTH first loops through all assets to eliminate equipment from consideration since TOXDTH processes only personnel lethalties. TOXDTH then skips over the 0-dose bins, that is, personnel who have not received a dose and goes on to the other bins. If a personnel asset is found in the bin, TOXDTH tests to see if it has received the maximum dose, which means the individual is already a casualty. If the personnel asset has received the maximum dose, the bin is flagged and TOXDTH goes on to the next asset. If the asset has not received a maximum dose, then the asset is a potential casualty. In this case, TOXDTH determines which chemical agent caused the dose and then calls PRBTOX to calculate the effective dose. Once the effective dose is calculated, PZPTOX is called to test for a casualty. PZPTOX calculates the time at which the personnel asset becomes a casualty. If that time is less than the current time with respect to when the dose was received (ETIME), then a casualty has occurred. Once this is done, TOXDTH updates the running totals for asset survivors, asset survivors at this reconstitution time and the number of casualties due to chemical dose. TOXDTH then checks to see if enough time remains in the simulation to produce additional casualties. If there is, a flag is set which prompts RECEVN to call TOXDTH in the next reconstitution.

(13) Calculation of Amount of Vapor Dosage Over Time (per Target Point). TOXLTH is called by EVNTDO to evaluate the amount of vapor dosage present at each target point from all chemical weapons, over time. TOXLTH first checks to see if the round produced a vapor threat. It then checks to see if the target point lies within the vapor grids of the weapon. If so, it calculates the target point's sensitivity (i.e., whether or not it is vulnerable to the agent's effects depending on MOPP posture and penetration factor). TOXLTH calls UNPK6 to see if the target point is in its original MOPP posture. TOXLTH then checks to see if there is an alternate posture for the target and whether or not the target changed during the current time interval. DOSAGE is then called

to evaluate the dosage received by the target up to the MOPP time. Once all weapons have been processed with this procedure, TOXLTH calls the MOPP subroutine for the MOPP change. Finally, it updates the last evaluation time.

d. Application of the Hypothetical Case to the Chemical Lethality Model. The main threats to the Ammunition Supply Point (ASP) from a chemical attack are the direct and indirect effects of the agent on unit personnel. To a smaller extent, the agent also has an effect on unit equipment and supplies.

The indirect effect on unit personnel results from the wearing of the protective gear (MOPP) and the associated performance degradation. Subroutine MOPP and its entries are used in this instance to determine what posture unit personnel are in (original or alternate) and to set the time for the change if it has not already taken place. The actual degradation values associated with the various job categories within the ASP are assigned using the DEGRADATION mnemonic in the input runstream. For the ASP, eight job categories were developed. Unit personnel were assigned to a job category depending on the physiological requirements of their job; degradation values were then assigned to the job categories, again, depending on the physiological requirements of the job, i.e., vision acuity, manual dexterity, etc. A related mnemonic is Toxic Kill Criteria (T.K.C.) which equates a job code number with its verbal description and chemical dose multiplier. Also input with this mnemonic are the associated heat stress parameters. The following is an example of inputs for these two mnemonics:

DEGRADATION

0,1,1.0 #MOPP Level, Job Category Number, Degradation

0,2,1.0

4,1,0.9

4,2,0.6

END

T.K.C.

Supervisor, 1, 1.0, .1, 100.0, 25.0 #Verbal Description, Job

#Category No., dose

#multiplier and heat

#stress parameters

Ammo Loader, 2, 1.2, .3, 15.0, 30.0

END

In this example, there are two job categories, supervisor (1), and ammo loader (2). Both are 100% effective when not in MOPP (MOPP 0) while the supervisor is degraded to 90% and the ammo loader to 60% when in MOPP 4. From the T.K.C. mnemonic, one can see that the supervisor has a dose multiplier of 1.0 and the ammo loader 1.2. The supervisor's dose is unaffected by his job while the ammo loader's dose will be multiplied by 1.2 to simulate a higher than normal ratio of dose. Also listed are the heat stress parameters. For example, in the supervisor category, a 0.1 probability exists of becoming a heat stress casualty;

the 100.0 is the associated heat stress lag while the 25.0 is the characteristic probability growth time.

The direct effects on personnel are the lethal and sublethal doses received as a result of the chemical attack. For example, to determine the amount of vapor dosage present at each target point in the unit, subroutine TOXLTH is used. TOXLTH checks to see what posture unit personnel are in and then evaluates the dose received at a target point up to the time of the MOPP posture change. A number of other subroutines are called to calculate such things as dosage, sublethal doses and lethalties; these include DOSAGE, TOXDTH, PRBTOX, PZPTOX and DGDSTX.

Contamination as a result of the chemical attack is the main worry of the ASP with respect to unit equipment and the ASP ammunition stacks. If equipment is contaminated, then a number of modeling options are available. The following example input describes just a few of the options available in AURA

```
CONTAMINATED USAGE
RECORDS
END
PERSISTENCE
AGENT1, 1.5
END
REPAIR
VEHICLE
$CBR PEOPLE, 0, 1., 30.,10.,100.,100.
END
```

In this example, use of contaminated equipment is determined through the CONTAMINATED USAGE option. For the ASP in this example, only office records, referred to as RECORDS, can be used while contaminated. The user has the option of specifying particular pieces of equipment or using ALL to indicate that all equipment can be used while contaminated. Note, though, that if this option is set, unit personnel can use the equipment only if they remain in their alternate MOPP posture until all equipment is decontaminated, either by natural evaporation or deliberate decontamination.

Another option available is changing the persistence time calculated by the chemical model, NUSSE. Since NUSSE calculates persistency based on Canadian grassland, the user may wish to change this value to reflect the persistency of the agent on different surfaces, i.e., alkyd-painted vehicles. In the example above, the persistency of AGENT1 is increased by a factor of 1.5

Finally, in this example, personnel are identified for performing decontamination. Using the REPAIR mnemonic, items to be decontaminated are identified as well as the link or subchain required, the type of repair, associated penalty in

mission performance, the time and associated standard deviation to affect the repair, and the X and Y location. The equipment VEHICLE is to be decontaminated by CPR PEOPLE with a mission performance penalty of 1.0; decontamination will take 30 minutes, with a 10 minute standard deviation, at the 100., 100. location. Note, though, that this option is possible through the use of the REPAIR mnemonic. Thus, although the need for decontamination arises from a chemical attack, the actual playing of decontamination in AURA is part of the Repair algorithms.

Discussion of typical outputs associated with a chemical scenario is more complicated as these can change with selected output options. However, default outputs include the number of surviving assets including contaminated assets, the dose bins, dose degraded capability and incapacitation of personnel, and total number of items contaminated and those still contaminated at the end of the simulation. Again, this is only a small number of the output options available when modeling chemicals.

This hypothetical case provides an example of the application of chemical modeling within AURA to a military unit. The actual running of the code and the code's responses to these inputs are beyond the scope of this discussion. Although this hypothetical case in no way exhausts AURA's chemical modeling capabilities, it does present a typical chemical scenario and related inputs.

4. Nuclear Vulnerability Model

a. Triggering a Nuclear Scenario. An AURA nuclear scenario is triggered when the mnemonic card NUCLEAR VULNERABILITY DATA is encountered in the AURA runstream. This card causes AURA (via subroutine NUCIN) to open and read a vulnerability data file (for equipment only) as FORTRAN unit 3.

The nuclear vulnerability file inputs data describing the vulnerability of the equipment to the nuclear environments produced by the weapon. Weapons produce a set of environments and a target's vulnerability is assessed as a function of environmental strength. Since all weapons are converted to environments, only one data line is necessary for each target. The data line contains the asset name (from the ASSET list under NAMES), a code number which indicates the environments to which the item is susceptible and the required data. AURA calculates blast-overpressure and dynamic impulse ($dP \cdot I_q$), neutron fluence, electromagnetic pulse (EMP) and thermal fluence since these are the environments currently in the Harry Diamond Laboratory (HDL) nuclear vulnerability data base (NUDACC). The code number and required data are taken directly from this data base. See section V.4.c.4 for further discussion on the nuclear vulnerability file and the NUDACC data base. AURA, also, has the ability to simulate enhanced radiation weapons.

A nuclear vulnerability scenario is a time dependent event in which each nuclear detonation produces a set of environments and each target point's vulnerability is assessed as a function of environmental strength. Personnel and equipment are susceptible to different environments and must be treated separately. For personnel, their vulnerability to blast (psi), thermal fluence ($\frac{cal}{cm^2}$) and ionized radiation (rads) is assessed for each weapon at each target point. Equipment vulnerability is assessed for each weapon at each target point according to the data in the nuclear vulnerability file (see section V.4.c.4). The vulnerability of each individual asset (personnel and equipment) is stored and used in other subroutines to determine asset allocation, link effectiveness and unit effectiveness as a function of time.

b. Overview of Nuclear Vulnerability Model. The model requires inputs in the following five areas in order to assess the vulnerability of each asset against each weapon: weapons, unit deployment, nuclear shielding factors, vulnerability file (unit #3) and degradation values for sublethal radiation doses. This section will discuss these inputs and specifically how the vulnerability of assets (personnel and equipment) is assessed against the various nuclear environments.

c. Inputs.

(1) **Weapons.** The two weapon inputs needed to assess nuclear vulnerability are the actual ground zero (AGZ) and the yield of the detonation. The AGZ is calculated based on the designated ground zero (aimpoint) and any associated delivery errors. The yield is a user input. AURA models enhanced radiation weapons by allowing the user to input two yields. The first being the customary blast yield and the second describing the radiation effects. Both the AGZ and yield information are passed to the nuclear vulnerability subroutines from the weapon subroutines.

(2) **Unit Deployment.** The x and y coordinates of the target point, asset identification, how many assets at the target and the nuclear posture are the unit deployment inputs. Each asset is identified by an integer which is assigned as the asset is read from the AURA runstream. The x and y coordinates of the target are used to locate the target relative to the AGZ. The nuclear posture is a code number which describes the protective posture of personnel (i.e. *in the OPEN, in a TANK, in a FOXHOLE*, etc.). This posture code points to shielding factors which can (user's choice) afford protection from all three nuclear environments: blast, radiation and thermal. (See the next section for a more detailed explanation of how these nuclear shielding factors work.) This unit deployment information must be defined at each target point and passed to the nuclear vulnerability subroutines.

(3) **Shielding Factors.** The nuclear shielding factors afford personnel protection against nuclear environments (blast, thermal and/or radiation). (NOTE: Shielding for equipment is built into the HDL data set.) Each shielding factor, associated with a nuclear posture, is input by the user in the AURA runstream. Each entry (one per shielding factor) in the runstream contains a verbal description, the nuclear posture code number and four radiation transmission factors. The verbal description describes the posture of the personnel (*in the OPEN*, *in a TANK*, *in a APC*, etc.). The nuclear posture code is an integer which acts as a liaison between the shielding factor entry and each person deployed in the runstream. The first three (1 - 3) code numbers are reserved in AURA for *in the OPEN*, *in the OPEN-BUT-THERMALLY-SHIELDED* and *in a FOXHOLE*, respectively. A vehicle cannot be associated with these first three codes. Code numbers 4 thru 61 are selected by the user. The four transmission factors describe protection from ionized radiation by determining the total radiation dose that is transmitted through the protective posture. The first transmission factor is for incident neutrons, the second is for secondary neutrons due to incident gammas, the third is for secondary gammas due to incident neutrons and the fourth is for incident gammas.

The shielding factors can provide protection from blast and thermal environments as well as from radiation, which is afforded by the transmission factors. For the user to specify protection from a blast environment, an asset (usually a vehicle) is associated with a posture code (4 thru 61) by placing the asset name (preceded by a dollar sign (\$)) underneath the shielding entry in the AURA runstream. This affords personnel in that posture the same blast criteria (defined in the nuclear vulnerability file) as the asset. As for protection from the thermal pulse, only posture code number 1, *in the OPEN*, is subjected to a thermal exposure. For example, code number 5 may be the nuclear posture of *in a TANK*. This provides complete protection from a thermal pulse and protection from radiation according to the transmission factors listed. To give personnel in that posture the same blast criteria as the tank, the tank's asset name (as it appeared in NAMES list) is listed below that line in the AURA runstream as follows:

```
in the TANK, 5, trans1, trans2, trans3, trans4
$ tank
```

These shielding inputs provide a realistic nuclear scenario and are necessary for the nuclear vulnerability model to assess a target's environmental strength.

(4) **Nuclear Vulnerability file.** The fourth required input is the data from the nuclear vulnerability file (read in by subroutine NUCIN). This file contains the asset name, code number and required data (from HDL NUDACC data base)³⁴ to calculate the probability of kill for each piece of equipment at

each target point. The asset name is taken from the NAMES list. The code number identifies to which of the four nuclear environments (blast-overturn, neutron fluence, EMP or thermal) the particular asset is susceptible. The code numbers are used as follows: 1 = EMP, 2 = neutron fluence, 4 = blast-overturn and 8 = thermal. The code number can represent as many of these environments as necessary. For example, code number 7 (1 + 2 + 4) identifies EMP (1), neutron fluence (2) and blast-overturn (4). The required data provides the information needed to determine the probability of kill from that particular environment(s) for that piece of equipment.

The required data are slightly different for each of the four environments. The only input to the NUDACC data base is the piece of equipment of interest. For EMP, the NUDACC data base provides the mean and standard deviation for the log-normal curve which is calculated (in function PSFUNC) as a probability of kill versus environment level. A log-normal curve (PK vs environment level) is also used for neutron fluence and blast-overturn. For neutron fluence, the data base gives the log-normal mean and standard deviation along with a transmission factor for neutrons. Along with the mean and standard deviation, the data base provides a threshold value for blast-overturn which shifts the entire curve so as to get no effect until this level is achieved. For thermal fluence, the data base provides a thermal level called the damage fluence at or above which the equipment will not survive. The NUDACC database has not been updated in a number of years; consequently, it lacks data on many of the newer pieces of equipment.

(5) **Degradation Input for Sublethal Dose of Radiation.** The nuclear subroutines assess personnel vulnerability to blast, thermal and radiation. Based on this assessment, if the individual survives and is not experiencing early transient incapacitation due to a radiation dose, then degradation is assessed as a function of radiation dose and time. The degradation data is set in subroutine DGOSET. The determination of the individual's effectiveness involves interpolation and will be discussed in section V.4.e.1. This degradation data is provided by AURA, but the user can input his/her own set.

d. **Nuclear Vulnerability Methodology.** As previously stated the nuclear vulnerability of targets is assessed as a function of environmental strength. Personnel and equipment are susceptible to different nuclear environments. This section will discuss the flow through the nuclear subroutines for both personnel and equipment. A more detailed discussion of these routines will be the topic of the next section.

34. Vault, William L., "Vulnerability Data Array: The Agreed Data Base - Final Report (U)," Harry Diamond Laboratories, HDL-TR-1906, (July 80), (SECRET).

(1) **Personnel.** Personnel vulnerability is assessed for blast (psi), radiation dose (rads) and thermal fluence (cal/cm**2) using the inputs described above. First, the maximum radius of interest for these three environments is calculated by weapon yield in the function RMAX. This is done by setting the minimum environmental level and then calling subroutine ITRNUC to find the range at which this minimum level is achieved. If the distance from the actual ground zero (AGZ) of the weapon to the target point is greater than the maximum radius of interest then that target point is not analyzed for that particular weapon and environment. If the distance falls within the maximum radius, then subroutine NUCENV calculates the nuclear environments based upon the weapon yield at this distance.

Next, for all lethality events, subroutine NUCDMG calculates the total probability of kill (PK) due to blast and thermal for each target point using the survivor rule ($PK=1-(1-BLASTPK)(1-THERMALPK)$). NUCDMG calls NUCENV to calculate the environment strength at the target point and then based on the individual's posture, determine the PK (see subroutine NUCDMG). The total radiation dose and neutron to gamma ratio is also calculated for each target point. NUCDMG calls NUCENV to calculate the level of radiation and based on the individual's posture, determine the total dose received and neutron to gamma ratio. Finally, subroutine NUCLTH updates the casualties due to blast and thermal at each target point by using the PK's calculated in NUCDMG.

For each reconstitution event, personnel vulnerability to radiation dose is assessed since radiation sickness/death is a function of time; whereas personnel vulnerability for blast and thermal is only calculated when a lethality event occurs. Subroutine CUMDOS accumulates the radiation doses received at each target point. Subroutine NUCDTH determines if an individual survives from a prior dose. If death due to radiation has occurred, then the casualties' array is updated. If the individual survives, then subroutine ETI is called to determine whether or not the individual is experiencing an early transient incapacitation (ETI) based on the dose level. If the individual is not experiencing ETI and is not dead (according to NUCDTH), then the individual's degradation is calculated for a sublethal dose in subroutine DGDSTM. The information calculated in NUCDTH, ETI, DGDSTM and NUCLTH is stored for use later in determining asset allocation, link effectiveness and unit effectiveness.

(2) **Equipment.** The four nuclear environments used to assess equipment vulnerability are electro-magnetic pulse (EMP), neutron fluence, blast-overturn and thermal fluence. The vulnerability inputs are taken from HDL's NUDACC database and input via the nuclear vulnerability file. In this file, a code number designates to which of the four environments a particular piece of equipment is susceptible. First, subroutine NUCCHK checks to see that each piece of equipment has vulnerability data. If one does not, then AURA prints an error message. Next, as with personnel, the distance of the target from the AGZ is checked against the maximum radius for that weapon. If the target

falls within the maximum radius, then subroutine NUCDMG computes the total probability of kill, using the survivor rule, for the environments to which the equipment is susceptible. The probability of survival for equipment due to EMP, neutron fluence, blast-overturn and thermal fluence is calculated in subroutines PSEMP1, PSNF, PSDPIQ and PSTHRM, respectively, and passed to NUCDMG. Finally, NUCLTH is called to update the equipment lost. These update are done for each lethality event and stored to later determine asset allocation, link effectiveness and unit effectiveness.

e. Nuclear Vulnerability Subroutines and their Functions.

(1) Calculation of Probability of Kill due to Nuclear Attack.

For each lethality event, subroutine NUCDMG calculates the probability of kill (PK) from a nuclear detonation for personnel and equipment by target point. The PK is based on the target's (personnel or equipment) vulnerability to the nuclear environments generated by each weapon. Personnel and equipment will be discussed separately since their vulnerability is assessed for different environments.

First, NUCDMG sets the environments to zero for no effect. Next, the radius of the fireball is calculated and the target point is checked to see if it falls inside this radius. If so, then the PKs are set to one, the radiation dose is set to -10. and the other nuclear environments are set to the maximum at 1.0E35. The code then returns to the calling routine. For those target points outside the fireball, the routine differentiates between personnel and equipment at the target.

(a) **PK for Personnel.** For personnel, the PK for blast and thermal are calculated in this routine, but radiation is handled separately in RADDTH. The first input established (via nuclear shielding factors) is the nuclear posture of the individual. If the nuclear posture is associated with a vehicle, then the individual has the same blast criteria as the vehicle and the PK (1. - PS) for blast is obtained from subroutine PSDPIQ. Also, the PK for thermal fluence is zero.

To determine the blast PK for the other nuclear postures, a cumulative log-normal curve is calculated (PK vs. static overpressure) in subroutine CUMNRM. The U.S. Army Nuclear and Chemical Agency's (USANCA) blast kill criteria³⁵ is used to develop the curve. For a specific weapon yield, the blast kill criteria give a static overpressure level (50% criterion) at which 50% of the population will die and a variance which is arbitrarily set at the square of 20% of the

35. Klopacic, J.T., Watson, LTC D.L., "Modeling Casualties in Nuclear Warfare", USA Ballistic Research Laboratory Memorandum Report No. BRL-MR-3771, July 1989, (UNCLASSIFIED).

50% criterion. Given this criteria and the static overpressure for that target point, CUMNRM will return a PK for personnel at that target point. NUCENV is called to obtain the static overpressure level for the target point of interest. For personnel *IN A FOXHOLE*, the 50% criterion is set at 43 psi (variance = 73.96) and CUMNRM is called to return the PK for blast. For all other postures, CUMNRM is called and USANCA's blast kill criteria provides the inputs based on the weapon yield.

Next, the PK for thermal fluence is calculated for personnel at each target point. The only nuclear posture that is susceptible to a thermal pulse is *IN THE OPEN*; therefore, the PK for thermal is zero for any other nuclear posture. Again, subroutine CUMNRM calculates the cumulative log-normal curve (PK vs. thermal fluence) based on USANCA's thermal kill criteria. For a specific weapon yield and uniform type (summer or winter), USANCA's thermal kill criteria provide the 50% criterion in calories per square centimeter and the variance (same as blast kill criteria). Then, NUCENV is called to return the thermal fluence level at that target point based on the atmospheric conditions (good, average or poor) set by the user. For a particular thermal fluence level and thermal kill criteria, CUMNRM returns the PK for thermal (PKTHM).

Finally, the PKs for blast (PKBLT) and thermal (PKTHM) are combined using the survivor rule to obtain a total PK for personnel. The survivor rule assumes independence between the two probabilities.

$$PK = 1. - (1. - PKBLT)(1. - PKTHM)$$

(b) **PK for Equipment.** NUCDMG calculates the total probability of kill (PK) for equipment based on the nuclear environments to which a piece of equipment is susceptible. The environments calculated in AURA for equipment are neutron fluence, electro-magnetic pulse (EMP), blast-overture (DPIQ) and thermal fluence. First, to determine which of these four environments a particular piece of equipment is susceptible, the code number in the nuclear vulnerability file is retrieved (see section V.4.c.4). Then, the appropriate subroutine (PSEMP1, PSNF, PSDPIQ or PSTHRM) is called to return the probability of survival (PS1, PS2, PS3 or PS4, respectively) from that environment. If a piece of equipment is not susceptible to one or more of the four environments, then the probability of survival from that environment is set to one. The total PK is calculated using the survivor rule.

$$PK = 1. - (PS1 \times PS2 \times PS3 \times PS4)$$

If the stochastic option (user input) is turned on, then a uniform random number is drawn and compared to this total PK. If the random number is greater than the total PK, then PK is set to zero. Otherwise, the PK is 100%. If this option is not specified, then AURA defaults to the deterministic mode and the PK calculated (above) from the survivor rule is be used.

(2) **Calculation of Nuclear Environment.** Subroutine NUCENV computes the nuclear environments for each lethality event in AURA in accord with a Harry Diamond Laboratory report³⁶. NUCENV is the controlling routine for several other routines and functions which were developed by Horizons Technology Inc. (HTI) for the DNA. The referenced HDL report was augmented to use the algorithms developed by HTI³⁷. The HTI developed subroutines, called by NUCENV, are as follows:

PQAIR QTHERM
PT TDOSE
QT

The nuclear environments and the manner in which they are calculated are as follows:

Define:

W = weapon yield (kiloton)
DS = range from target to AGZ (meters)
 $HOB = (60)(W^{\frac{1}{3}})$ (height of burst)
 $SR = \sqrt{DS^2 + HOB^2}$ (slant range (meters))
 $SW = W^{\frac{1}{3}}$ (scaled yield)
 $SGR = \frac{DS}{SW}$ (scaled ground range)
SHOB = 60 (scaled height of burst)

1) Neutron fluence ({neutrons over {cm sup 2}}) :

$NFLUENCE = (4.82E12)(W)(DS^{-2})(e^{-1.44})(DS)$
(calculated in NUCENV)

2) Rate of change due to time of gamma flux (rads-silicon/sec) :

36. Kelley, C.S., et al., "Nuclear Damage to Point Targets," Harry Diamond Laboratories, HDL-TR-1876, December 1978, (UNCLASSIFIED)

37. Jordan, D., "Weapon Effects ROM, Volume II-Reference Handbook," Horizons Technology Inc., prepared for Defense Nuclear Agency, DNA-EH-84-01-A-V2, 31 May 1984, (UNCLASSIFIED)

$$\text{GAMMADOT} = (6.45E9)(W^{0.987})(DS^{-2.79})(e^{-3.11})(DS)$$

(calculated in NUCENV)

3) Neutron component of total dose :

$$\text{NDOSE} = (\text{calculated in TDOSE})$$

4) Gamma component of total dose :

$$\text{GDOSE} = (\text{calculated in TDOSE})$$

5) Thermal fluence (cal per cm^2) (for personnel only) :

$$\text{THERM} = \frac{(8E6)(C)(W)}{(SR)(SR)}$$

(calculated in QTHERM)

a) poor atmospheric conditions:

$$\text{VISIBILITY} = 1200 \text{ meters}$$

b) average atmospheric conditions:

$$\text{VISIBILITY} = 10001 \text{ meters}$$

c) good atmospheric conditions:

$$\text{VISIBILITY} = 79999 \text{ meters}$$

6) Static Overpressure (psi) :

$$P = (PREG)(\sigma) + (PMACH)(1-\sigma) \text{ (pascals)}$$

(calculated in PQAIR)

$$\text{STATP} = \frac{P}{6894.757} \text{ (psi)}$$

(calculated in NUCENV)

7) Total overpressure impulse (psi-sec) :

$$\text{PT} = \frac{\text{SUM}}{6894.757}$$

(calculated in PT)

$$\text{STATI} = (\text{PT})(\text{SW})$$

(calculated in NUCENV)

8) Dynamic pressure, ideal (psi) :

$$Q = (.5) ((1+\sigma)(\cos\theta^2)-1) (P) (ETA-1) \text{ (pascals)}$$

(calculated in PQAIR)

$$DYNPIDEL = \frac{Q}{6894.757} \text{ (psi)}$$

(calculated in NUCENV)

9) Dynamic pressure, light dust (psi) :

$$DYNPDUST = (3.46E-2) (W^{0.747}) (DS^{-2.24}) (e^{0.596}) \left(\frac{W^3}{DS}\right)^{\frac{1}{2}}$$

(calculated in NUCENV)

10) Total dynamic pressure impulse, ideal (psi-sec) :

$$QT = \frac{SUM}{6894.757}$$

(calculated in QT)

$$DYNIDEL = (QT)(SW)$$

(calculated in NUCENV)

11) Total dynamic pressure impulse, light dust (psi-sec) :

$$DYNIDUST = (5.78E-3) (W^{0.853}) (DS^{-1.56}) (e^{0.522}) \left(\frac{W^3}{DS}\right)^{\frac{1}{2}}$$

(calculated in NUCENV)

12) Electric field (Volts/m) :

a) In theta direction:

$$ETHETA = (-1.39E4) (W^{0.215}) (DS^{-1.28})$$

(calculated in NUCENV)

b) In radial direction:

$$ERADIAL = (1.53) (W^{0.4}) (DS^{-2.3})$$

(calculated in NUCENV)

13) Magnetic field (Amperes/m) :

$$BPHI = (-0.462) (W^{0.215}) (DS^{-1.28})$$

(calculated in NUCENV)

Subroutine NUCENV is called by NUCDMG to compute the total probability of kill for personnel from blast (DPIQ) and thermal fluence. NUCDMG also retrieves the total radiation dose received by each target point.

(3) Calculation of Electro-magnetic Pulse in an EMP Environment. Subroutine PSEMP1 calculates the probability of survival (P_s) of equipment from an electro-magnetic pulse (EMP) associated with a nuclear detonation. Having the weapon yield (W) and the distance (DS) of the target point from actual ground zero (AGZ) passed in FORTRAN common blocks, this routine calculates the EMP with the following equation:

$$EMP = \left(\frac{(7930.)(W^{0.154})}{DS} \right) \left(e^{(0.688)\left(\frac{W^{0.154}}{DS}\right)} \right)$$

Then, the mean and standard deviation of the cumulative log-normal curve are retrieved from the nuclear vulnerability file via the primary storage array RA. Finally, these values are passed to function PSFUNC which calculates the cumulative log-normal curve as probability of kill (PK) versus EMP and returns the PS (1. - PK). (Note: PSFUNC, also, calculates the cumulative log-normal as PK versus environment level for neutron fluence and blast-overturn.)

(4) Calculation of Probability of Survival from Neutron Fluence. Subroutine PSNF calculates equipment's probability of survival (PS) from neutron fluence associated with a nuclear detonation. Again, the weapon yield (W) and distance (DS) of the target point from AGZ are passed in FORTRAN common blocks. From these values, the neutron fluence (RNF) is calculated as follows:

$$RNF = (4.82E12)(W)(DS^{-2})(e^{(-4.44)(DS)})$$

There is a shielding factor (between 0 and 1) associated with each nuclear posture for neutron fluence and input via the nuclear vulnerability file. The computed neutron fluence (RNF) is multiplied by this shielding factor to obtain the actual neutron dose received by that particular piece of equipment in that nuclear posture. As with EMP, the cumulative log-normal mean and standard deviation are retrieved from the nuclear vulnerability file via the primary storage array RA. Finally, function PSFUNC is called to calculate the cumulative log-normal curve and to return the P_s from neutron fluence.

(5) Calculation of Probability of Survival due to Nuclear Blast Dose. Subroutine PSDPIQ calculates the probability of survival (PS) from blast-overturn as a function of overpressure-impulse (DPIQ = delta P (peak overpressure) * Iq (dynamic impulse)) for equipment. The nuclear weapon yield (W) and distance (DS) of the target point from AGZ are passed in FORTRAN common blocks. Using these two values, blast-overturn (DPIQ) is computed by the following equation:

$$\text{DPIQ} = (1.59E-2)(W^{1.713})(DS^{-4.14})$$

Input from the nuclear vulnerability file for each piece of equipment is a threshold blast-overturn level (NUDACC database). If DPIQ is less than or equal to the threshold level, then the PS is 100%. Otherwise, the cumulative log-normal mean and standard deviation are taken from the nuclear vulnerability file (via the primary storage array RA) and function PSFUNC calculates the curve as probability of kill (PK) versus DPIQ. The PS (1. - PK) for blast-overturn is then passed to subroutine NUCDMG to compute the total PK for that piece of equipment.

(6) Calculation of Probability of Equipment Survival from Neutron Fluence. Subroutine PSTHRM calculates the probability of survival (PS) for equipment from the thermal fluence (THERM) associated with a nuclear detonation. First, parameters for closed form approximation of thermal environments (from HDL-TR-1876)³⁸ as a function of yield and range are read into the routine. There is a set of these parameters for each of the three types of atmospheric conditions (good, average and poor). The better the atmospheric condition, the greater the thermal fluence. The atmospheric condition is selected by the user (default is average). Next, the weapon yield (W) and distance (DS) of the target point to AGZ are passed in FORTRAN common blocks. Using these values (W and DS) and the HDL parameters, the thermal environment is computed for particular atmospheric conditions by the following equations:

For poor atmospheric conditions:

$$\text{THERM} = (2.03)(W^1)(DS^{-2.23})e^{(-0.309)(DS)}$$

For average atmospheric conditions:

$$\text{THERM} = (2.88)(W^1)(DS^{-1.90})e^{(-0.116)(DS)}$$

For good atmospheric conditions:

$$\text{THERM} = (3.95)(W^1)(DS^{-1.84})e^{(-0.0843)(DS)}$$

Finally, the damage fluence is obtained from the nuclear vulnerability file (via the primary storage array RA) and compared to THERM. If THERM is greater than or equal to the damage fluence, then the PS is zero. Otherwise, the PS is 100%. These equations were developed by HDL and are used for computing the thermal pulse against equipment only. The thermal pulse against personnel is computed in NUCENV and was developed by the Defense Nuclear Agency (DNA). The HDL calculation is used for equipment because they provided the data (in the

nuclear vulnerability file) which models equipment's vulnerability to nuclear environments. Subroutine NUCLTH calls this routine to obtain the PS for equipment susceptible to thermal fluence.

(7) Calculation of Casualties Resulting from a Nuclear Attack. Subroutine NUCLTH calls NUCDMG to obtain the probability of kill (PK) for assets in order to update the total number of survivors at each target point. This routine basically does the bookkeeping for nuclear lethalties (personnel and equipment) and stores them in the appropriate arrays within AURA. Lethality means death for personnel and unusable for equipment. First, it retrieves the PK from NUCDMG for a particular target point. NUCDMG has already determined whether the asset is personnel or equipment. Then, the number of assets lost at that target point (CAS) is calculated by multiplying the number of assets by the PK. This value is used to update the array of the running total of survivors, array of current number of assets at that target point and the arrays of number of lethalties due to thermal and blast effects. For the latter, a death is attributed to thermal or blast effects based on their respective PKs.

(8) Accumulation of Radiation Dosages. For each reconstitution event, subroutine CUMDOS accumulates radiation dosages from each target point and places them in dose bins according to asset type and amount of dosage. A dose bin for a specific asset (stored in array RA) contains the following information:

- RA(IJ+0) = dosage in this bin
- RA(IJ+1) = pointer to next bin (-1 means no next bin)
- RA(IJ+2) = time of earliest dosage
- RA(IJ+3) = type of degradation:
 - 0 - demanding task
 - 1 - undemanding task
 - 999 - no degradation
 - 1 - all personnel are dead in this bin
- RA(IJ+4) = neutron to gamma ratio
- RA(IJ+5) = personal hardness value
- RA(IJ+IBNL) = number of assets in the bin
- RA(IJ+IBNL+1 to 6) = working storage

A set of dosage bins, which are stored in ascending order according to amount of dosage, is created for each asset type or group. The personal hardness value

38. Kelly, C.S., et al. op cit.

determines how susceptible an individual is to a radiation dose.

Subsequent dosage at a target point is assessed against personnel assets located there in the following manner. Depending on the dose received, CUMDOS either moves personnel to a higher dose bin or creates a new bin. Due to substitutions (asset allocation) from job to job (i.e., target point to target point), which individual of a particular asset group has what dose or is deployed at what target point is not specifically known. Since the personnel asset at this target point represents some fraction of the asset group, CUMDOS assumes that this fraction of the asset group in each dose bin receives the new dose.

Once CUMDOS determines that a new dose has been received, the bins have to be updated. As a result, the fraction of personnel assets receiving the new dose are moved to a bin of appropriate dose and matching characteristics. For personnel assets to match a bin, the difference between hardness values can be no greater than $1.E-5$. Doses are differentiated to the nearest $1/2\%$ of a lethal dose. If no such bin exists, a new one is created. This is done for all personnel assets.

To illustrate this process, a simple example follows: Two personnel from the same asset group are located at two different target points (A and B) but can perform the same job at either location. Suppose the following doses are experienced at the two target points during two distinct time intervals:

Dosage		
	Target Point A	Target Point B
	-----	-----
Initial:	5 rads	0 rads
Later :	3 rads	0 rads

Initially, the first asset receives 5 rads and the second 0 rads. At some later reconstitution time, one receives an additional 3 rads. Since substitutions (possibly fractions) may have been made, and AURA does not differentiate between members of the same asset group, determining which asset gets the additional 3 rads is not straightforward. CUMDOS handles this situation in the following manner:

No. of Personnel	Dose
-----	-----
0.5	8 rads
0.5	5 rads
0.5	3 rads
0.5	0 rads

CUMDOS divides the dose among the two assets, assigning 3 rads to 0.5 of the first asset and 0.5 of the second asset. It then moves those fractions (0.5) to new

bins (see bins above). As these bins indicate, the assets have a 50% probability of accumulating one of the four dose levels.

(9) Calculation of Time Dependent Casualties from Radiation Dosages. For each reconstitution event, RADDTH determines whether an individual (by dose bin) is dead, experiencing ETI or a survivor from his/her accumulated radiation dose and elapsed time since receiving the dose. First, each dose bin is checked in order to skip the zero dose bins. Then, the current time and previous time of evaluation are calculated. Now, subroutine RADDTH is called to return the conditional probability of survival (see Subroutine RADDTH). This probability is compared to a uniform random number and if the random number is greater than the probability of survival, then the individual in that dose bin is dead. Otherwise, a check is made for the possibility of future lethalties in that dose bin by checking the next reconstitution time and calling RADDTH again. If the probability of survival (from RADDTH) is one, then NUCDTH doesn't need to be called again for that dose bin. Next, subroutine ETI is called to return the probability that an individual is not experiencing ETI. This probability is compared to a uniform random number and if the random number is greater than the probability (from ETI.f), then those individuals in that dose bin are experiencing ETI and are removed from the usable assets pool. ETI personnel are not dead, but they are not available for use when the unit is optimized. This process is done for each dose bin at each reconstitution time.

(10) Calculation of Probability of NOT Experiencing Early Transient Incapacitation. Subroutine ETI computes the probability that an individual at a particular time is not experiencing early transient incapacitation (ETI). This probability is calculated based on the free-in-air dose of the individual, time after having received the dose, individual's task (physically demanding or visual discrimination) and the specific neutron to gamma ratio. Subroutine NUCDTH calls this routine if it has been determined that the individual has received a dose, but is not dead. Subroutine ETI was written by the Armed Forces Radiobiology Research Institute (AFRRI)³⁹.

(11) Calculation of Probability of Personnel Survival as Function of Radiation Dose and Time. Subroutine RADDTH calculates the conditional probability at a particular time that an individual has survived (from a radiation dose) the period of time since some previous time given that he/she survived to that previous time. The previous time is the last time that the individual was evaluated (by this routine) for receiving a radiation dose. If this is the first time the individual is being evaluated by RADDTH for receiving a dose, then

39. Private communication with Bill Jackson of AFRRI.

the previous time would be time zero. To compute this conditional probability RADDTH needs the present time, last time the individual received a dose, radiation dose received, individual's task (physically demanding or visual discrimination) and the specific neutron to gamma ratio. Subroutine RADDTH was also written by AFRRI⁴⁰.

(12) Calculation of Degraded Performance as Function of Radiation Dose and Time. Subroutine DGDSTM evaluates each job (link) to determine its effectiveness after having been subjected to a radiation dose. If the dose is sublethal, then this routine determines if, and to what degree, the individual is degraded based on the amount and time of dose. (This is for personnel only.) The degraded effectiveness values used in AURA (found in subroutine DGOSET) were developed by the DNA in their Intermediate Dose Program.⁴¹ Each effectiveness value corresponds to a dose and time combination. There are 19 doses and 39 times represented for both demanding and cognitive tasks with corresponding effectiveness values ($2 \times 19 \times 39 = 1482$). The first 18 doses range from 61.11 to 4529.94 rads with the last dose being 10000 rads. The times range from 12 to 75535.8 minutes. Corresponding with each dose/time combination is an incidence parameter which gives the probability that the job (link) will be degraded at that dose/time combination. There are two sets of effectiveness values and incidence parameters (same doses and times). One set for cognitive tasks and the other set for physically demanding tasks.

First, given the dose and time of dose for a particular job (physically demanding or cognitive), these values are checked to see if they exceed the upper or lower limits of the doses and times given by DNA. If the dose or time is too small or the time is too large, then the job effectiveness is 100%. If the dose is too large, then the job effectiveness is 0%. Otherwise, the dose and time are bracketed by an upper and lower value from the dose and time arrays from DNA in subroutine DGOSET. Next, based on the upper and lower dose and time values, four (four dose/ time combinations) incidence parameters and effectiveness values are selected from the DNA arrays. Using these four values, an interpolation is performed to get the incidence and effectiveness for that job at the desired dose and time. After the interpolation for the incidence parameter, the nuclear hardness value for that job is compared to the incidence parameter and if it is greater, then the job effectiveness is 100%. (The hardness number is drawn from

40. Private communication with Bill Jackson of AFRRI.

41. Dore, M.A., Anno, G.H., Wilson, M.A., "Acute Radiation Effects on Individual Crewmember Performance", (U), Pacific-Sierra Research Corporation under contract to the Defense Nuclear Agency, DNA-TR-85-52, Contract No. DNA 001-83-C-0015, August 1984, (UNCLASSIFIED)

a uniform distribution for each job to model the fact that some people are less susceptible to radiation sickness.) Otherwise, the interpolation for the effectiveness value is performed. Both interpolation processes are identical. Given the type of task, the algorithm is as follows:

Define:

DOSE = dose (rads) received by individual

TIME = time that dose was received

	dose	time
	----	----
upper	UPD	UPT
lower	LD	LT

EFF(dose, time) --> array of effectiveness values from DNA

T1 = EFF(LD, LT)

T2 = EFF(LD, UPT)

T3 = EFF(UPD, LT)

T4 = EFF(UPD, UPT)

Calculate:

$$\text{FRAC} = (\text{TIME} - \frac{\text{LT}}{\text{UPT}}) - \text{LT}$$

$$\text{D1} = \text{T1} + \text{FRAC} * (\text{T2} - \text{T1})$$

$$\text{D2} = \text{T3} + \text{FRAC} * (\text{T4} - \text{T3})$$

Interpolate:

$$\text{EFFECTIVENESS} = \frac{\text{D1} + (\text{D2} - \text{D1})(\text{DOSE} - \text{LD})}{(\text{UPD} - \text{LD})}$$

f. Application of Hypothetical Case to Nuclear Vulnerability Model. In summary, a typical nuclear scenario will be discussed using the hypothetical case study as our sample unit (Ammunition Supply Point). The focus will be AURA's modeling of nuclear weapon effects on personnel and equipment. Expected output from this scenario will also be discussed. This discussion will not exhaust the possibilities of AURA's ability to model a nuclear scenario, but will provide a typical example of the nuclear modeling in AURA.

The analyst will need inputs for the delivery of the weapon(s), the shielding factors for personnel protection and data for the vulnerability of equipment to the nuclear attack. The weapon inputs needed are the yield, aimpoint(s) and delivery errors. AURA allows the user to specify two yield inputs. The first represents the blast and thermal output in kilotons and the second is used for modeling enhanced radiation weapons. The height of burst is treated internally as sixty times the first yield raised to the one-third power.

To model personnel protection from nuclear environments in the ASP, personnel would be placed in protective postures. These postures will provide protection from the blast, thermal and radiation environments. The following is an example of postures for the ASP:

- (1) in the open
 - (2) in the open-but-thermally-shielded
 - (3) in a foxhole
 - (4) in a crane, 4, tr1, tr2, tr3, tr4
 - (5) in a truck, 5, tr1, tr2, tr3, tr4
 - (6) in a trailer, 6, tr1, tr2, tr3, tr4
- \$trailer

The first three postures are built into AURA. The next three are selected based on the unit and its mission. For radiation protection, the analyst obtains the transmission factors (tr1,tr2,tr3,tr4) associated with each protective posture (i.e. piece of equipment) from USANCA. The transmission factors model the fraction of incident neutrons and gammas which are received by the individual while in the protective posture. Only individuals placed in the first posture (in the open) are susceptible to the thermal fluence. These individuals are afforded no protection (i.e. 100% exposed). For protection from blast effects, individuals placed in the sixth posture (in the trailer) are given the same blast criteria as the trailer (i.e. if the trailer is killed by blast, so are the individuals in it). This can be done for any posture by placing a dollar sign (\$) followed by the name of the piece of equipment on the line below the posture input (see \$trailer above).

Inputs for modeling the protection of equipment from the nuclear environments are taken from the HDL NUDACC database and are input via the nuclear vulnerability file. This data are the criteria by which AURA determines when a piece of equipment is lost. An example of a nuclear vulnerability file for the ASP is as follows:

crane, 4, 2.2, 3.5, 2.56
truck, 4, 1.8, 2.5, 2.23
trailer, 4, 1.5, 2.2, 1.83
radio, 3, 1.6, 2.6, 1.0, 10.68, 3.2

Every piece of equipment must have one of the preceding names associated with

it, so that each will have nuclear vulnerability data. (Note: These numbers are being used for illustrative purposes only.) The first three pieces of equipment are susceptible to blast (indicated by the 4). The last is susceptible to the electromagnetic pulse (EMP) and neutron fluence (indicated by the 3). The remaining numbers are the criteria by which AURA determines whether or not the equipment has been killed by the environment to which it was susceptible.

The nuclear module of the AURA code is started with the detonation of a nuclear weapon (i.e. lethality event). First, subroutine NUCDMG is called to calculate the probability of kill (PK) from blast and thermal effects at each target point. NUCDMG determines the environment level at each target point (blast or thermal) by calling subroutine NUCENV. Then, to calculate these PKs, the posture of the target point must be considered. For exposed personnel (i.e. postures (1) and (2)), USANCA blast and thermal kill criteria (see subroutine NUCDMG) are used to calculate the PKs. (Note: Only those personnel in protective posture (1) receive thermal effects.) Personnel in postures (3), (4) and (5) are treated as 100% exposed to the blast effects; whereas posture (6) personnel are given the same blast criteria as the trailer and therefore will have the same PK as the trailer. The PK for equipment is calculated by calling the appropriate probability of survival (1. - PK) subroutine (PSNF for neutron fluence, PSDPIQ for blast, PSTHRM for thermal fluence and/or PSEMP1 for electromagnetic pulse). For personnel and equipment, the PKs for the various environments are combined using the survivor rule. Finally, subroutine NUCLTH is called to calculate lethality. The number of surviving assets at each target point is multiplied by the total PK (for that target point) to determine the number of lethality.

Since radiation effects are time dependent, the nuclear module assesses effects on personnel at each reconstitution event following the first (if more than one) lethality event. First, subroutine CUMDOS is called to update the radiation dose bins (see Subroutine CUMDOS) which categorize personnel according to their dose level, asset type and susceptibility to radiation effects (personal hardness value). Then, subroutine NUCDTH is called to assess radiation effects on personnel. For each dose bin, subroutine RADDTH is called to determine if those individuals (in the bin) are lethality from having received such a dose. For survivors subroutine ETI is called to determine if the individuals are experiencing early transient incapacitation which is associated with specific levels of radiation dosage. Individuals experience temporary periods of comatose-like unresponsiveness interspersed with periods of activity lasting until eventual death. Finally, for those individuals that have received a dosage, but are not dead or experiencing ETI, subroutine DGDSTM is called to assess their degraded effectiveness.

Typical AURA output from a nuclear run consists of asset (equipment and personnel) damage reports and unit effectiveness as a function of time. For equipment, the number of remaining assets of each type are reported for each reconstitution event. Also, a table is output which reports the number of pieces of equipment that are initially deployed, unharmed, contaminated, received light damage

and medium damage for each piece of equipment. For personnel, the number of casualties for each reconstitution event (for all personnel) and the number of remaining personnel (for each asset type) at each reconstitution event are output. In addition, a dose table is output which reports the number of personnel (by asset identification) that have received specific dosage levels. These levels are grouped into twelve specific intervals (0-5, 5-75, 75-150, 150-300, 300-450, 450-530, 530-830, 830-1100, 1100-1500, 1500-4500, >8000 rads). Included in this table is the number of lethalties due to radiation, blast and thermal (separately) for each asset type. Finally, unit effectiveness and the frequency (number of replications) distribution of the effectiveness are reported at each reconstitution time.

The purpose of this section was to present an example of an AURA run using the nuclear vulnerability module. First, the proper inputs for such a run were described, as well as, AURA's methodology for determining nuclear lethalties and sublethal effects. Finally, an example of the type of output available when making a nuclear run were presented. This example in no way exhausts AURA's nuclear capabilities, but rather shows a typical nuclear scenario.

5. Combined Weapons/Lethality Effects in AURA

There is a limited capability within the AURA model to play the combined weapons effects of different lethality types, that is, one weapon having two lethality effects. Currently, the only combined weapons effect playable is conventional/chemical. AURA is not configured to allow both chemical and nuclear attacks in the same scenario nor can one play a combined chemical/nuclear weapon. (In order to play both chemical and nuclear weapons against a unit, one would have to make a run with one attack, attrite the unit accordingly, and then play the second attack in a separate run, using the attrited unit.)⁴² In addition, although conventional and nuclear attacks could be played in the same runstream, this type of combined attack has not been modeled to date.

A combined conventional/chemical weapon is indicated by giving the weapon both "conventional" and "chemical" as secondary names for the weapon under the NAMES mnemonic. Use of these secondary names invokes certain defaults for the weapon. For example, with a conventional weapon, the code will check the conventional files to make sure there is lethality data for everyone in

⁴² Juarascio, S.S., et al, "Land Warfare Systems Vulnerability Program (LSVP), Chemical and Combined Nuclear/Chemical Personnel Attrition Analysis: Europe V, VII Corps", USA Ballistic Research Laboratory Technical Report BRL-TR- 3071, January 1990, (SECRET-NOFORN).

the unit; with a chemical weapon, the code will check for an associated agent type. As with non-combined weapons, the TOXIC LETHALITY and CONVENTIONAL LETHALITY mnemonics must be included in the input runstream to alert the code to look for these lethality files. The inputs required for individual conventional or chemical weapons are also required when playing combined weapons effects.

Although a combined conventional/chemical weapon can be played, AURA does not currently model synergistic effects. The effects of this type of weapon are handled separately as they would be for individual conventional or chemical weapons. For example, a person in MOPP4 would still be fully protected from the chemical hazard even though he may have been struck by conventional fragments and his protective ensemble compromised. The protective ensemble also does not provide any protection from fragments unless it has been specified in the conventional lethality file. Similarly, the probability of lethality for personnel does not increase in the combined scenarios. That is, a person with a fragment wound and a chemical dosage will not become a casualty faster because he is suffering from the effects of both kill mechanisms.

VI. Current Efforts in AURA Methodology

Current work at the Ballistic Research Laboratory is directed toward the development/improvement of AURA methodologies for disciplines such as:

- Evaluation of capability/effectiveness of 'smart' munitions;
- Improvement to chemical lethality model by incorporating a non-linear internal chemical dosage methodology;
- Improved model for personnel degradation due to heat stress.

This work, as well as changes to the current AURA methodologies, will be reported in periodic addendums to this report.

VII. Summary

This, the first of a three volume report, has presented the programmer/analyst with a comprehensive understanding of the methodologies which embody the AURA model. The approach taken was to progress from a general overview of the AURA model to a detailed description of the derivation, capability and primary algorithms for each AURA methodology. Throughout the report, a simple, hypothetical combat support unit is used as a 'working' example to describe the role of each methodology in the overall simulation process.

Volume 2 of this report contains a detailed description of the organization and conventions of the FORTRAN source code that embodies the AURA methodology.

Volume 3 will contain an in-depth description of the conduct of an AURA analysis, from data preparation through output analysis, as presented from an analyst's perspective.

Finally, the auxiliary/utility programs which support the AURA methodology are currently being documented by BRL analysts. When completed, the report detailing the source code and methodologies of these programs will provide a companion to this set of reports.

REFERENCES

1. "Computer Program for General Full Spray Material MAE Computations", Joint Technical Coordinating Group for Munitions Effectiveness, 61 JTCG/ME-79-1-1, (SECRET)
2. Vault, W.L., "Vulnerability Data Array: The Agreed Data Base - Final Report (U)," Harry Diamond Laboratories, HDL-TR-1906, JUL 80, (SECRET).
3. Saucier, R., "NUSSE3 Model Description", U.S. Army Chemical Research, Development and Engineering Center, CRDEC-TR-80746, May 1987, (UNCLASSIFIED).
4. Klopacic, J.T., "Input Manual for the Army Unit Resiliency Analysis (AURA) Methodology: 1988 Update", USA Ballistic Research Laboratory, Technical Report No. 2914, MAY 88, (UNCLASSIFIED)
5. Stark, M.M., Klopacic, J.T., "The Resiliency of an Ammunition Supply Point to Combat Damage (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2614, December 1984, (SECRET).
6. Stark, M.M., Klopacic, J.T., "The Resiliency of Ammunition Supply Points in a Pre-Defined, Integrated Battlefield Scenario (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2609, November 1984, (SECRET).
7. Roach, L.K., "The Resiliency of a Supply and Service Company to Combat Damage (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2669, August 1985, (SECRET).
8. Juarascio, S.S., "Evaluation of an 8-Gun M109A2 Artillery Battery with Replacement of Combat Damaged Mission Essential Components (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2682, October 1985, (SECRET).
9. Juarascio, S.S., Abell, J.M., "Impact of Chemical Survivability Criteria on Unit Performance (U)", USA Ballistic Research Laboratory Report No. BRL-TR-2870, November 1987, (SECRET).
10. Stark, M.M., Klopacic, J.T., Juarascio, S.S., "The Effects of Chemical Contamination/Decontamination of a M109A2 8-Gun Artillery Battery (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2633, February 1985, (SECRET).
11. Roach, L.K., "The Effects of Chemical Contamination/Decontamination on an M1 Tank Company (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2723, April 1986, (SECRET).
12. Roach, L.K., Juarascio, S.S., "The Resiliency of Selected Soviet Units to Candidate Chemical Warheads for Use in the Corps Support Weapon System (CSWS) (U)", USA Ballistic Research Laboratory Technical Report No. BRL-TR-2724, April 1986, (SECRET).
13. Stark, M.M., Klopacic, J.T., op. cit., BRL-TR-2614

14. Baum, W., "Theatre Nuclear Force Combat Capability Degradation Methodology, Development and Demonstration", (U), 2 Volumes, Interim Note No. SV-7, April 1978, US Army Materiel Systems Analysis Activity, Aberdeen Proving Ground, Maryland. (SECRET)
15. Dore, M.A., Anno, G.H., "Effects of Ionizing Radiation on the Performance of Selected Tactical Combat Crews", (U), Pacific-Sierra Research Corporation under contract to the Defense Nuclear Agency, PSR Report No. 1846, Contract No. DNA 001-85-C-0352, July 1988, (UNCLASSIFIED)
16. Lawler, E., "Combinatorial Optimisation: Networks and Matroids", Holt, Rinehart and Winston (1975), (UNCLASSIFIED).
17. Papadimitriou, C. and Steiglitz, K., "Combinatorial Optimization: Algorithms and Complexity", Prentice Hall (1982), (UNCLASSIFIED).
18. op cit.
19. op cit.
20. op cit.
21. op cit.
22. Stark, M.M., Klopac, J.T., "The Resiliency of an Ammunition Supply Point to Combat Damage(U)", USA Ballistic Research Laboratory Technical Report BRL-TR-2614, December 1984, (SECRET).
23. Klopac, J.T., "The AURA Fatigue and Heat Stress Algorithms", USA Ballistic Research Laboratory Memorandum Report No. BRL-MR-3802, December 1989, UNCLASSIFIED.
24. op cit.
25. Groves, Art, "Handbook On The Use of The Bivariate Normal Distribution In Describing Weapon Accuracy(U)", Memorandum Report No. 1372, USA Ballistic Research Laboratory, September 1961, (UNCLASSIFIED)
26. Myers, K.A., "Lethal Area Description", BRL Technical Note No. 1510, July 1963, (UNCLASSIFIED)
27. "Performance Characteristics of Artillery and Mortar Systems", Handbook Series G, No. 5, Army Materiel Systems Analysis Activity, 1985, (SECRET-NOFORN).
28. Juarascio, S.S., "Evaluation of an 8-GUN M109A2 Artillery Battery with Replacement of Combat Damaged Mission Essential Components(U)", BRL-TR-2682, October 1985, US Army Ballistic Research Laboratory, Aberdeen Proving Ground, Md., (SECRET).
29. "Simplified Artillery Projective Effectiveness Model - Artquick Computer Program, User Manual", 61 JTCG/ME-81-8, Oct 1980, Unclassified.

30. "Computer Program for General Full Spray Materiel MAE Computations, Volumes 1 and 2", 61 JTCG/ME-79-1-1, January 1979, (UNCLASSIFIED).
31. Saucier, R., "NUSSE3 Model Description", U.S. Army Chemical Research, Development and Engineering Center, CRDEC-TR-80746, May 1987, (UNCLASSIFIED).
32. Saucier, R., "NUSSE3 User's Guide and Reference Manual", U.S. Army Chemical Research, Development and Engineering Center, CRDEC-SP-86009, March 1986, (UNCLASSIFIED).
33. Litchfield, J.T. and Fertig, J.W., "On a Graphical Solution of the Dosage-Effect Curve", Bulletin: Johns Hopkins Hospital, Volume 69, pp. 276-286, 1941, (UNCLASSIFIED).
34. Vault, William L., "Vulnerability Data Array: The Agreed Data Base - Final Report (U)," Harry Diamond Laboratories, HDL-TR-1906, (July 80), (SECRET).
35. Klopacic, J.T., Watson, LTC D.L., "Modeling Casualties in Nuclear Warfare", USA Ballistic Research Laboratory Memorandum Report No. BRL-MR-3771, July 1989, (UNCLASSIFIED).
36. Kelley, C.S., et al., "Nuclear Damage to Point Targets," Harry Diamond Laboratories, HDL-TR-1876, December 1978, (UNCLASSIFIED)
37. Jordan, D., "Weapon Effects ROM, Volume II-Reference Handbook," Horizons Technology Inc., prepared for Defense Nuclear Agency, DNA-EH-84-01-A-V2, 31 May 1984, (UNCLASSIFIED)
38. Kelly, C.S., et al. op cit.
39. Private communication with Bill Jackson of AFRRI.
40. Private communication with Bill Jackson of AFRRI.
41. Dore, M.A., Anno, G.H., Wilson, M.A., "Acute Radiation Effects on Individual Crewmember Performance", (U), Pacific-Sierra Research Corporation under contract to the Defense Nuclear Agency, DNA-TR-85-52, Contract No. DNA 001-83-C-0015, August 1984, (UNCLASSIFIED)
42. Juarascio, S.S., et al, "Land Warfare Systems Vulnerability Program (LSVP), Chemical and Combined Nuclear/Chemical Personnel Attrition Analysis: Europe V, VII Corps", USA Ballistic Research Laboratory Technical Report BRL-TR- 3071, January 1990, (SECRET-NOFORN).

INTENTIONALLY LEFT BLANK.

<u>No of Copies</u>	<u>Organization</u>	<u>No of Copies</u>	<u>Organization</u>
1	Office of the Secretary of Defense OUSD(A) Director, Live Fire Testing ATTN: James F. O'Bryon Washington, DC 20301-3110	1	Director US Army Aviation Research and Technology Activity ATTN: SAVRT-R (Library) M/S 219-3 Ames Research Center Moffett Field, CA 94035-1000
2	Administrator Defense Technical Info Center ATTN: DTIC-DDA Cameron Station Alexandria, VA 22304-6145	1	Commander US Army Missile Command ATTN: AMSMI-RD-CS-R (DOC) Redstone Arsenal, AL 35898-5010
1	HQDA (SARD-TR) WASH DC 20310-0001	1	Commander US Army Tank-Automotive Command ATTN: AMSTA-TSL (Technical Library) Warren, MI 48397-5000
1	Commander US Army Materiel Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333-0001	1	Director US Army TRADOC Analysis Command ATTN: ATAA-SL White Sands Missile Range, NM 88002-5502
1	Commander US Army Laboratory Command ATTN: AMSLC-DL Adelphi, MD 20783-1145	(Class. only) 1	Commandant US Army Infantry School ATTN: ATSH-CD (Security Mgr.) Fort Benning, GA 31905-5660
2	Commander US Army, ARDEC ATTN: SMCAR-IMI-I Picatinny Arsenal, NJ 07806-5000	(Unclass. only) 1	Commandant US Army Infantry School ATTN: ATSH-CD-CSO-OR Fort Benning, GA 31905-5660
2	Commander US Army, ARDEC ATTN: SMCAR-TDC Picatinny Arsenal, NJ 07806-5000	1	Air Force Armament Laboratory ATTN: AFATL/DLODL Eglin AFB, FL 32542-5000
1	Director Benet Weapons Laboratory US Army, ARDEC ATTN: SMCAR-CCB-TL Watervliet, NY 12189-4050		<u>Aberdeen Proving Ground</u>
1	Commander US Army Armament, Munitions and Chemical Command ATTN: SMCAR-ESP-L Rock Island, IL 61299-5000	2	Dir, USAMSAA ATTN: AMXSY-D AMXSY-MP, H. Cohen
1	Commander US Army Aviation Systems Command ATTN: AMSAV-DACL 4300 Goodfellow Blvd. St. Louis, MO 63120-1798	1	Cdr, USATECOM ATTN: AMSTE-TD
		3	Cdr, CRDEC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-MSI
		1	Dir, VLAMO ATTN: AMSLC-VL-D

<u>No. of Copies</u>	<u>Organization</u>	<u>No. of Copies</u>	<u>Organization</u>
1	Office of the Secretary of Defense Joint Chemical Warfare Joint Test Force Suite 300, Five Skyline Place 5111 Leesburg Pike Falls Church, VA 22041	1	Commander HQ AMCCOM ATTN: AMSMC-IMP-IL Rock Island, IL 61299-7300
10	C.I.A ATTN: OIR/DB/Standard GE-47 HQ Washington, DC 20505	1	Commander U.S. Army Aviation Systems Command ATTN: AMSAV-ES 4300 Goodfellow Blvd. St. Louis, MO 63120-1798
2	Director Defense Nuclear Agency ATTN: STBE, Dr. D. Auton Dr. R. Young Washington, DC 20305	1	Commander, USACECOM R&D Technical Library ATTN: ASQNC-ELC-I-T, Myer Center Fort Monmouth, NJ 07703-5301
2	Commander Defense Nuclear Agency/Field Command ATTN: FCPR, MAJ Fleming CPT Thorton Kirtland AFB, NM 87115	5	Commander U.S. Army Harry Diamond Laboratories ATTN: SLCHD-NW-RA 2800 Powder Mill Road Adelphia, MD 20783
3	DA/ODCSOPS Pentagon ATTN: DAMO-ZDF, Mr. Wolpert DAMO-ZXG, Mr. Patterson DAMO-ZDF Washington, DC 20310-0404	2	Director U.S. Army Missile and Space Intelligence Center ATTN: AIAMS-YDL AIAMS-TSL Redstone Arsenal, AL 35898-5500
1	Commander U.S. Army Materiel Command ATTN: AMCDRA-AT 5001 Eisenhower Avenue Alexandria, VA 22333-0001	11	Commander TRAC-WSMR ATTN: ATAA-SL, Mr. Pena Mr. Kirby Mr. Fernandez Mr. Billingsley Mr. Belk Mr. Leach Mrs. Hodde Mr. Baldauf ATRC-WCC, CPT Greg Stover CPT David Jones ATOR-TSL White Sands Missile Range, NM 88002-5522
2	Commander U.S. Army ARDEC ATTN: SMCAR-FSS, Mr. Ostuni Mr. Brooks Picatinny Arsenal, NJ 07806-5000		

<u>No. of Copies</u>	<u>Organization</u>
2	Director TRAC-FLVN ATTN: ATRC-FTC, CPT Perry ATRC-F, Dr. Robert LaRocque Ft. Leavenworth, KS 66027
1	Director TRAC RPD ATTN: ATRC-RPR, Mr. Radda Ft. Monroe, VA 23651-5143
2	Commander U.S. Army Aviation Logistics Center ATTN: ATSP-CD, CPT White Mr. Pollack Ft. Eustis, VA 23604
3	Commander U.S. Army Combined Arms Combat Developments Agency ATTN: ATZL-CAS-SP, CPT DeWulf ATZL-CAP, COL Henderson CPT Grand Ft. Leavenworth, KS 66027
2	U.S. Army Nuclear and Chemical Agency ATTN: MONA-WE MONA-CM, MAJ Joe Fernandez 7500 Backlick Road, Bldg 2073 Springfield, VA 22150-3198
1	Director TRAC-Ft. Lee ATTN: ATRC-OS Ft. Lee, VA 23801-6000
3	Director Concepts Analysis Agency ATTN: CSCA, Mr. D. Hurd LTC Barrett Mr. Chandler 8120 Woodmont Avenue Bethesda, MD 20014

<u>No. of Copies</u>	<u>Organization</u>
1	Commandant U.S. Army Air Defense Artillery School ATTN: ATZK-CD Bldg 5800 Ft. Bliss, TX 79916
1	Commandant U.S. Army Armor School ATTN: ATZK-CP-SD, Mr. Hekemeyer Ft. Knox, KY 40121
1	Commander U.S. Army Engineering School ATTN: ATSE-CDM Ft. Leonard Wood, MO 65473-6620
1	Commander U.S. Army Field Artillery School ATTN: ATSF-CD Ft. Sill, OK 73503
1	Commandant U.S. Army Transportation School ATTN: ATSP-PD-C Ft. Eustis, VA 23604
2	Commander U.S. Army Missile and Munitions Center and School ATTN: ATSK-CD Redstone Arsenal, AL 35898
1	Commander U.S. Army Quartermaster School ATTN: ATSM-CDC Ft. Lee, VA 23801
1	Commander U.S. Army Signal Center & School Directorate of Combat Developments Ft. Gordon, GA 30905

<u>No. of Copies</u>	<u>Organization</u>	<u>No. of Copies</u>	<u>Organization</u>
4	Commandant U.S. Army Chemical School Directorate of Combat Developments ATTN: ATZN-CM-CC, LTC Gary Stratton Mr. B. Gould Ms. Sherry Barnes Ms. Rena Seals Ft. McClellan, AL 36201	3	Institute for Defense Analysis ATTN: Mr. Graham McBryde Mr. M. Kerlin Dr. M. Charren 1801 N. Beauregard Street Alexandria, VA 22311
1	Commander Armed Forces Radiobiological Research Institute National Naval Medical Center ATTN: Mr. William Jackson Bethesda, MD 20014	3	U.S. Army Personnel Command ATTN: DAPC-MOP, H. Rowland Ludden TAPC-MOP, MAJ M. Brice Elliot CPT I. McMahon 200 Stovall Street Alexandria, VA 22332-0432
2	Commander Combined Arms Center Scores Working Group ATTN: ATZL-CAD-LN Ft. Leavenworth, KS 66027	1	U.S. Army Chief of Staff ATTN: DACS-DPZ-A The Pentagon Washington, DC 20310
2	Commander Naval Surface Warfare Center ATTN: Code 31-TJY, Mr. Yench Mr. P. Kirk Dahlgren, VA 22448	3	U.S. Army Deputy Chief of Staff for Operations and Plans ATTN: DAMO-SWC, LTC Gordon Miller DAMO-ZD, Mr. John Riente Mr. James Metzger The Pentagon Washington, DC 20310
1	President Center for Naval Analyses 4401 Ford Avenue P.O. Box 16268 Alexandria, VA 22302-0268	1	Defense Nuclear Agency 6801 Telegraph Road Alexandria, VA 22310
2	Walter Reed Army Institute of Research Division of Medicine Department of Respiratory Research ATTN: SGRD-UWH-E, MAJ Gary R. Ripple, MD LTC Kenneth Torrington Washington, DC 20307-5100	2	Office of the Surgeon General 5111 Leesburg Pike ATTN: DASG-HCO-P, MAJ Ray Keith DASG-HCD, MAJ Robert Eng Falls Church, VA 22041
		1	U.S. Army Engineer Studies Center ATTN: ESC-EO Casey Bldg 2594 Ft. Belvoir, VA 22060-5583

<u>No. of Copies</u>	<u>Organization</u>	<u>No. of Copies</u>	<u>Organization</u>
1	Commanding General U.S. Army Operational Test and Evaluation Agency Park Center IV 4501 Ford Avenue Alexandria, VA 22302-1428	2	WRDC/FIVS/SURVIAC Wright-Patterson AFB, OH 45433-6553 <u>Aberdeen Proving Ground</u>
2	Director TRAC-MTRY Naval Postgraduate School ATTN: LTC Vernon M. Bettencourt, Jr. MAJ R. Wimberley P.O. Box 8692 Monterey, CA 93943-0692	6	Dir, USAMSAA ATTN: AMXSY, Mr. M. Miller Ms. C. Horley Ms. D. Smoot AMXSY-C, M. Carroll AMXSY-G, M. Sandmeyer AMXSY-J, R. Mezan
1	U.S. Army Command and General Staff College Office of the Commandant Ft. Leavenworth, KS 66027-6900	8	Cdr, CRDEC, AMCCOM ATTN: AMSTE-SI-F SMCCR-ST, Mr. D. Sloop Mr. R. Jablonski Mr. J. Razulis Mr. J. Walther Mr. C. Crawford Mr. L. Davis Mr. L. Baer
3	JAYCOR ATTN: Mr. J. David Claiborne Mr. Hubert Lockerd Mr. William Jenkins P.O. Box 87 Edgewood, MD 21040	2	Cdr, USAHEL ATTN: W. DeBellis O. Zubal
3	EAI Corporation ATTN: Mr. Dennis Metz Mr. Bruce Fischer Mr. S. Matthew Hutton 1308 Continental Drive, Suite J Abingdon, MD 21009	1	Cdr, USAOC&S ATTN: ATSL-CD-CS
2	Science Applications International Corporation ATTN: R. McNally M. Stark 626 Towne Center Dr. Joppa, MD 21085		
1	Battelle ATTN: D.W. Harper 7501 Memorial Parkway South Suite 101 Huntsville, AL 35802-2258		

INTENTIONALLY LEFT BLANK.

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. BRL Report Number BRL-TR-3156 Date of Report OCTOBER 1990
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

Name

Organization

Address

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD
ADDRESS

Name

Organization

Address

City, State, Zip Code

(Remove this sheet, fold as indicated, staple or tape closed, and mail.)

-----FOLD HERE-----

DEPARTMENT OF THE ARMY

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-5066
OFFICIAL BUSINESS

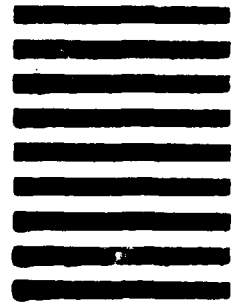


NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT No 0001, APG, MD

POSTAGE WILL BE PAID BY ADDRESSEE

Director
U.S. Army Ballistic Research Laboratory
ATTN: SLCBR-DD-T
Aberdeen Proving Ground, MD 21005-9989



-----FOLD HERE-----