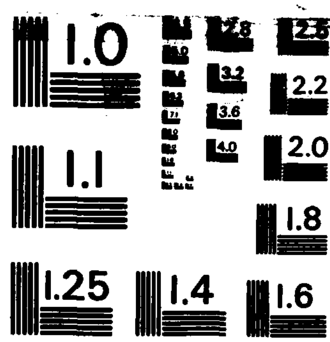
AD-A141 219 AAPMOD: AN INTERACTIVE COMPUTER MODEL FOR ANALYSIS OF CONVENTIONAL WEAPON..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. R N MIGLIN 1/2

UNCLASSIFIED MAR 84 AFIT/GST/OS/84M-13 F/G 15/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

"AAPMOD",
AN INTERACTIVE, COMPUTER MODEL
FOR ANALYSIS
OF CONVENTIONAL WEAPONS EFFECTIVENESS

THESIS

AFIT/GST/OS/84M-13 Robert N. Miglin
Captain USAF

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	**1b. RESTRICTIVE MARKINGS**
2a. SECURITY CLASSIFICATION AUTHORITY **2b. DECLASSIFICATION/DOWNGRADING SCHEDULE**	**3. DISTRIBUTION/AVAILABILITY OF REPORT** Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GST/OS/84M-13	**5. MONITORING ORGANIZATION REPORT NUMBER(S)**
6a. NAME OF PERFORMING ORGANIZATION School of Engineering **6b. OFFICE SYMBOL** *(If applicable)* AFIT/ENS	**7a. NAME OF MONITORING ORGANIZATION**
6c. ADDRESS *(City, State and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB OH 45433	**7b. ADDRESS** *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION **8b. OFFICE SYMBOL** *(If applicable)*	**9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER**
8c. ADDRESS *(City, State and ZIP Code)*	**10. SOURCE OF FUNDING NOS.** PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE *(Include Security Classification)* See Box 19	

12. PERSONAL AUTHOR(S)
Robert N. Miglin, B.S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1984 March 13	175

16. SUPPLEMENTARY NOTATION

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER 7 May 84
Dean for Research and Professional Development

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
01	05	
09	04	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Conventional Weapons, Weapon Effectiveness Model, Airbase Damage Assessment, Attack Simulation

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Title: "AAPMOD", AN INTERACTIVE COMPUTER MODEL FOR ANALYSIS OF CONVENTIONAL WEAPONS EFFECTIVENESS

Thesis chairman: James R. Coakley, Major, USAF

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	**21. ABSTRACT SECURITY CLASSIFICATION** UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
James R. Coakley, Maj, USAF	513-255-3362	AFIT/ENS

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

This research effort was directed towards developing a flexible, operationally oriented methodology to assess the effectiveness of conventional weapons. Ease of use has been stressed, to enable aircrews and weapons experts to use the methodology.

The methodology centers on an interactive computer program, AAPMOD, that simulates a user defined attack against a user defined target. The program is a derivative of Attack Assessment Program, originally developed by the University of Oklahoma for the Joint Technical Coordinating Group for Munitions Effectiveness. This study provided the program interactive capability, improved its structure by adding Fortran V constructs, and developed a data-input program AAPIN, to provide laundered input files for AAPMOD.

Program outputs include probabilities of cutting surfaces and denying aircraft operations, as well as expected values for number of hits and area damaged.

Validity and capability of AAPMOD are demonstrated in a three factor, two level statistical experiment. The experiment consisted of an airfield attack, with associated discussion of effects of the three factors.

AFIT/GST/OS/84M-13

"AAPMOD", AN INTERACTIVE, COMPUTER MODEL FOR ANALYSIS OF CONVENTIONAL WEAPONS EFFECTIVENESS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the Degree of
Master of Science

by

Robert N. Miglin
Captain USAF

Graduate Strategic and Tactical Sciences

March 1984

Preface

This project is dedicated to the young men who fly jets, drop bombs, and drink beer. Sometimes considered a "resource", or a number, these men risk their lives for the ideals for which America stands. While Congress and the media talk about the weapons these men could have, the men train with the weapons they do have. And if called upon, they must *fight* with the weapons they *do have*. Hopefully, this research will help them fight more effectively.

I am grateful to my advisor, Major James R. Coakley, for his assistance and direction. Major Coakley is a fighter pilot, but he kept this '*gator* on course.

I also appreciate the advice and consultation of many of the professors at AFIT, especially Lt Col Ivy Cook, my thesis reader, and the other instructors of the Operational Sciences department. These men have often inspired a group of emerging analysts. I am proud to have learned *from* them, and later will be proud to work *with* them.

Finally, I extend special thanks to Mr. Dan McInnis, Mr. Jerry Bass, and Mr. Elijah Green, all from the Armament Development Laboratory, Eglin AFB, Florida. Not only did they provide me a copy of Attack Assessment Program, their guidance throughout program conversion was invaluable.

It goes without saying that my deepest appreciation and love remains with my wife, Susan. Not only was she a typist, throughout these efforts she has provided me comfort and encouragement. Susan married me during my AFIT studies. We *know*, with Our Lord's help, we can make it through everything!

Robert N. Miglin

Contents

List of Figures

List of Tables

Abstract

This research effort was directed towards developing a flexible, operationally oriented methodology to assess the effectiveness of conventional weapons. Ease of use has been stressed, to enable aircrews and weapons experts to use the methodology.

The methodology centers on an interactive computer program, AAPMOD, that simulates a user-defined attack against a user-defined target. The program is a derivative of Attack Assessment Program, originally developed by the University of Oklahoma for the Joint Technical Coordinating Group for Munitions Effectiveness. This study provided the program interactive capability, improved its structure by adding Fortran V constructs, and developed a data-input program, AAPIN, to provide laundered input files for AAPMOD.

Program outputs include probabilities of cutting surfaces and denying aircraft operations, as well as expected values for number of hits and area damaged.

Validity and capability of AAPMOD are demonstrated in a three factor, two level statistical experiment. The experiment consisted of an airfield attack, with associated discussion of effects of the three factors.

I. Introduction

This study will determine the effectiveness of a modern, fighter-attack aircraft, delivering conventional munitions against a runway. The measure of effectiveness is the probability the aircraft denies the clear length and width of runway surface, required for take-off and land operations.

Background

Tactical aviation is a vital part of the firepower the United States can muster against an enemy. Also called tac air, the intrinsic characteristics of tactical aviation include elements of surprise, mass, and even flexibility. Given that the enemy will choose the time and place of the next conflict, tactical air power offers fast, concentrated response to aggression, and offers in-place ground units a better chance to maintain position until reinforcements arrive.

As with all resources, the availability of tac air is limited. Furthermore, modern air power faces an increasingly sophisticated enemy defense network. In recent years potential enemies have improved air defense networks with the deployment of new missiles, guns, radars, and aircraft. Along with these deployments has been the employment of a new command and control system. (Ref.15:19)

The allocation of the limited resources of tactical air, throughout the hostile arena, is therefore a decisive element in the success of combat operations. And the task is not easy.

For example, consider the European theater. The commanders, whether assigned to USAFE or NATO, allocate their aircraft in three phases: 1)identification of targets, 2)prioritization of targets, and finally, 3)the assignment of assets against the targets. Each phase will be briefly discussed.

There are several ways to identify targets. Prior to conflict, intelligence personnel can study potential hot-spots and identify targets of obvious military value. Munitions or tactics experts can then recommend particular attack options. Such preparation can permit development of preplanned attacks, and save valuable time when the war breaks out. Similarly, during the fight, planners, aircrews, intelligence sources, or even the battlefield commander can recommend additional targets. But, as the list grows, it soon exceeds the number of aircraft available to cover the targets. This target-rich situation requires that commanders prioritize targets.

Prioritization occurs when a commander decides which targets should be attacked first. But target priorities are dynamic, and influenced by perspective. For example:

> 1) The Army commander, repelling an armor assault, thinks the attacking column has highest priority.
>
> 2) The commander at Ramstein thinks 24 FLOGGER's massing in western Czechoslovakia have the highest priority.
>
> 3) All agree that denial of chemical or nuclear potential is a high priority at all times.

Regardless, targets should be struck in such an order that they maximize the damage inflicted *on the enemy*, and minimize the damage inflicted *by the enemy.*

In the past, prioritization has been an art. But today, it can more properly be termed a science. Rigorous techniques, developed under the broad spectrum of *operations research*, are frequently applied to military decision making or problem solving. Some of these techniques have included network theory, applied to aircraft movements, various types of programming methods, applied to weapon buys and prioritization processes, and computer simulations of nearly all phases of combat operations. This thesis, itself, applies one of the tools of operations research, computer modeling, to the last phase of the allocation process: resource assignment.

The last phase of the allocation process is to assign a specific weapon system to a specific target. Experience shows that thorough analysis is sometimes absent from this phase. Understandably, the crisis of the situation may inhibit logical, optimum allocation. But at other times, aircraft are merely assigned targets based on geographical sectors: *this* wing covers the targets of *that* sector. Although range or other performance characteristics must influence allocation decisions, the convenient grounds of:

"... the only one available ..."

should not. Each type of aircraft has its own performance advantages and capabilities, as well as disadvantages and limitations. For example, one aircraft may have poor maneuverability, but excellent payload capacity. Another may offset small payloads with high accuracies. To arbitrarily assign a weapon system against a target, without considering the effectiveness of the aircraft against the target is absurd. It can negate the efforts of the previous phases, and can contribute to unnecessary loss of life or other valuable assets.

Some of the methods of operations research (OR) may serve to reduce the effort associated with the allocation process. These methods may also enhance the results of the process. The rest of this chapter will develop the framework within which modern OR can improve United States defense capabilities.

Operations Research. Operations Research tries to blend the skills of many, varied, science and military experts and optimize our defense capabilities. The first use of OR dates back to 1943.

During World War II, British and American scientists tried to describe and predict the way two armies act. They modeled the allocation of scarce resources, and contributed to Allied victories in several campaigns: such as the Air Battle of Britain, and the Island Campaign in the Pacific. (Ref.9:3)

Today, nearly every level of command in the U.S. Air Force has an operations research branch, even though the size of the branch may vary. For example, Air Force Headquarters has a 192-person Studies and Analysis Division (AF/SA). On the other hand, a typical, tactical fighter squadron (TFS) might have only a 3-person, additional-duty, plans and analysis working group. Nevertheless, both organizations provide decision support to their commander. AF/SA analyzes major weapon system alternatives, or perhaps force employment plans, while planners in the TFS optimize delivery tactics when two or three aircraft attack a target.

Some indications suggest, however, that current analysis techniques may fall short of their full potential. For example, during an exercise in Europe, Headquarters, United States Air Forces Europe (USAFE), tasked a fighter wing to plan suppression of an enemy airbase. USAFE limited the number of aircraft to be used for the attack. Intelligence personnel and weapons experts hastily worked to develop an *optimal* attack plan. They targeted storage sites, defense positions, repair facilities, and the runway. But before submitting the plan, the commander wanted some experienced aircrews to verify the plan. The crews questioned the feasibility of targeting two aircraft against an enemy runway. The planners responded that only two sorties were left after "optimal" targeting, and they decided *some* damage to the hardened runway was better than none.

Is it?

Should a commander risk damage to, or loss of, two aircraft and crews to attack the runway?

Another indication reflects an even higher level of authority. In a recent lecture at AFIT, Brigadeer General Wilfred L. Goodson, Assistant Chief of Staff, Studies and Analysis, Headquarters, USAF, expressed dissatisfaction with the current approach of models used in analysis. (Modeling is a frequent tool of analysis. Models quantify often elusive characteristics of a system, and the numbers are used to develop mathematical relationships, describing the system.) General Goodson feels that today's theater-level warfare models lack proper sensitivity to new data. (Ref.5) For example, if analysts input a new capability to their model, they usually do not adjust the enemy's response to the change. While, in reality, if a capability enters a theater prior to conflict, opponents will attempt to deny the advantages of the new system. They will develop, purchase, or deploy counter-weapons, or tactics. Likewise, *during* conflict, both sides adjust tactics and strategies in response to daily developments: their own effectiveness, or perhaps an enemy's surprise system.

But, according to experts, most models do not make such adjustments. (Ref.5) In essence, most models are inadequately sensitive to parameter changes. They do not modify target values, which in-turn modify strategies. Such modification requires a recursive, dynamic model design: a design difficult to achieve because it requires a well defined system of target values. And yet, according to other experts, sensitivity analysis is crucial to theater-level models. (Ref.10:132) No model could possibly portray war in all its complexity. Rather, models of theater-level warfare should be used to examine *alternative* systems, tactics, or force structures. (Ref.10:132) And so, the models should be *sensitive* to attribute modifications.

Finally, the last evidence supporting the inadequacy of modern analysis appeared in the tear sheet of a 1980, Comptroller General's *Report to the Congress*:

> A major contention of this report is that quantitative techniques have considerable potential as an aid in the analysis of public policy issues, but that this potential is impaired by the current design and management of quantitative tools....
>
> From a scientific point of view, the present "understanding of war"--insofar as the effectiveness of conventional military forces is concerned--is in relatively primitive state. Basic research aimed at understanding the fundamentals of combat is needed, but quantitative or numerical techniques have not been systematically applied to achieve these discoveries. (Ref.17:ii)

Consider runways again. How do the above ideas relate to runways? How does a decision maker answer the following questions: What is the value of a runway? Of what value is the damage two aircraft might inflict on a runway? How about four aircraft? Eight? More?

The ultimate answers to these questions are beyond the methodology of this study. Nevertheless, their discussion validates the need to develop the low-order, responsive, informative, targeting analysis described in this thesis. Although this analysis can not specifically assign target values, the study will help define the level of damage that two, or four, or eight aircraft can inflict on a runway.

Problem Statement

Current, operationally oriented, targeting analysis methods do not clearly illustrate the relationship between applied attack effort and target damage response.

Research Method

In response to the problem statement, this thesis will establish a methodology to rate the effectiveness of different elements of tac air against different targets. The methodology is examined within the framework of determining the effectiveness of a conventional attack against one type of target: runways. Of course, the methodology can be extended to cover the gamut of systems-target combinations, and valid comparisons of system effectiveness can be made.

Decision makers should implement these analyses before future conflicts erupt. Such preparation can afford greater overall effectiveness in the allocation of tac air. (NOTE: For purposes of this report, *weapon system* implies not only a type of aircraft, such as F-111 or F-16, but also a specific weapons load and delivery tactic. And, to avoid compromise, *generic* aircraft and *generic* weapons data will be used.)

Objectives Solution of the problem statement lies in developing an easy, clear methodology to relate given levels of attack effort to the damage the attack can produce. Such development suggests the following three objectives:

1) Develop a method to relate an attack to its expected damage results.

2) Define the significant factors affecting damage expectancies.

3) Develop a concise, clear method of presentation of results.

<u>Methodology</u> The methodology of this research follows from the objectives. A model will be developed to relate attack effort to expected damage.

Experience recommends a simulation over an analytical solution. As will be presented in Chapter III, the system of an airfield attack includes many complex interactions of numerous stochastic variables. And it was felt, that a purely mathematical analysis of expected value is beyond the scope of this research.

The completed model will be exercised in an experiment to demonstrate its operation and capability. The experiment will focus on three of the factors under aircrew control when planning an airfield attack. Manipulating these factors will provide data for the effectiveness study described above, as well as suggest the influence of the factors on system effectiveness.

Finally, the results of the effectiveness study will be clearly graphed. A series of these types of charts can be developed for possible use by aircrews during attack planning.

This chapter has developed the need to improve the methods for the optimal targeting of the limited assets of United States air power. The chapter recommends seeking solution within the science of operations research. The chapter summarized the problem at hand in a concise, clear, and limited statement, and proceeded to describe the research effort designed to correct the problem. Chapter II will discuss some of the earlier works preparing the way for this thesis.

II. Previous Studies

By no means is this the first study to identify the requirement to improve tacticians', aircrews', and commanders' understanding of the relationship between aerial attack efforts and target damage results. Projects, programs, and literature have addressed the issue, and this thesis will draw on those works and apply them to the methodology required to satisfy the problem statement of Chapter I. This chapter will highlight both strong and weak areas of some of these earlier works.

Theater-Level Warfare Models

In 1967, Air Force Studies and Analysis developed a tactical air warfare model, or TAWM, with a recursive, and dynamic, simulation concept. In other words, given a change to the model data, the change itself could cause other changes in the model. The model used a novel methodology that begins with the *last* day of the war, and moves backwards. The model optimizes each day, back through DAY-1. Optimization for the future occurs each day, regardless of the course of events followed to arrive at the current day.

It has been suggested, however, that a new and more responsive model for theater warfare be developed. (Ref.5) An important concept of the new model, call it TAWM84, will be the value of target damage. Once the many, continuous levels of target damage are quantified, TAWM84 will optimize warfighting strategy. The model will find the optimal return for investing the available attack resources.

But first, sub-models must clarify the relationship between level of attack and specific target damage. And value must be quantified.

The concensus of literature, though perhaps argued by some fighter pilots, is that the only *value* of air power is in support of the ground battle. With minor variation, both versions of TAWM use the following categories of air support:

1) attack aircraft on enemy airfields;
2) defend friendly airfields from enemy attack;
3) defend the airspace over the battlefield; and,
4) participate in combat air support.

Only combat air support might need further definition. Combat air support is basically *ground support*. Combat-air takes air power to the enemy. As the ground commander maneuvers and employs organic firepower against the enemy, combat-air provides additional aerial firepower. The targets of combat-air include war-fighting capability on the battlefield, such as vehicles, armor, or troops. The targets can also include the enemy's means to bring these capabilities to the fight: roads, rails, and bridges.

Both versions of the model use game theory. The *value* of air power is defined as support of ground operations. The payoff of the "game" is defined as the difference between the combat-air ordnance delivered by the opposing sides. The models use tonnage of ordnance, delivered in combat support, to measure value. Note how each category above, can contribute to this overall measure:

1) attack aircraft--denies enemy potential;

2) defend friendly airfields--prevents loss of friendly potential;

3) defend airspace--again, both denies enemy potential and preserves friendly potential; and

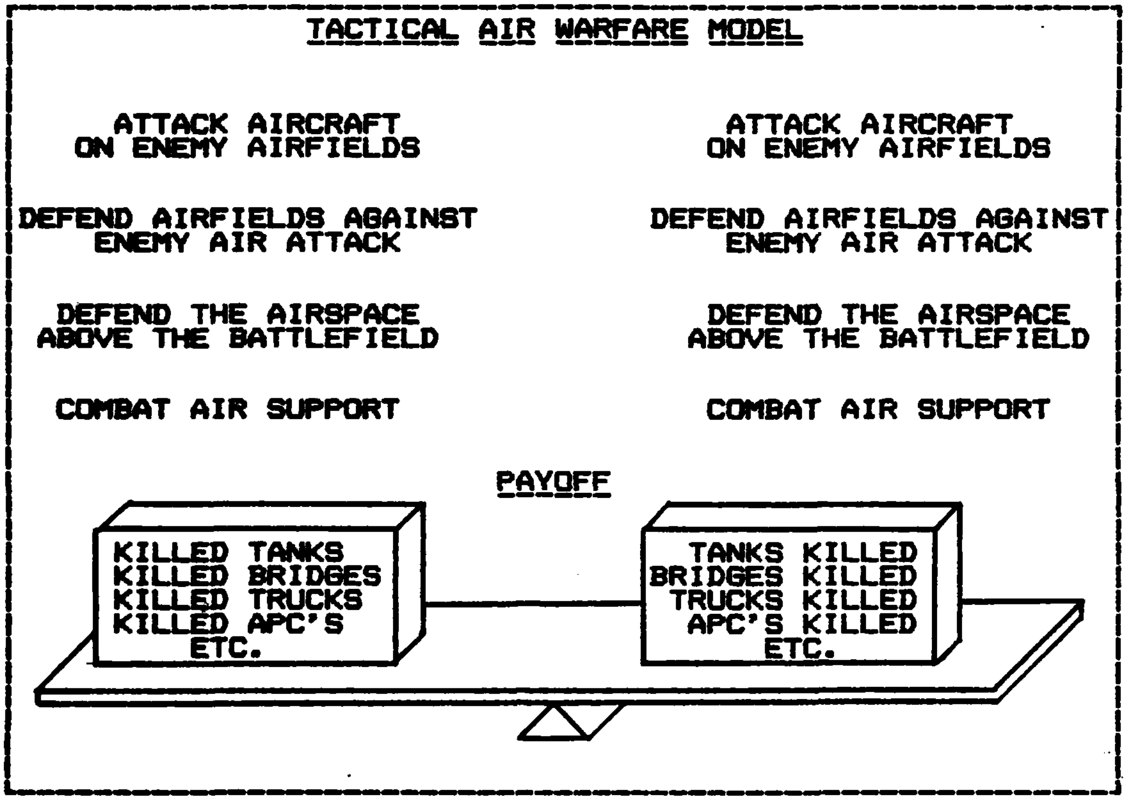


Figure 1 Overall Concept of Theater Air Warfare Model, 1967. (Ref.5)

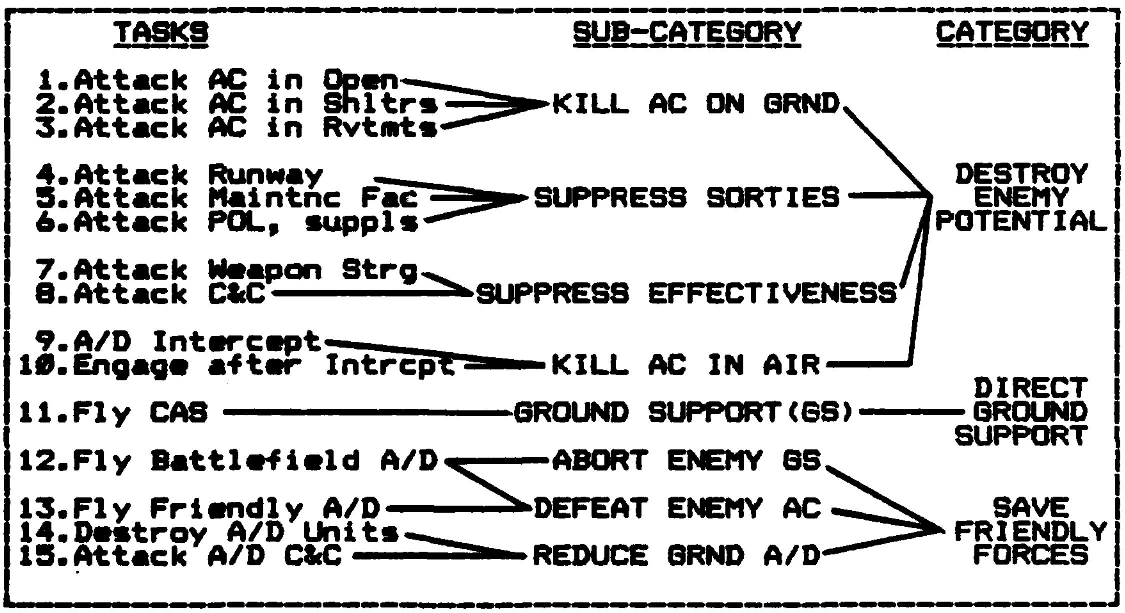


Figure 2 Details of Air Operations. (Ref.5)

4) combat air support--the direct, numerical, tonnage contribution.

The concept of TAWM is to normalize air operations by relating the contribution of each category of air operation to the payoff of the game: the difference between friendly and enemy tonnage. With this design, changes to model inputs, such as improved weapon accuracy or higher reliability, can cause changes in values. It can cause a change in the relationships between effort and damage. Analysts may then measure the relative merit of one target over another.

Figure 1 depicts the overall model. Figure 2 details the specific tasks associated with the three categories of the 1984 proposal. Figure 2 also highlights a minor difference between the 1967 and 1984 models: three categories of operations, rather than four. These new categories are as follows:

1) destroy enemy potential;
2) save friendly potential; or,
3) participate in combat air support.

One other theater-level warfare model built by AF/SA in 1974-1975, is TAC WARRIOR. TAC WARRIOR bears close resemblance to TAWM, in both concept and design. To determine air-to-ground effectiveness, TAC WARRIOR uses a sub-model, BLUE MAX. And BLUE MAX uses *Joint Munitions Effectiveness Manual* techniques to compute effectiveness of weapons delivery. However, TAC WARRIOR may be *too* big. It may be too complex to perform the level of analysis required to correct the problem of Chapter I. Problem solution does not require the extensive capability of theater-level warfare models, and in fact, solution of the problem in Chapter I can *contribute* data to these more extensive models.

Targeting Works

The *Joint Munitions Effectiveness Manual* (JMEM) is a classified collection of target and weapons data. JMEM provides a targeting methodology. Written and revised several times since 1975, JMEM does not *optimize* aim-points. Rather, the JMEM method is mechanical. The planner enters charts and graphs with categorical parameters of target characteristics, delivery parameters, and desired damage and confidence levels, and determines the number of sorties required to achieve a desired level of damage.

JMEM is convenient for weaponeering a point-target, like trucks or buildings. It can quickly solve a task such as: destroy a SAM (surface-to-air missile) site with 75% probability of success.

Agencies have recently begun funding purchase of software based on the JMEM methods. Notably, magnetic cards with stored JMEM routines are available for both the TI-59 and the HP-37 handheld calculators. Also, several versions of JMEM programs exist for both WANG and Hewlitt-Packard microcomputers. Such software enhances JMEM utility. However, JMEM's overall performance becomes marginal when targeting an area target, like a runway.

Simple, probabilistic equations analyze weapons effects well for point targets like the SAM site. But JMEM gets more complicated for runways. Runways are usually built larger than combat minimums require. Although weapons might tear up 4,000' of a 9,000' runway, if aircraft can operate on the remaining 5,000', it is difficult to evaluate the success of the mission. Therefore the mathematics behind the charts and graphs take an order statistics approach to determine a probability of cut. Using approximations, the method calculates the probability that the largest clear width, CW, within a line of craters across the runway, is less than the minimum width required for TOL operations, WR.

In the simple case, assuming a uniform distribution across the runway, and normalizing CW for a runway width of 1, this probability is given by:

```
Pr{CW>WR} = n*(1-WR)**(n-1)-(n/2)(1-2*WR)**(n-1)+...
                 ((-1)**(i+1))*(n/i)*(1-i*WR)**(n-1)
where n = number of spaces = number of weapons + 1
```

The series continues until (1-i*WR) <= Ø. (Ref.4:81)

The order statistics approach gets more complex when dealing with normal distributions. Furthermore, in addition to the normal error distributions associated with the attack, there also exists a chance of weapons dudding on impact. Clearly, computerizing such complex relationships is beyond the scope of this research.

Furthermore, sensitivity analysis when using JMEM's is not possible. For example, if the JMEM output indicates 24 sorties to close a runway, what is the expected damage if only 6 sorties are flown?

Since its release, JMEM has set targeting standards. But JMEM can be improved concerning runways. This thesis will contribute one part of that improvement.

Other targeting works include two, unpublished, AFIT M.S. theses. One is by John C. Pemberton, and the other by Howard M. Hachida.

Pemberton's work optimally assigns aimpoints for perpendicular runway cuts. He used set theory to find an "open" cell, through a method called discrete approximation. The event of interest is the event that the runway is cut (the minimum clear width is denied).

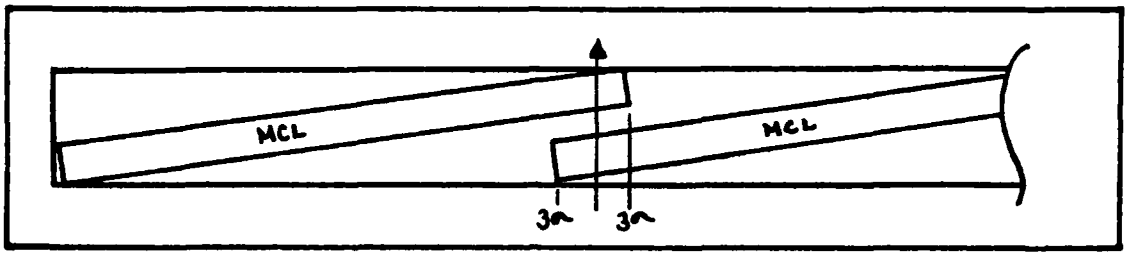


Figure 3 Independence of Runway Cuts (Ref.7:20)

A reasonable assumption of this approach is that minimum clear lengths are long compared to the standard deviation (S/D) of the errors. Then, if cuts are aimed at least three S/D inside the end of a minimum clear length, each cut can be considered an independent event. Figure 3 illustrates the concept.

With a discrete approximation, the runway is approximated with a number of discrete, overlapping, minimum launch widths. Since the widths overlap, the closing of each discrete section is not independent of closing other sections. Therefore, probability of cut is obtained from the complex set theory of combining these events.

Pemberton intended his work to be used during wartime operations, so one of his constraints was fast execution. He limited his analysis to singly released, high precision weapons.

Hachida's work improved the discrete approximation used by Pemberton. He found redundancy in the analysis of certain, individual sections. He eliminated the redundant sections, and reduced the time required to run Pemberton's program. He also improved the search algorithm determining optimum aimpoints.

Both works are excellent research, and contributed to the understanding of weapons effects and the optimal targeting of single-warhead, singly-released weapons.

Neither work, however, considered multiple weapon releases.

The current thesis is designed to enlighten decision makers on alternate targeting concepts. It will provide data that should be studied before conflict. Therefore, it is capable of examining multiple releases. Also, it has no requirement for perpendicular cuts. And when considering several weapons released on a single pass (a *stick* delivery), a perpendicular pass can be harmful. For example, a typical delivery airspeed of 540 knots, and the minimum intervalometer (or time-between-releases) setting of 0.05 seconds, results in an impact spacing of just under 50'. At best, a perpendicular pass on a runway, 200' wide, could produce only four impacts. Therefore, depending on aircraft weapon load, and expected accuracy, some angle-off to the runway centerline will maximize the number of impacts per pass. The current thesis will analyze targeting not only single weapons, but also strings of weapons.

Computer Simulation Models

In addition to the theater-level warfare models discussed earlier, other smaller scale simulations of air-to-ground weapons delivery exist. These include: AIDA, AHAB, RUNW, and AAP--all designed specifically for airbase attack.

AIDA is a large-scale, damage assessment model used by Air Force Studies and Analysis. It simulates many of the elements of airbase attack, including enroute attrition of the attackers. Runway damage is assessed by sliding a rectangle of required clear dimensions along the runway, and looking for a clear area. Although otherwise comprehensive, when assessing runway damage, AIDA only considers point-impact weapons. (Ref.7:11) The analysis is thorough, but program size makes execution difficult, and limited to large capacity machines.

AHAB is an interactive RAND model that uses decision maker (DM) value functions to maximize attack results. However, the DM does not have full authority in the design of the attack. AHAB assumes evenly spaced, perpendicular cuts, and allows only one weapon type in the attack.

RUNW is a simple, calculator method for determining the probability of closing a single runway. It was developed by SHAPE Headquarters in the early '7Ø's. Though effective for small attacks with point-impact weapons, RUNW cannot handle the variance of weapons that can be delivered by tactical aviation, nor will it allow flexibiltiy in designing attacks.

Finally, AAP is another large-scale, Monte Carlo-type attack assessment program. It has slightly less target capacity than AIDA, but AAP allows more flexibility in the design of attacks. Specifically, AAP will evaluate cluster munition effects against runways, as well as assessing the effectiveness of point-impact weapons. But again, because of AAP's large size, it is difficult to use and does not permit interactive execution.

Given the shortfalls of each of these models or methods, it was originally decided to develop a new model. Consideration was given, and attempts made, to use either QGERT or SLAM simulation languages. However, the intricacies of the clear strip and taxi searches forced the effort to study the detail of one of the above models. The choice, based on flexibility of the allowable attacks, was to use the search algorithm of AAP.

An attempt to transport the search algorithm to the QGERT or SLAM driver programs failed, due to the complexity of the routines. Therefore, it was finally decided to modify AAP to satisfy the needs of the problem. The modification would make the program a useable tool for tacticians and operations planners. Chapter IV documents the conversion of AAP into AAPMOD. But the rest of Chapter II presents further details of AAP.

Attack Assessment Program (AAP) was developed by the University of Oklahoma, under contract F-Ø8635-79-C-Ø255, for the Joint Technical Coordinating Group for Munitions Effectiveness. AAP has excellent program design. AAP will evaluate the effects of multiple warheads delivered against a target complex composed of multiple elements of three types:

1) Take-off and landing (TOL) surfaces: pavements or sod areas capable of supporting TOL operations;

2) Minor taxi-ways: pavements or sod capable of supporting only taxi operations; and

3) Structures: buildings, bunkers, POL storage or delivery facilities, etc.

As indicated earlier, AAP has substantial input capacity. But the price is paid when loading for execution. For example, AAP will allow up to 1Ø separate attacks per mission, with up to 64 delivery passes per attack, with up to 16 different delivery patterns, with up to 36 weapons released per pass. However, even with a CDC CYBER NOS/BE operating system, AAP was too big to run interactively.

During execution, user defined attacks are assessed for the damage they cause to a user defined target complex. Locations and orientations within the complex are referenced to a right-handed, two-dimensional, Cartesian coordinate system. All angles, for both target element orientation, and attack definition, are measured in degrees, CCW from the positive X-axis.

The allowable limits for target definition are as follows:

> 207 target elements,
> of which up to 43 may be pavements,
> of which 3 may be TOL pavements.

As overhead to these limits, AAP further allows up to 11 types of surfaces, each with a different hardness code, called the surface code. Together with 6 different types of warhead codes, the various combinations of the two codes define the size of craters.

Finally, implementation of AAP is straight-forward:

1) Each Monte Carlo iteration represents a mission.

2) Within an iteration, the program first "flies" out the mission. AAP loops first on attack number, then pass number, assigning an impact location to each warhead or submunition. If proper fuzing occurred, the resultant crater is evaluated in its proximity to target elements. Both hits and near-misses are stored for later damage assessment.

3) When the mission is complete, AAP assesses the hits for target damage. Search routines determine TOL status, taxi-way status, or structural damage.

4) Finally, AAP accumulates the damage of each Monte Carlo iteration and yields output statistics of the expected damage of the overall mission.

Each of the works addressed in this chapter, in some way enhances understanding attack efforts and damage results. And, given the expected damage of a defined target, a commander can decide whether his efforts, and possible losses, are worth the expected damage.

The current research has drawn from these works to develop a methodology enabling a clear understanding of damage versus effort. Limiting the scope of the associated experiment to one type of aircraft, against one type of target, this thesis remains a reasonable, yet functional study.

This study should stand on its own, to assist tacticians and aircrews to optimally plan weapons deliveries. Additionally, it fills the *practical* void in current runway targeting analyses, and helps AF planners avoid the difficulties encountered in the USAFE exercise. Finally, this thesis can yield the return-value of attacks against targets. It can help clarify the relationship between level of attack and expected damage. And in proper format, the data produced by this research can become an input to larger scale models.

III. System Specification

Background

In recent years, both allies and enemies have hardened their airbases. "Hardening" means to reduce vulnerability to attack. A case in point is RAF Upper Heyford, in Oxfordshire, England. Recent construction includes over 60 hardened aircraft shelters (HAS), as well as several operation centers and maintenance facilities. The shelters, for example, are constructed of reinforced concrete, over 36" thick at the base, and over 18" thick at the top. This design is depicted in Figure 4, below. For clarity, sliding doors, weighing over 50-tons each, are omitted. When buttoned-up, these HAS can withstand most conventional attacks, as well as some small-yield, nuclear near-misses. These shelters eliminate the once lucrative target: aircraft in the open.

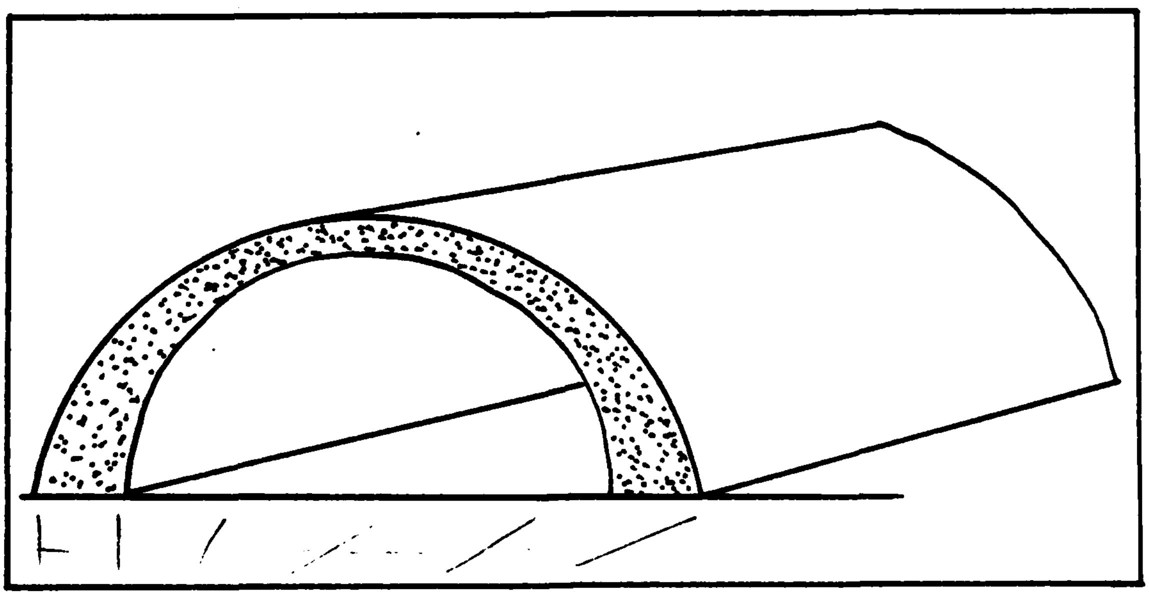


Figure 4 Typical Hardened Aircraft Shelter (HAS).

But just as hardening improves NATO survivability, similar efforts have been matched by the Soviets. They have hardened their main operating bases, though to a lower proportion. Comments by General Wilbur L. Creech, the Commander of Tactical Air Command (TAC), as reported in *Armed Forces JOURNAL (AFJ)*, January 83, indicate Warsaw Pact HAS capacity does not exceed a shelter to aircraft ratio of 1:3. (Ref.14:28) 1/ Regardless, their hardening efforts have reduced the vulnerability of their aircraft to attacks by our tactical aviation.

The Israeli Air Force (IAF) can take credit for the resurgence of modern hardening efforts. The concept of cover to protect resources is not new. But as with most projects that require funds, hardening efforts received low priority. Then, on 5 June 1967, the Israelis plainly demonstrated the utility of sheltering aircraft in HAS. On that day, the IAF attacked 26 Arabian airbases. In one day, the IAF destroyed over 350 aircraft on the ground. The IAF swiftly established air superiority, after which the Arabs could only muster harassment attacks. In total, the Arabs lost about 450 aircraft in the Six-Day War. Of those losses, 393 aircraft were killed on the ground. Meanwhile, the Israelis only lost about 40. (Ref.20:80)

But the Arabs and their supporters took the lesson. With their rearmament between '67 and '73, the Arabs built hangerettes as they reacquired equipment. And in October of 1973, the IAF's counter-air efforts were less successful. The IAF destroyed only 22 aircraft on the ground: hangerettes worked. In '73, IAF counter air had to attack runways and taxiways to suppress Arab air. And as will become apparent later in this chapter, denial of these

---

1/ The Warsaw Pact currently has over 7,240 combat aircraft in-place, in Europe. (Ref.15:17) surfaces required frequent, heavy attack. Coupled with sophisticated missile defenses, the IAF lost 109 aircraft. Of these losses, 73 occurred in the early part of the fight, with as many as 24 in one day. (Ref.20:80)

The obvious question is: Why attack airfields? The answer lies in a complex analysis of modern warfare, perhaps conducted with the aid of a model such as TAWM, discussed at length in Chapter II. Recall that one framework for a model of theater level warfare can be based on *game theory*. The *players* are the two sides: red forces and blue forces. The value of the game is net tonnage, delivered on the enemy, in support of the ground battle. Each of the actions specified in Figure 2 contribute value, or in some way, effect a positive change to the net tonnage figure.

Although the opportunity, as General Creech reminds us, to destroy enemy aircraft on the ground is not totally eliminated, this discussion of airbase hardening should infer that trying to destroy the enemy's potential, by destroying his aircraft on the ground, is becoming an increasingly more difficult task. Destroying runways, to prevent TOL, is therefore one alternative.

In-depth consideration of other attack options is beyond the scope of this study. Factors affecting the decision include the following:

1) Availability and traffic capacity of alternative TOL surfaces, such as taxiways and grass strips. And,

2) The *value* of alternative targets such as POL or maintenance facilities in denying sortie potential.

But destroying runways is the primary option studied in this research.

A fundamental purpose of this study is to develop an understanding of the relevant factors affecting the probability of cutting a runway. To optimally allocate their fighters, decision-makers must know the effectiveness of the particular aircraft against various types of targets. To ensure a common level of understanding, the "system" of the attack is detailed below. The response of the system is the probability of denying enemy TOL operations from the runway.

The System

For purposes of this study, the process of runway attack begins with the aircraft 20 nm from the runway. The crew has survived enemy defenses to this initial point (IP) for their attack. The navigation systems are updated as well as possible, and the aircraft makes its target run. The crew encounter terminal defenses. The crew, aided by the aircraft systems, must visually acquire the runway, and release the weapons at an appropriate point to impact the runway. The damage mechanism is a crater, surrounded by a disrupted, cracked ring of pavement, over which an aircraft cannot operate. The term "crater radius", implies both the crater and the unuseable ring around it. If the impact pattern occurs so that no clear rectangle of the minimum required dimensions exists on the runway, the runway is closed (unuseable).

Typically, the crew plans an attack by first studying the attack request. If the attack must deny use of a runway for some length of time, they will choose an axis-of-attack for cuts, based on enemy threats, navigation pointers, and damage requirements. Note that maximizing damage is not the only factor affecting the choice. If threats or the potential for poor navigation accuracy deny the optimal angle-off, the crew must settle for a sub-optimal attack plan. (This thesis can provide an analysis of the expected damage for any angle chosen.)

With their plan the crew tries to maximize:

1) their chances of surviving;
2) their chances of finding the target; and
3) their chances of damaging the target.

Generally, someone else chooses the weapons the crew will deliver; however, the crew can request a change if they do not agree with the choice. On the other hand, the type of weapon pattern employed is totally at the crew's discretion.

The weapon pattern is the result of complex interactions of many variables. Some are controlled variables, defining a type of pattern, while others are stochastic variables, affecting the actual locations of craters within the pattern. These variables are individually addressed later. But first, the reader is reminded of the four types of elements, or *variables* used in simulation models:

1) Stochastic variables: variables over which the user has no control.

2) Controlled variables: variables that the crew or planners can control:

3) Modified control variables: planners or crew have control over the parameters of the parent distribution, but once the process begins, values are randomly drawn from the distribution.

4) Parameters of the system: these are variables that once set, remain constant. (Ref.19:15)

The above types of variables comprise the system inputs. The system processes inputs, and produces an output: a response. In fact, the complex system yields numerous responses. But of primary concern in this study is the response of the probability of closing a runway. Figure 5, that follows, graphically relates inputs to response with a causal diagram of the interactions of the input variables. The next few pages discuss these inputs in detail, followed by discussion of the response variables.

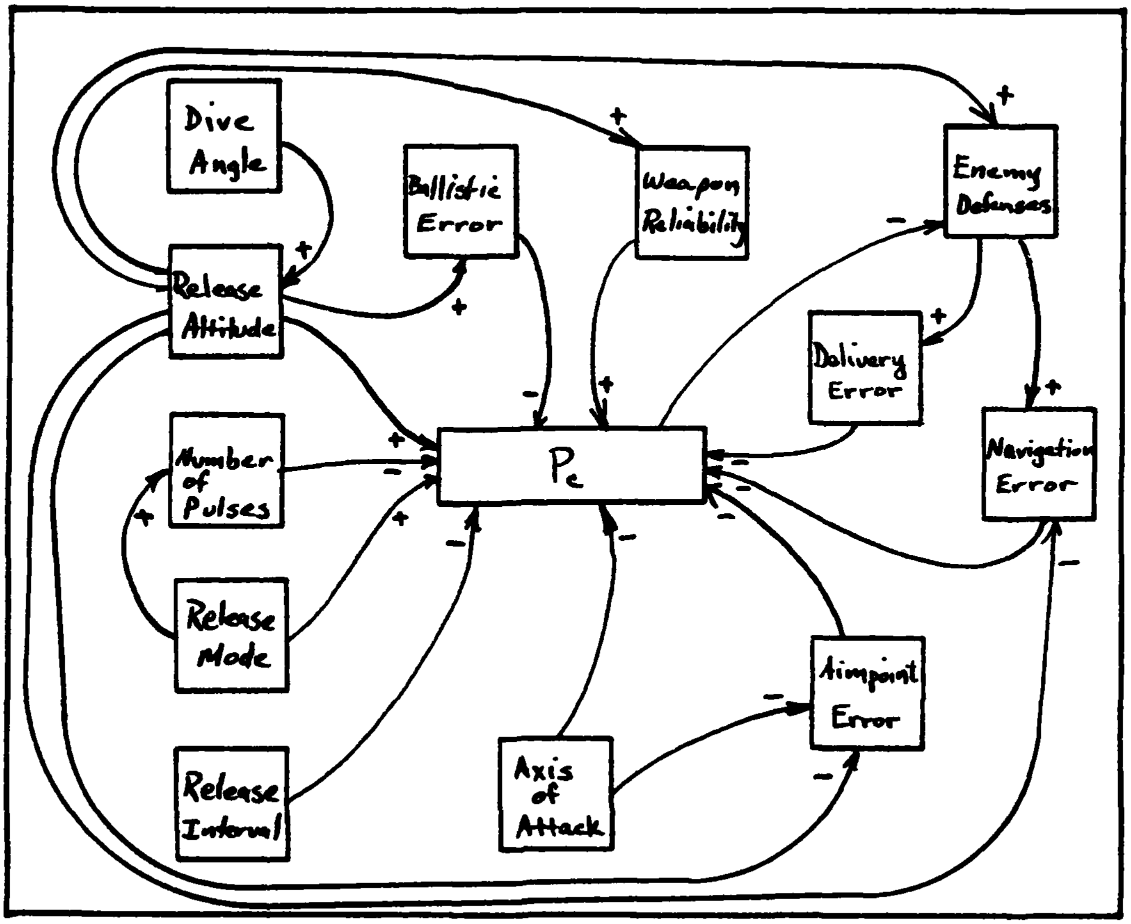


Figure 5 Runway Attack Causal Loop Diagram.

In the following list of variables, note the assortment of variable types. Variables range from continuous, ratio-type quantities, such as error distributions, to qualitative, categorical variables like release mode.

Navigation Error. Navigation error is a stochastic variable, based on crew abilities and aircraft systems. The crew may or may not find the runway.

Aimpoint Error. Given that the crew finds the target, they may misjudge the pre-planned aimpoint. This type of error is called aimpoint error. Aimpoint error is minimal when considering point targets like radar sites or isolated buildings. However, it can become large when considering area targets, such as large tank farms or runways. Under combat conditions, there can be a strong tendency for the crew to misjudge the one-third or one-quarter point of a nine or ten thousand foot runway. Aimpoint error is a stochastic variable that can depend on axis-of-attack. The error will always be greatest along the longitudinal axis of the runway. Data for this type of error is not currently available. However, discussion with several classmates and instructors, with a combined experience of over 35 years in ground attack fighters, suggests use of a triangular distribution.

Delivery Error. Delivery error is a controlled variable that describes the error attributed to a combination of the inaccuracies in:

1) crew release procedures, and flight parameters, at time of release; and
2) the aircraft release system.

Delivery error is considered a controlled variable, because the parameters of the distribution representing the error can be controlled. Crew proficiency, developed through training, will affect the crew's accuracy. Similarly, the accuracy of the aircraft armament system depends on the quality and availability of its maintenance.

If the crew properly identifies their aimpoint, delivery error will still displace the weapon pattern from the aimpoint. Historical data supports use of a single, normal distribution with a mean of zero to provide one term to incorporate both errors. However, delivery error has two components:

1) Range error--or error in the flight direction of the aircraft; and

2) Deflection error--or error transverse to the flight direction.

These two components define a bi-variate normal error distribution. And if the two components are identically distributed, i.e., their distributions possess the same standard deviation, they describe a circular normal error distribution. The reader is referred to either Pemberton or Hachida for further detail of these error terms.

Ballistic Dispersion (Bd). Each weapon will have its own random error due to slight differences in center of gravity, weight, release orientation, wobble, etc. This error is usually described in radians, so actual ground-distance depends on the range of the free-flight trajectory of the bomb after aircraft separation. This study considers Bd a stochastic variable. Refer to Appendix B for further detail.

Weapon Reliability. Due to the high speed of impact, and the hardness of the concrete, the bomb could ricochet or break, instead of explode, and no crater forms. By selecting the weapon and the delivery parameters, the crew can control reliability. Therefore, weapon reliability is a modified controlled variable.

Release Interval. The time interval between release pulses of the armament system, typically measured in milliseconds, is the release interval. This variable is a controlled variable, above some system-dependent minimum interval.

Release Mode. Weapons may be released one or more per pass. If an even number of weapons are to be released,

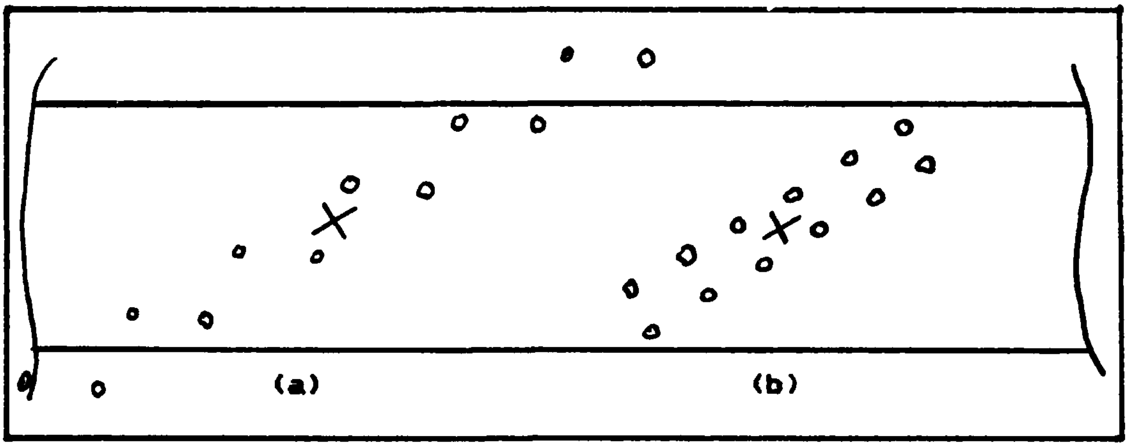


Figure 6 Weapon Impact Locations for a Stick of Weapons. Pattern Resulting from (a) Release--SINGLES, (b) Release--PAIRS.

the crew then chooses to release the weapons singly, or in pairs. The aircraft armament system will either release weapons simultaneously from both sides of the aircraft, or will alternately step releases from side to side.

The resulting patterns are illustrated in Figure 6. Release Mode--SINGLES results in a long pattern, while release Mode--PAIRS results in a shorter, more dense impact pattern. By its nature, release mode is a controlled variable.

Number of Pulses. The armament system can be set to send any number of release pulses to the bombracks. The number of pulses determines the number of weapons released per pass. Based on mode selection, one or two bombs will drop with each pulse. If more than one pulse is selected, the string of releases is called a "stick of bombs". The number of pulses is another controlled variable.

Release Altitude. Release altitude is a controlled variable. It represents the height of the aircraft, above the ground, at the release point for the weapons pass. Due to free-flight of the weapons as they drop, this point is usually well short of the desired mean point of impact (DMPI). An error in achieving this variable, during the release can cause significant miss distances. However, during a systems analysis, miss distance due to altitude, dive, or airspeed errors, is lumped together in a part of the delivery error term, defined earlier.

Release Speed. Another controlled variable, release speed is the true airspeed of the aircraft at weapons release. When interacting with the release interval, release mode, and dive angle, release speed sets the ground spacing between impacts. To inject realism, one may safely assume the crew will choose the fastest release speed weapons will permit.

Dive Angle. The dive angle is the angle the flight path of the aircraft makes with the ground at weapons release. Dive angle also affects other variables of the system. For example, a diving delivery implies higher altitude, resulting in better accuracy, and weapons reliability, but possibly more exposure to threats, and so less survivability. Dive angle is a controlled variable.

Weapon Pattern. The weapon pattern is the result of the interaction of release mode, release interval, release speed, altitude, and dive-angle. One can consider two types of weapon pattern: intended and actual. The intended will be a symmetric, neat pattern, centered on the aimpoint. The actual weapons pattern perturbs the center of the pattern from the aimpoint because of aimpoint error and delivery error, and Bd perturbs the individual impacts within the pattern. Figure 7, on the next page, illustrates these concepts.

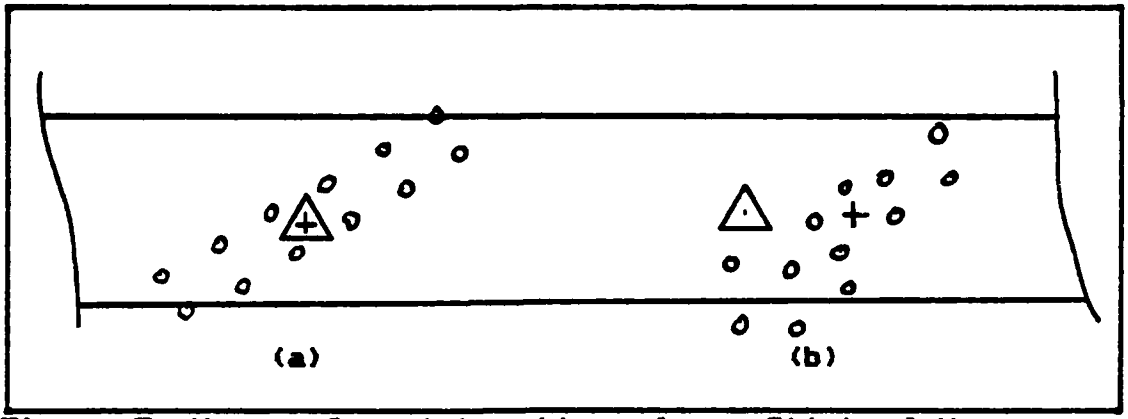


Figure 7 Weapon Impact Locations for a Stick of Weapons. Pattern Depicted is (a) Intended Pattern, (b) Actual Pattern.

<u>Aimpoint</u>. An important consideration of the attack is the desired mean point of impact (DMPI) for a stick of bombs, or the desired point of impact (DPI) for a single release. This point is chosen by the crew, and is thus a controlled variable.

<u>Axis-of-Attack</u>. Axis-of-attack is the angle the flight path of the aircraft makes, referenced to the longitudinal axis of the runway. A controlled variable, driven by considerations as follows:

1) Navigation Aids--the crew will choose an IP that will maximize their chances of finding the runway. So to preclude gross maneuvers departing the IP, axis-of-attack is somewhat limited.

2) Target Defenses--the crew may be denied optimum axis-of-attack if on the run-in line, three miles short of the runway, the enemy has established a gun emplacement.

<u>Crater Radius</u>. Crater radius is the size of the hole produced by the exploding warhead. Crater radius is a function of the type of weapon, depth of penetration of the warhead before exploding, and type of surface. AAPMOD considers crater radius a parameter of the system. By virtue of the physical interactions of warhead and target,

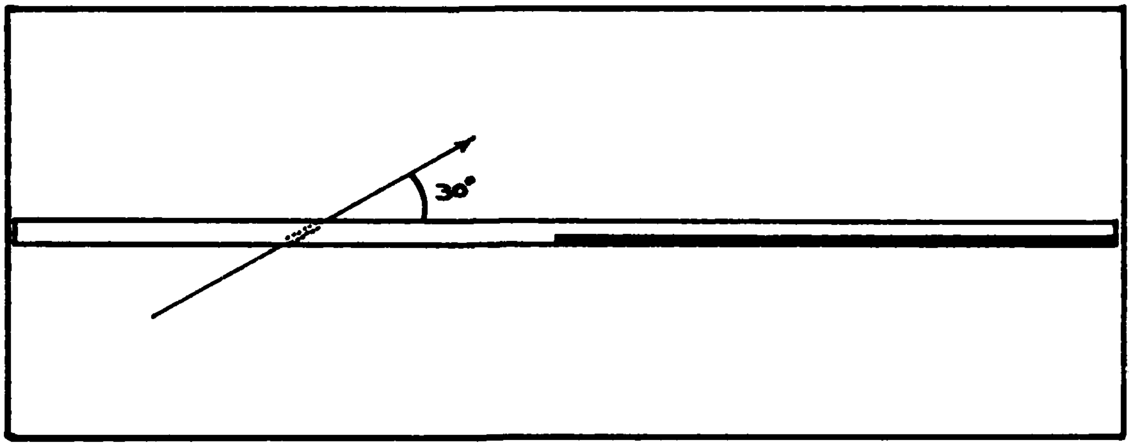


Figure 8 Accurately Scaled Runway with Craters.

crater radius can be considered a parameter. However, since the set of interaction conditions are chosen by the user, this study will consider crater radius as a controlled variable.

Runway Dimensions. A parameter of the system is the original size of the runway to be attacked. To ensure the proper perspective of this system, Figure 8 is an accurately scaled drawing of an 8,000' x 150' runway, with 12 craters, from weapons released in pairs, at 480 kts, and 50 ms spacing. The shaded area represents a minimum clear area for TOL, chosen for this example to be 4,000' x 50'.

Minimum Clear Dimensions. Another parameter, for any one system, the minimum clear dimensions are those clear dimensions required to permit aircraft take-off and land (TOL) operations. These dimensions are a function of the aircraft operating from the runway.

Survivability. Sortie profile, routing, and tactics all affect the overall chances of the aircraft making it to the weapons release point. The intricacies of this variable are complex, beyond the scope of this research. Survivability is a function of weather, equipment status, operator proficiency, degree of saturation, plus many more factors. Therefore, the judgement of the individual user will determine the value for aircraft survivability. This element has been retained because it is felt newer aircraft have greater survivabiltiy in combat operations, and this fact must be considered by the model when considering competitive effectiveness. Given this discussion, the probability of the aircraft surviving to the release point is a modified controlled variable.

System Response

The system response is damage. But damage can be a nebulous term. Damage is deleterious change to the system. The intended damage of an airfield attack is denial of the use of the base. Recall from earlier in this chapter, that there are several ways to achieve the response. The most obvious, and the response of interest in this study, is to deny the physical, clear area.of pavement required to support TOL.

Damage itself is hard to measure, so measurement of the response requires surrogate measures. Area cratered, or number of hits are some ratio-type measures. Airfield status or runway status, open or closed, are other, categorical measures. The idea of two categories gives rise to Bernoulli trials, and ultimately a probability of the attack closing the runway, or the airfield. And airfield status is the response of primary concern in this research.

Understanding the damage response is crucial to understanding the *system*. Four important events are associated with the response. These events are a runway

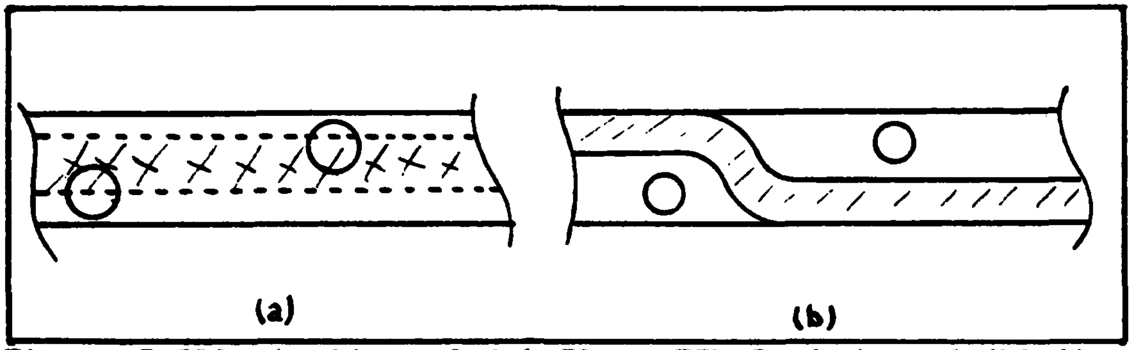


Figure 9 Illustration of (a) Clear TOL Denied, and (b) the Available, Meandering Path for Taxiing.

cut, a taxiway cut, runway closure, and field closure. Each of these events is defined, below.

A runway cut is a chain of craters across the runway. The width between the craters must be less than the minimum width required by the using aircraft. Because an aircraft cannot maneuver around craters during hi-speed TOL, offsets of the craters, along the length of the runway, do not abrogate the denial of TOL capability, *if the minimum width is denied*. Figure 9(a) depicts two craters cutting a runway.

A taxiway cut is slightly different. Again, a chain of craters must exist. But now, lateral displacements between the craters must also be minimized. Due to slower speeds, and the possibility of ground marshallers, taxiing aircraft can meander their way around craters. Also, the effective size of the disruption changes. Because of the slower speed, and better accuracy of tire placement, the radius of disruption, severe enough to deny taxiing, is less than the radius used to deny TOL.

Figure 9 depicts two craters. As mentioned above, the craters in (a) deny the minimum clear area required for TOL, so the runway is cut. But in (b), the same crater locations do not deny taxi. Not only are the disruptions smaller, an aircraft can meander around the craters, so the surface is not cut. NOTE: The same surface and impact pattern can have different status depending on the type of activity required of the surface.

A TOL strip can be so large that it requires two or more cuts to deny the minimum clear dimensions. And, if as addressed earlier, the aimpoints for the cuts occur three standard deviations from the ends of clear areas, the cuts can be considered independent. Then the probability of closing a large runway is the product of the probabilities of the single cuts. And by Pemberton, these probabilities are taken to be identical. (Ref.16:5) For this case, the adjective, *physical*, applies. Craters physically prevent a clear operating area, and the runway is closed.

The last case borders on the limits of this thesis. The obvious way to deny operations from a field is to close each runway. But another way is to deny taxi to the runway or the clear area remaining open for TOL operations. A gross simplification would be to assume independence of all these events. Perhaps, in an analysis limited to highly accurate, point-impact weapons, the approximation would be good. But experience suggests that the size of sticks of weapons, interacting with low delivery accuracies, and small target element separations produce collateral damage responses. And the events of interest are no longer independent.

Although only the event, closing all runways, is studied here, the concept of denial can be extended to denying *access* to the runways (or the clear areas of runways) that remain after an attack. So although the minimum clear required dimensions may physically exist, without access, they cannot support TOL.

But the system is not limited to these probabilistic responses. Other responses include the total number of craters, the locations of craters, or the minimum number of craters requiring repair to regain open status. Another response is number of aircraft lost in the attack, or number of weapons dudding. Each of these responses may have significance. An ideal model of the system will accept each of input variables that were discussed, as well as output all of the responses.

The model resulting from this thesis effort is not ideal. However, AAPMOD, a modified version of AAP, does input and use 11 input variables, and allows up to 6 definitions of weapon pattern. Also, each of the above responses is an output of AAPMOD.

In summary, Chapter III has defined and detailed the *system* of a runway attack. Chapter IV will now describe the implementation of these concepts in the AAP derivative model, AAPMOD.

IV. <u>Implementation</u>

The preceding chapters have demonstrated the need to better understand the relationship of attack tactics and target damage. They have illustrated the interactions of some of the variables comprising the attack-target system. A discussion of Attack Assessment Program--MODIFIED (AAPMOD) will now provide tacticians the methodology to achieve the understanding suggested in Chapter I.

Chapter IV describes AAPMOD, which was developed from the Attack Assessment Program (AAP), discussed in Chapter II. The three sections of Chapter IV begin with a brief discussion of computer simulations. The discussion of simulations is followed with a discussion of the conversion of AAP to AAPMOD. To facilitate data input, conversion included the development of AAPIN. AAPIN enables interactive implementation of AAPMOD. The chapter ends with discussion of program execution of both AAPIN and AAPMOD.

<u>Computer Simulation</u>

Models are *descriptions* of systems. AAPMOD is a model. Specifically, AAPMOD is a computer simulation of the complex interactions that occur when tactical aviation delivers ordnance against the enemy. The variables discussed in the previous chapter characterize the *state of the system*. Based on user inputs, AAPMOD moves the system from one state to the next with discrete events. These events include aircraft survival, weapons release, weapons function, and attack termination. The states of primary concern are pre-attack target status, and post-attack target status. This section of Chapter IV addresses the cogent concepts of AAPMOD.

A Monte Carlo Simulation Two of the ways available to examine stochastic systems, or systems that contain probabilistic elements, are: 1) expected value analysis and 2) Monte Carlo sampling techniques. Each method has advantages and disadvantages. Some mathematicians require the rigorous proof of a probability analysis, and claim Monte Carlo sampling should only be used as a last resort. (Ref.2) But others defer to the success demonstrated by the technique since the late 1940's. These supporters point out that Monte Carlo techniques can be used to solve completely deterministic problems, that cannot be solved analytically. (Ref.19:65)

Briefly, Monte Carlo sampling generates random, artificial data to simulate experience. The process first establishes a random value for each of the probabilistic elements of a system. Once a value has been assigned to each element, the system is analyzed for its overall response. The response is stored, and the sampling continues, defining new component values, and producing new responses. After an appropriate number of iterations (*appropriate* will be defined later), an average or "expected" response becomes the output of the process.

The accuracy and fidelity of the simulation depend, in part, on the choice of distributions and parameters describing the probabilistic elements. Next will follow a discussion of the distributions, and their parameters, used in AAPMOD.

Probability Distributions and Parameters The only probabilistic variables in AAPMOD are weapon impact error, weapon reliabilities, and aircraft survival. The distributions assigned to these variables have been validated with years of data collection, and by either combat experiences or intelligence projections.

Weapon errors consist of two types. The first is aimpoint error, and the second is ballistic dispersion error. Data has been collected from operational test and evaluation of weapons, aircraft, and tactics, and from both combat and training weapon delivery records, and supports the choice of normal error distributions. During weapons delivery, the parameters discussed in Chapter III affect the mean point of impact (MPI) of the weapon or weapons. And although *mean* point of impact may not be entirely accurate when describing the release of a single weapon, this report will generically use MPI to represent the actual impact point of either a singly released weapon, or the center of impacts for a multiple release. By definition, MPI implies that random, normally distributed error displaces the center of weapon impacts—the MPI, from the aimpoint (which is also called the *desired* mean point of impact—DMPI).

The other type of weapon error is ballistic dispersion (Bd). This error was discussed in detail in Chapter III. Recall that each of six weapons may have a slightly different center of gravity, or receive a different ejection velocity from the bomb-rack. The resulting impact pattern depends not only on the aimpoint error of the stick, but also on the individual errors induced by wobble as the weapon falls, or random velocities as the weapon begins its trajectory.

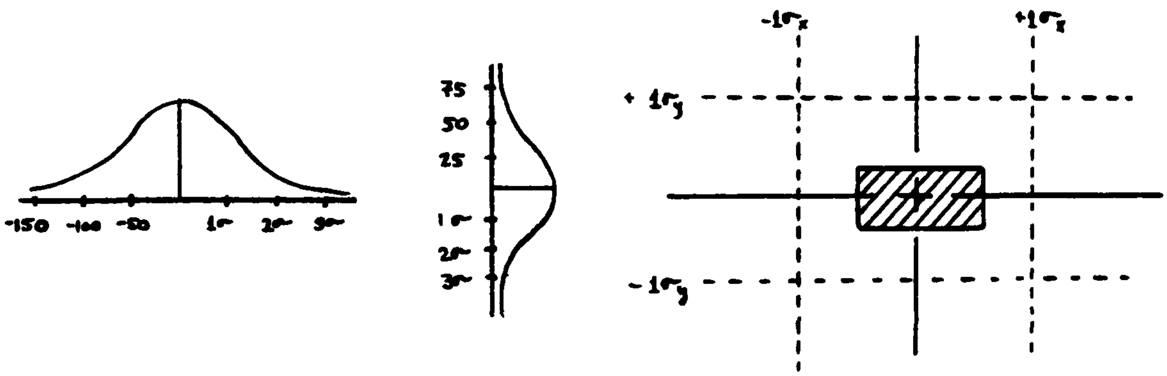
Given the distributions and parameters for the aimpoint errors and ballistic dispersion, one can determine the expected number of weapons impacting the target. The process is simple, as demonstrated in the following example:

Example 4.1: The ballistic error of a new gun, at a given range, is independently, normally distributed, in both the X and Y direction. The standard deviations are 50' in the X direction, and 25' in the Y direction. The gun is aimed at a target that measures 25' in the X direction and 10' in the Y. If the bullet hits the target, it will destroy the target.

To keep it simple, assume the gun is perfectly aimed.

What is the probability of target destruction?

Sketches illustrate the concepts:



The probability of the projectile hitting, and thereby killing the target is simply the product of the probability of X-error being less than +/-12.5', and Y-error being less than +/-5'. From *CRC* normal tables, these probabilities are as follows:

$$Pr\ (-12.5 < X < +12.5) = 0.1974$$
$$Pr\ (-5 < Y < +5) = 0.1586$$
$$Pr\ (Hit) = Pr\ (Hit|X) * Pr\ (Hit|Y) = 0.0313$$

In reality, Example 4.1 is grossly simplified to illustrate the basic probability theory of weapon effectiveness. The bullet would have real area, and particular components of the target would be more or less vulnerable to the impact.

The other probabilistic elements considered by AAPMOD are aircraft survival, and weapon reliability. The aircraft must survive enroute attrition to release its weapons. If the aircraft reattacks, it must also survive target area defenses. AAPMOD uses simple, discrete probabilities when testing for these events. The user enters the probability of surviving to the release point. Prior to weapons release, AAPMOD draws a random number and compares it to the aircraft's chance of survival. If the random number is less than the input probability, the aircraft survives. If the random number is greater than the probability of survival, the aircraft is lost, and none of its weapons impact the target area. The same process controls reattacks, as well as proper weapon detonations, CBU dispenser openings, and CBU bomblet detonations.

If one assumes these events are independent, the ultimate probability of the desired response is the product of these individual probabilities. Suppose, for example, that in Example 4.1, there was only a 0.5 probability the gun would fire. Also, say that the enemy fielded a decoy target, so the chances of correctly aiming were only 50-50. The new probability of target kill would be:

$$0.0313 * Pr\ (Fire) * Pr\ (Correct\ Aimpoint) = \\ 0.0313 * 0.5 * 0.5 = 0.0078$$

Now, when the damage mechanism becomes cratering, and the target is a runway, the complicated probabilistic interactions strongly encourage the analyst to use Monte Carlo methods. An operational runway merely requires a minimum, undamaged width, for a minimum, undamaged length. Typically, runways are built longer and wider than the minimum size required for aircraft operations. To deny operations, these minimum rectangular dimensions must be denied. But they must be denied everywhere on the original strip. Denial occurs because the disruption of cracks, rubble, and craters prevents aircraft operation.

The attack generates a pattern of craters, that, four at a time, bound the possible clear operating area. Any one of the bounded areas may be large enough to support TOL operations.

Computing the probability of denying such a clear area depends on the interaction of many variables, both deterministic and stochastic. These variables were described in Chapter III. Two analytical methods include order statistics, as presented with the earlier commentary on the JMEM's method, and discrete approximations, used by Pemberton and Hachida. But to keep AAPMOD simple, the current analysis uses an alternative to pure statistical analysis: a numerical search. The results of the search are either success or failure, destroyed or not-destroyed, take-off denied or not-denied. These results are called *Bernoulli variables*, and are characterized with the binomial distribution. This is the type of analysis for which AAPMOD is optimized.

<u>Confidence in AAPMOD</u> AAPMOD is a typical, computer simulation. It uses random numbers, random variates, and replication to produce output. Of primary interest is the probability of an attack denying use of a runway (or runways). Described in Chapter III is the chain of probabilistic events that interact in complex fashion to produce weapons damage to a target. These interactions are modeled in AAPMOD. If AAPMOD is run enough times, the simulation results tend to be *more accurate*. And Bernoulli tells us, that as the number of replications, n, approaches infinity, the error, d, between the true denial probability of the population, and the sampling probability, approaches zero.

But what is enough? And earlier, what is appropriate? Since much of this study concerns the open or closed status of a runway, the problem becomes one of estimating a proportion: the number of closures per number of attempts. Referring to Shannon's [19] discussion of the binomial distribution:

Let p equal the true probability that in one trial, a given attack will close a runway. Let $q = 1 - p$ equal the true probability the attack will fail. And let P equal the sample probability of closure, obtained from Monte Carlo sampling.

Rewriting Bernoulli's theorem:

$$| P - p | \leq d \quad \text{as} \quad n \longrightarrow \infty, \text{ and } d \longrightarrow 0$$

Enough, or appropriate, is when the user can stand the probable error in the simulation results. If the user desires 90% confidence that the simulation probability of closure, Pc, does not differ from p, by more than 0.05, the problem can be written as:

$$Pr \{ | Pc - p | \leq 0.05 \} = 1 - \alpha = 0.90$$

If n is large (> 120), and if neither p or q are close to zero (< 0.05), the binomial distribution can be closely approximated with the normal distribution. Then using $Z_{\alpha/2}$, the two-tailed standardized normal statistic in the following formula, one can determine the minimum sample size required: (Ref.19:191-2)

$$n = \frac{Z_{\alpha/2}^2}{4d^2}$$

But a problem remains. These *accurate* results are accurate only so long as each of the event probabilities affecting the chain, is accurate. Since the probability of each event has some inaccuracy associated with it, there is inherent inaccuracy in the simulation results. This is not to say that the inaccuracy invalidates AAPMOD, but that the results must be used knowing that inaccuracies exist. AAPMOD offers a theory describing the interactions of airplanes, weapons, and targets. It is soundly conceived. The following sections will show that AAPMOD's output does bear meaningful relation to the real world interaction of attack efforts and expected target damage.

Program Conversion

Attack Assessment Program-Modified (AAPMOD) is a pseudo-interactive, Monte Carlo simulation of an attack against a target complex. The user inputs descriptions of the target complex and the attack, and the Fortran V program returns damage assessment.

AAPMOD is a modification to Attack Assessment Program (AAP), earlier described in Chapter II. AAP is currently used at the Armament Development Laboratory, Eglin AFB, Florida, as well as at 50-60 other Air Force and civilian contractor locations. The Armament Lab has been studying airbase suppression by conventional weapons. The Lab is primarily concerned with the sensitivity of damage results to changes in the following variables:

1) crater radius;
2) reliability of either:
   a) weapon/dispenser fuze reliability, or
   b) submunition fuze reliability;
3) ballistic dispersion of released weapons;
4) footprint of cluster weapons; and
5) number of cluster-weapon submunitions.

The above factors influence early, design-phase decisions. Such experimentation corresponds to the charter of the Armament Lab: to develop improved conventional weapons. However, until new weapons are delivered to the operational wings of TAC, PACAF, and USAFE, tactical aircrews must optimally employ current inventory weapons.

As discussed in Chapter III, damage results depend on numerous factors affecting combat weapons deliveries. AAPMOD provides tacticians and aircrews the opportunity to study the factors that are under their control, namely the following:

1) weapons load;
2) axis of attack;
3) probability of correct aimpoint identification;
4) definition of the stick pattern; and
5) delivery errors (REP/DEP, or CEP).

Each of these factors are controlled at a level of command no higher than a tactical fighter-wing commander. For preplanned targets, or after study of results of different analyses, both crews and commanders should be able to optimize these variables, and produce maximum damage with the weapons currently available.

AAPMOD is described as pseudo-interactive, because the bulk of interactive communication occurs in a front-end program called AAPIN. AAPIN generates a laundered file of user inputs to AAPMOD.

AAP was received from Eglin, and with comments, consisted of 2,310 lines of Fortran IV source code. Table I includes a listing of program statistics.

However, to be useful to aircrews, tacticians, or even commanders, AAP had to be made more "friendly." This implied interactive. Interactive processing could avoid the delays associated with batch mode, such as preparing job control cards, or fetching output from remote files or printers.

Consequently, a primary task in converting AAPMOD was to reduce its loading size. New input limits were imposed. These are presented later, in the section on inputs. Also, the coding of AAP was upgraded to include the facilities of Fortran 77. For example, the upgrade improved program

TABLE I

Comparison of Program Compilation Statistics 2/

AAP		Program Unit Length (words)	Blank Common (words)	Labelled Common (words)	Time (secs)
Program	MAIN	3,287	35,757	1,251	6.041
Subroutine	NORAN	43	0	2	.074
	INITL	163	35,757	0	.491
	SORT	43	0	0	.147
	BLDG	100	0	0	.208
	CLSTRP	2,621	0	0	.473
	MINCW	2,100	0	3	1.016
	CHECK	257	0	3	.422
	BETWN	264	0	3	.455
	OVLAP	187	0	901	.447
	REPAIR	332	35,757	0	.717
	RESULT	1,119	35,757	17	1.574
	CATLOG	39	35,757	338	.066
	MOVE	53	0	0	.051
	NCOMP	128	35,757	17	.117
Column Totals	(words or secs):	10,736	35,757	1,251	12.299
	(bits):	644,160	2,145,420	75,060	

Total Loader Req'ts: 47,744-Decimal, 60 bit words
135200-Octal, 60 bit words
2.86 Megabits (MB)

AAPMOD		Program Unit Length (words)	Blank Common (words)	Labelled Common (words)	Time (secs)
Program	MAIN	1,882	6,621	1,228	3.473
Subroutine	NORAN	24	0	1	.057
	INITL	70	6,621	0	.269
	SORT	43	0	0	.152
	BLDG	100	0	0	.211
	CLSTRP	2,632	0	0	.507
	MINCW	2,099	0	3	1.033
	CHECK	260	0	3	.445
	BETWN	264	0	3	.461
	OVLAP	187	0	901	.455
	REPAIR	321	6,621	0	.715
	RESLTS	993	6,621	15	1.325
	NCOMP	73	6,621	15	.241
Column Totals	(words or secs):	8,948	6,621	1,231	9.344
	(bits):	536,880	397,260	73,860	

Total Loader Req'ts: 16,800-Decimal, 60 bit words
40640-Octal, 60 bit words
1.01 MB

Core Memory Requirement Reduced 65%

2/ Compiler optimized the binary file at LEVEL-2, and supressed DEBUG utilities.

structure, enabling later embellishment of the program. The resulting statistics for AAPMOD, compiled with the same options as OLDAAP, are also presented in Table I. 3/

A large part of the 3,287 words of AAP PROGRAM--MAIN was trapping errors, and producing output as directed by user options. In response, AAPIN was developed to control inputs. Additionally, long output versus short output, random number storage and other "nice", but costly output options were eliminated. For example, the results of the conversion included a 42.6% reduction in words in PROGRAM--MAIN. Elimination of about 50, formatted, input error messages alone saved 254 words.

To further reduce the size of the program, superfluous routines such as MOVE and CATLOG were cut. Nowhere in the program was there a call to SUBROUTINE--MOVE. Discussion with Eglin indicates the routine may be left over from earlier versions, where it may have been used to move the minimum clear TOL rectangle, while executing the clear area search.

SUBROUTINE--CATLOG was an emergency save routine. Armed by an early call to CYBER intrinsic routine, RECOVR, CATLOG would execute if AAP abnormally terminated for a reason other than a fatal, run-time error. Such termination might have occurred if the requested job time was too short or the operating system glitched.

This study continued to use the CDC CYBER, and CYBER reliability in interactive execution was considered high. Class polls revealed no instance of debugged, operational programs abnormally terminating during interactive

---

3/ BLOCK-IF statements replaced many of the originally 110 GO TO's in PROGRAM--MAIN of AAP. AAPMOD retained only 28 GO TO's. Used at weapon reliability check-points, these 28 avoid sixth and seventh level IF statements, by stepping loop controls when weapons fail to release or properly function.

sessions. However, the intent of this work is to make AAPMOD transportable, for use by MAJCOM level or lower, where CYBER access may not be available. Therefore, to reduce program size and maintain transportability, and to permit faster execution times, intermediate data saves were eliminated.

AAPIN

A large part of the utility of AAPMOD comes from the front-end, user-friendly AAPIN. AAPIN not only makes it easy to run AAPMOD, it also reduces loader requirements and enables fast, interactive execution. To facilitate their discussion, inputs to AAPMOD will be discussed under the topic of AAPIN.

AAPIN generates the input-file for AAPMOD. As discussed earlier, a significant part of the main program loader size reduction is due to elimination of input error trapping, now accomplished in AAPIN. Therefore, the file produced by AAPIN can be considered "laundered", and the user can expect normal execution of AAPMOD.

There are basically four categories of inputs to AAPMOD. These categories are as follows:

1) Program Control,
2) Target Data,
3) Attack Data, and
4) Crater Size.

Program Controls. The user can control certain aspects of the attack simulation. The most obvious is control of the random number generation by setting its seed. Given the nature of AAPMOD, the user can change axis-of-attack and not affect the random number stream. Similarly, individual weapon reliabilities can be changed

Table II
Execution Times for Benchmark Runs

	WITH AREA TOTAL	WITHOUT AREA TOTAL
Bench	18.53 secs 19.43 secs	83.98 secs 85.08 secs
Test	0.98 secs 1.01 secs	10.47 secs 10.45 secs

without losing random number stream synchronization between runs. However, due to Fortran's lack of different random number streams, some other changes lose synchronization. Specifically, if either aircraft survivability or cannister opening reliability change, synchronization is lost. Ideally, reliabilities or survivability should be on one stream, and aimpoint errors and weapon ballistic errors should be on another.

Another obvious input factor is the maximum number of iterations. However, a facility in the program enables a subroutine of AAP/AAPMOD to reduce the number of iterations accomplished. The operation of SUBROUTINE--NCOMP, that accomplishes the reduction, will be discussed later. But upon input, if the user requests over 200 samples, and agrees to allow AAPMOD to reduce sample size, the user is prompted for the $Z_{\alpha/2}$ for their desired confidence. Thereafter, the user enters his allowable error: the difference between the Monte Carlo produced estimate of probability and the true population probability.

The next control is the interval for output of intermediate results. This was a convenient development facility, and now can be used to assess response variance.

Execution time is severely affected by whether or not the user chooses to compute the total area of crater damage, per target element. Runtimes presented in Table II document the additional time required to compute the total area damaged. The increase is due to execution time of checking for overlapping craters. When developing new weapons, such data is an important consideration. Also, if AAPMOD is run tactically, and the user wants to study time required for repairs, such data is needed. But for the expected utility of AAPMOD, this option may normally be suppressed.

This concludes the section on program control inputs. The discussion continues with the input of target descriptions.

Target Complex. The following three sections will discuss program inputs and enable fast development of input files. The user can then quickly analyze the outcome of defined missions against defined targets. This section on definition of the target complex is first.

Initially, the user inputs the numbers of targets and groups. Although some inputs become redundant, they are included for error trapping, to ensure the user enters values consistent with his intent.

Given the requirement for smaller loader requirements, the most obvious savings stems from reducing the large arrays used in AAP. Implicit with reducing array size is reducing capability. Table III tabulates the new limits of AAPMOD, and contrasts them to AAP.

Inputting target data is straightforward, as AAPIN leads the user through all data required to define the complex. The target complex is input referenced to a positive right-hand, rectangular coordinate system, defined by the user. Since most assessments will include runway attacks, it is recommended the center of the runway form the center of the target-complex coordinate system. Either feet or meters can be used in AAPMOD, but the user must be

Table III
Capability Comparison

AAPMOD	LIMIT	AAP
112	Target Elements	207
30	Pavements	43
3	TOL Surfaces	3
1	Attacks	10
32	Passes/Attack	64
15	Target Groups	20
12	Weapon Patterns	16
12	Weapons/Pattern	36
11	Hardness Codes	11
6	Warhead Codes	6
1	Reattack Passes/Aircraft	Unlmtd

consistent throughout.

AAPMOD permits trade-off studies between attacking the take-off and land (TOL) surfaces or their approaches. On entry, AAPIN categorizes pavements as either TOL capable or taxi-only capable. When a prompt requests the minimum clear length for TOL operations, entering "0" flags the pavement as a minor-taxiway. The search for the minimum clear TOL area will then be suppressed. But in any case, AAPMOD does search for meandering taxi capability, to determine if approach to the clear strip is possible.

Crater Data. Damage assessment in AAPMOD is done by checking craters from all detonating warheads, and assigning damage to targets that intersect the crater. A single crater can damage more than one target element.

Cratering is the damage mechanism of AAPMOD. The user inputs the expected crater size for the interaction of warhead code, target hardness code, and type of impact engagement (i.e., hit or near-miss). This is an important concept. The same warhead, a 500-pound GP bomb, for example, can have different warhead codes, against the same hardness code, if by changing impact velocity or angle, the size of the crater varies. (After careful consideration,

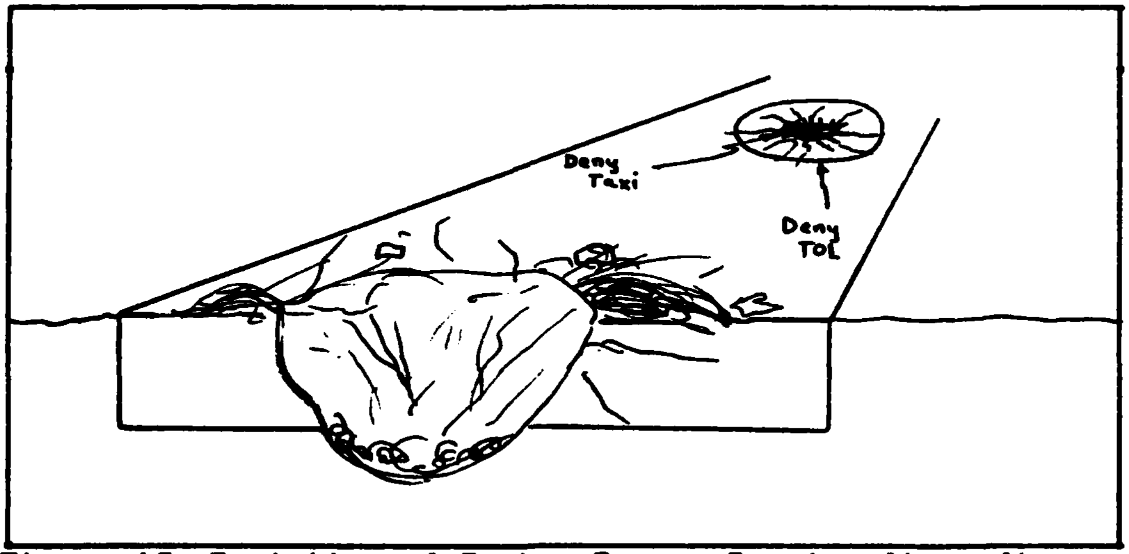


Figure 1Ø Depiction of Crater Damage Denying Aircraft Operations from a Pavement.

it was decided that internally computing crater sizes was not worth the increased execution times and loader requirements such computations require. To appreciably gain precision, the weapon trajectory would have to be modeled to a level of detail beyond that found in the rest of the program. An intermediate choice could have been to input impact velocity and angle, but that data is no more readily available than is crater size.)

The user, guided by AAPIN, creates the 3-D crater array needed by AAPMOD. For each combination of hardness code and warhead code, AAPIN requires two crater diameters. If the hardness code applies to a pavement, the two sizes relate to the size of the disruption severe enough to deny taxi operations, or to deny TOL operations. A profile view of a crater in a pavement is provided in Figure 1Ø, and illustrates the requirement for the two dimensions. Whereas an aircraft may be able to slowly taxi over small cracks, perhaps with the aid of a ground marshaller, high-speed take-off or landing operations, with its less precise tire positioning, will be denied over a much larger area.

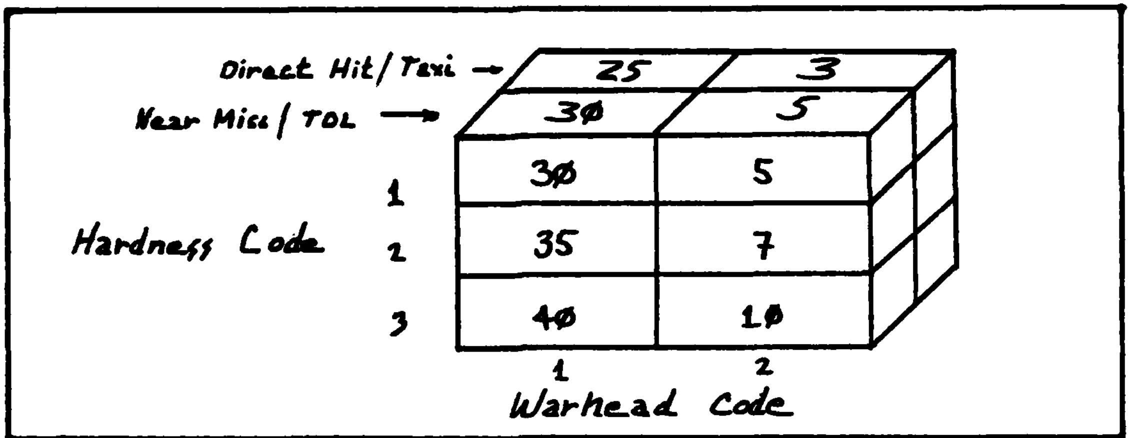


Figure 11 Illustration of 3-D Crater Radius Storage Array.

Conversely, when discussing structures or buildings, the crater will generally be smaller for near misses than for direct hits. The difference is due to the less severe weapons effects of the near-miss over the direct hit.

As mentioned earlier, crater size is one of the Armament Lab's primary considerations in weapons development, so crater size drives many of the analyses with AAP. Crater dimensions are normally supplied by the weapon developer. Since Eglin is often tasked to determine sensitivity to varying crater size, the tactical user of AAPMOD can obtain crater size either from classified weapons documents, or from classified tables produced at Eglin during weapon tests.

To illustrate the crater array, the data of the TEST and BENCH programs is depicted in Figure 11. These four programs considered damage due to three different hardness codes and two different warhead codes.

AAPMOD uses square craters when assessing damage. The craters are aligned with the user input target-element orientation when evaluated for hit/near-miss status. Since tactical planners generally think of circular craters, the user of AAPIN can select the input option. If square dimensions are available, one half the side of the square

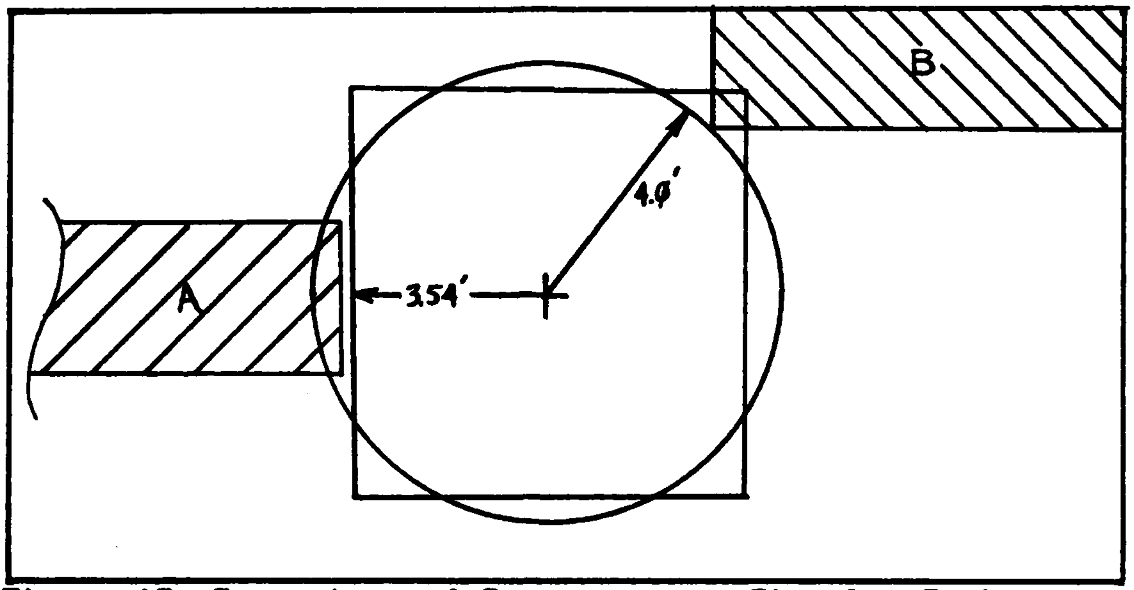


Figure 12 Comparison of Square versus Circular Craters.

is entered. Else, if crater diameters are available the user inputs the crater radius. AAPIN then transforms the radius into an equivalent square dimension. The square dimension is written to the input file as one-half the length of a side of a square, having the same area as that of the user's circle. The calculation simply multiplies the input radius by SQRT(PI/4). In illustration, if the user had 8' square craters, they would enter 4', as one-half length of a side. If the user instead, had craters with a 4' radius, AAPIN would store 3.54', so that AAPMOD will use a crater area of 50 sq ft, just as if the 4' radius had been used.

The use of square craters, aligned with the target axis facilitates the search routines to determine open or closed status of runways. And, as Figure 12 demonstrates, the approximation is good. There is equal likelihood that damage will be missed when using squares, such as to element A, as is there the likelihood that damage will be counted when it should not, as with B. Over the course of the simulation, the average error will null itself out.

Attack Data. Finally, the user inputs the mission scenario. AAPMOD restricts the analysis to one attack. Considering the purpose of AAPMOD, this is not unreasonable. Tacticians and planners at a level lower than full Allied Tactical Air Force, will exercise the program. Realistically, the limited fighter-bomber resources, available to NATO, or even a wing commander, will not allow large-scale, repeated attacks against the same target. Therefore, single attack results, in the form of probability of cutting each TOL surface, probability of denying all TOL operations, and the expected number of craters that must be repaired to regain TOL capability, will provide adequate planning data to effectively employ this level of force during conflict.

A further restriction of AAPMOD concerns the number of reattacks permitted of any aircraft. AAPMOD allows a maximum of one reattack. Again, this is not unreasonable. Tactically, even one pass might be too many, in a high threat environment. Very few crews will intentionally withhold weapons, and plan to fly a second or third pass over a target. Not only have they lost the favor of surprise, but the defenders still alive at the target are pretty angry!

As developed, AAPIN is a user friendly, input file generation program for AAPMOD. AAPIN will allow crews, commanders, and planners to use AAPMOD, to study tactics, the weapons they have available, and whatever targets they might be directed to attack, and to optimally attack the elements of the target to produce the best damage. AAPMOD, exercised at higher operational levels, such as MAJCOMS or at advanced fighter weapons schools, can also prepare the decision makers to make realistic weapon system assignments, so to optimally use their limited attack assets. The discussion of computer implementation continues with the discussion of the execution of AAPMOD.

AAPMOD

Given the defined target complex, the defined attack parameters, and the defined crater sizes, AAPMOD assesses expected damage to the target complex. It is superior to some alternative assessment programs in that it considers collateral damage, in addition to assessing the damage expectancy to the desired target. That is to say, if a weapon or stick of weapons miss their mark, they may cause desirable, though unintended, damage to other, closely located, target elements. Given such design, AAPMOD analyzes the target and the attack as part of an interacting system, and not as isolated elements or entire systems of themselves.

However, a disadvantage of AAPMOD is its simplified use of cratering as the damage mechanism. Cratering is the classical damage considered for pavements. And it is expected that most studies run with AAPMOD will key on runway or taxi surface damage. Under these circumstances, this disadvantage is minimized. However, the application of AAPMOD to building, or non-pavement type structure damage, is less than optimal. AAPMOD iteratively subtracts the intersecting area of a crater from the remaining, undamaged area of the target. The final output is simply total area damaged. No consideration is given to individual areas of target vulnerability. Also, the program does not specifically consider either the blast or the fragment-spray damage mechanisms.

Whereas cratering is adequate in analysis of area targets, and can possibly be extended to uniformly vulnerable, hardened buildings, it is insufficient in the assessment of damage to softer targets like radar vans, cargo trucks, or communication devices. Nevertheless, when estimates are required, AAPMOD can be made to work for structural damage. One must assume cookie-cutter damage functions. Cookie-cutter implies the delta function, and means that inside a defined range the target is killed, and beyond that range the target survives. But then the output will not be as rigorous as that offered in the analysis of pavements.

The design of AAPMOD is simple. Monte Carlo techniques simulate weapons deliveries according to specified parameters, such as attrition, accuracy, fuze reliability and the other variables discussed in Chapter III. Each Monte Carlo loop represents a planned attack. An attack consists of up to 32 aircraft, flying up to 32 weapons delivery passes across the target. Each weapons pass is described by aimpoint and the other parameters discussed under the section on inputs, this chapter. As each pass is made, impact locations are simulated, and each target is checked for a hit or near-miss. Both are recorded and the attack continues. At the end of the attack, overall damage to each target element is assessed. Results for each iteration are accumulated, and an average and standard deviation of damage expectancy is computed. Finally, a carry-over from AAP that has been retained to enable future embellishment, considers post-attack, airbase repair capability.

Program Execution. AAPMOD is designed with structured programming techniques. The clarity built into AAPMOD, over AAP, will enable later analysts to further modify and enhance AAPMOD to produce output precisely as desired. The discussion of program execution emphasizes PROGRAM--MAIN, but is followed with a brief listing of subroutines and program outputs.

Execution of AAPMOD closely resembles execution of AAP. Immediately after input and input echo, AAPMOD sets some control flags, and begins the sampling process. The iterations of the attack form the first level of program control. This is followed by loops on attack passes. Each pass is first assessed for survival of enroute defenses. If the aircraft survives to the release point, weapons release occurs according to the defined weapon pattern for the pass number. All weapons are released, but the formation of craters, and their location, is the primary stochastic assignment of AAPMOD. Each weapon must pass a test for fuze functioning. If the weapon is a point-impact, unitary weapon, a crater is assigned. If the weapon is a cluster unit, the probabilistic check represents cannister opening. If this first reliability check fails, the rest of the weapon loop is by-passed, and the next weapon of the pass is examined.

If the weapon functioned properly, the center of its impact is located. For point weapons, this is the point of impact. For area weapons, this is the center of the footprint of the submunitions encased in the weapon. For precision guided munitions (PGM), the point of impact is a stochastic variable drawn from one of three distributions. The X and Y coordinates are drawn from normal distributions with parameters representing optimal guidance, sub-optimal guidance, and ballistic, gross-errors. AAPMOD first determines the type of guidance of the PGM, then draws corresponding X-Y errors.

The process is less elaborate for unguided weapons, such as bombs, released singly or in a stick, or for CBU's. The aimpoint stored for the attack in both cases is first adjusted for aircraft and pilot induced errors. If the pattern calls for a single release, the location of the single weapon is displaced, representing ballistic dispersion of the weapon's free-fall (discussed in Chapter III). If, however, the weapon pattern was a stick release, the adjusted aimpoint forms the mean point of impact (MPI). In other words, the stick of weapons fall in a pattern centered on the MPI. Again, the discussion in Chapter III addressed the development of the pattern of impacts resulting from a stick release.

Once the MPI of the stick is defined, the locations of each impact in the stick are adjusted by separate, individual draws from the normal distribution representing ballistic dispersion.

The numerous impacts from CBU's carry this concept one step further. By their nature, the above described process determines the center of the footprint, or the center of the area covered by the distribution of submunitions. One of the assumptions of AAPMOD, and one of the limitations of the analysis, appear in the location assignment of CBU bomblets. The assumption is that the distribution of impacts over the described footprint is uniform. The limitation is that footprint voids, or areas within a footprint without bomblet coverage, are only permitted when processing elliptical footprints, and not rectangular footprints.

Initially, each bomblet is checked for fuze functioning. If the submunition worked, X-Y coordinates are assigned according to the assumption, limitation, and weapon pattern parameters entered by the user.

Whatever the type or size, the location of each crater is compared to every target element, to determine hit/miss status. Hits are stored, and the program continues in this series of loops until the attack is complete.

Overall, each iteration of Monte Carlo sampling can be described with the following series of nested loops:

```
DO (for each pass)
    DO (for each weapon)
        DO (for each warhead)
            DO (for each target element)
                check for a hit/near-miss
                save hits/near-misses
            NEXT element
        NEXT warhead
    NEXT weapon
NEXT pass
```

When, within a sampling iteration, the attack has been flown out, the program enters assessment phase. Assessing building damage is easiest, and will be discussed first, followed by minor taxi-ways, followed by TOL capable surfaces.

When assessing building damage, AAPMOD computes the total cratered area of the building or structure. Each hit or near-miss reduces the effective area of the structure with a call to SUBROUTINE--BLDG. Output of accumulated area only occurs when the long computation of total damaged area is requested. Currently, AAPMOD has been designed to compute total damage, but due to some unchanged logic of the original AAP, such printout is suppressed if computation of total area of pavement damage is suppressed.

The user is also provided damage area data concerning target element groups. As entered, each target element belongs to a target group. Perhaps several target elements are identical, except for location. An example may be three or four POL tanks, or a set of redundant approach aids. AAPMOD also computes output statistics for each target group.

The assessment of damage to pavements is more complex. Target elements that are not buildings or structures are pavements. And, if the required clear dimensions exist anywhere on the original surface, though some damage may have occurred, the function of the surface has not been denied.

The user's choice first controls whether a call to

SUBROUTINE--OVLAP computes the total area of crater damage for the pavement of interest. The time considerations of this decision have been addressed earlier. The assessment then continues with a decision. If the pavement is a minor taxi-way, without take-off and land (TOL) capability, a call to SUBROUTINE--MINCW searches for a meandering path of at least the minimum taxi width. If a clear meandering path is not found, AAPMOD will repair a clear path. The program will sum the area of crater disruption that must be repaired to enable taxi operations.

If, however, the pavement is a runway, or at least one of the three allowable TOL capable surfaces, the assessment algorithm delays the search for a meandering path. Some bookkeeping takes place as codes, counters, and sums are initialized. Then, initialization is followed with a call to SUBROUTINE--CLSTRP. CLSTRP searches for a clear operating area. If none exists, the runway is cut, and the area of crater damage, that must be repaired to enable TOL operations, is computed.

The assessment for the clear strip requirement is repeated for up to three TOL capable surfaces. In the real world, the airfield stays open if any of the three surfaces are not cut. Similarly, the airfield will reopen if any of the three surfaces are repaired. Therefore, AAPMOD computes cumulative statistics for the probability of denying a clear strip on all three TOL surfaces. (Unless the attack reflects substantial efforts, this probability is usually low.) Also, AAPMOD offers, similar to the taxi results, an expected number of craters that must be repaired to regain field operations.

However, it is felt that as is, the expected number of craters is deceiving. Currently, the output value simply reflects the cumulative number of craters actually denying TOL capability from the easiest, minimum clear strip to repair, divided by the number of samples. So on iterations where the runway is left open, a zero is added to the sum. An embellishment to AAPMOD, when more accurate consideration is required of repair capability, would be to determine a more appropriate computation for expected crater area to repair.

Finally, assessment includes one more search. After computing the probability of denying TOL capability for the airfield, AAPMOD computes the cumulative number and area of craters, requiring repair, to enable approach to the easiest minimum strip to repair. But again, this figure is a total divided by the number of iterations.

Assessment, as described above, occurs in each iteration of the Monte Carlo loop. Then, at the conclusion of the assessment, AAPMOD processes the data. Statistics collected along the way include real values such as areas, but also some integer counts. Additionally, squares of values are collected, to be later used to provide standard deviations (S/D) of some results.

At user defined output intervals, or by default at 200 samples, the data is processed, and printed to file. After the 200th iteration, AAPMOD may call SUBROUTINE--NCOMP, to determine the additional iterations required to ensure the user specified accuracy. If required, additional samples are taken, and again, output is printed on interval or at program completion.

As shown, execution of PROGRAM--MAIN is straightforward. AAPMOD uses very few assumptions or approximations: none beyond those previously addressed.

The next part of this section presents details of specific subroutines used by AAPMOD. These details may be omitted by the less technically oriented reader. However, the chapter resumes later, with a discussion of the model outputs.

TRISUB: This subroutine provides the random variates for the aimpoint error. The routine draws from a triangular distribution, taken from Law and Kelton (Ref.12:261). The subroutine is hardwired for a mean of zero, and high and low extremes of +/- 1,000'.

NORAN: An older, proven generator for normal, random deviates. The technique uses the exact inverse method, proposed by Box and Muller. Shannon [19] considers the method "accurate, easy..., and... fast.", while Law and Kelton [12] feel the routine should be replaced by more efficient methods.

Based on limited calls to NORAN, not exceeding 300 per iteration, it was decided to retain the Box-Muller method. Later, if AAPMOD is implemented on a slower computer, consideration should be given to replacing NORAN.

SORT: Calls to SORT arrange the arrays, or parts of the arrays of hits or near misses, in ascending numerical order. Various keys for the sort are set by the order of arguments passed from the calling routine.

BLDG: SUBROUTINE--BLDG assesses the crater damage occurring to buildings. The call is made with the complete set of craters, intersecting the given target element, passed as arguments. Within the routine, each crater successively reduces the area of the building remaining. With each area reduction, the length and width of the structure are reduced in the original ratio of length to width of the building.

CLSTRP: CLSTRP assesses denial of TOL capability. Recall from Chapter III that different denial potential exists for a given set of craters, depending on whether the denial affects taxi or TOL. CLSTRP searches for a clear area of the minimum clear dimensions input by the user. AAPMOD moves a rectangle over the original runway to see if the clear area exists. The clear area must be a rectangle.

While performing the search, CLSTRP also records the area of craters intersecting the moving rectangle. If the current location has less crater area than the previous, the current block becomes the "easiest strip to repair." If a clear block is found, the runway is open and crater area is zero.

MINCW: MINCW searches for a meandering taxi path. The hits array is passed to MINCW after being sorted by X coordinates. The subroutine first partitions the search into a number of subproblems. A subproblem is a group of craters with X distance separations all less than the minimum taxi width. Thereafter, each subproblem is checked for a cut that denies the required minimum width between craters across the pavement.

CHECK: CHECK is called by MINCW to perform the check for the cut, once the subproblem has been partitioned.

BETWN: BETWN is further called by CHECK to determine if an aircraft can taxi between two craters. BETWN considers the capability of the aircraft to meander its way between the craters.

OVLAP: OVLAP is a time consuming search for the true area of crater damage. The subroutine searches for overlapping areas of craters, and reduces the damage area by the area of overlap. OVLAP is costly in execution time.

REPAIR: Based on the user entered priority, REPAIR repairs craters. Its function in AAP was more important than in AAPMOD. AAP allowed several attacks, with a defined capability of the airbase to repair craters between attacks. REPAIR simply computes this number, and eliminates the repaired craters from the hits array. By virtue of the one attack limitation of AAPMOD, REPAIR's utility is decreased. However, it has been retained to enable flexibility if future modifications to AAPMOD are required.

RESLTS: As the above routines assess the damage to the complex, data is stored in separate arrays. As mentioned earlier, data is stored as both a simple measure, and as a squared value. Such storage simplifies the work of RESLTS, reducing its task to simple calculation of means and standard deviations (S/D). Also because of the summations, calls to RESLTS only occur at user selected output intervals, or by default at program termination.

NCOMP: NCOMP computes the number of iterations required for the Monte Carlo loop. With user consent, NCOMP can reduce the number of samples used for the program run.

As mentioned earlier, the open-closed status of any one runway or TOL surface, is singly a Bernoulli trial. But the call to NCOMP occurs after iteration number 200, so the distribution of sample results can be approximated with a normal distribution.

NCOMP uses a slight deviation from the sample size equation presented earlier. Shannon's 1/4 factor represents the highest product of the probability of success and probability of failure in a simple trial. If p=0.5, q would also equal 0.5, and their product would yield 0.25, or 1/4. However, if sampling reveals a p equal to something other than 0.5, the product of p and q will

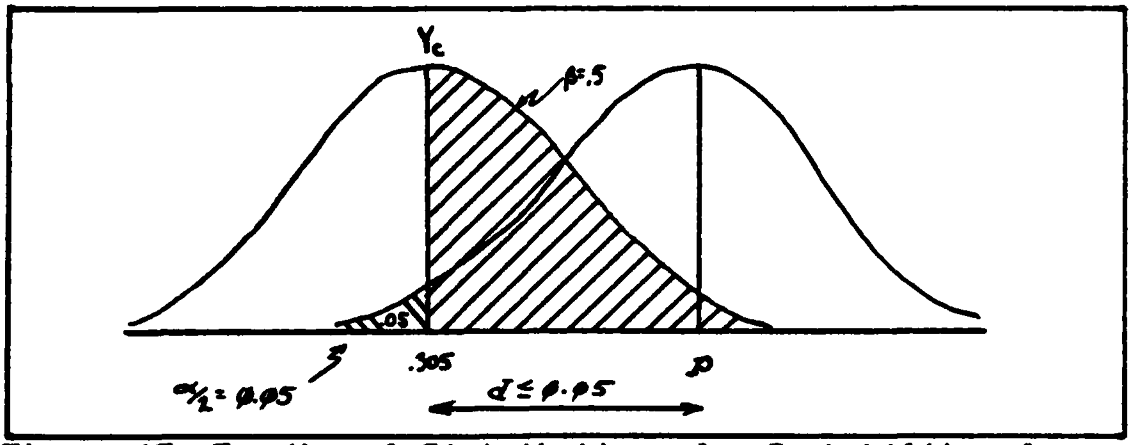


Figure 13 Two Normal Distributions for Probability of Closure of a Target Element.

always be less than Ø.25. AAPMOD uses the product of the observed p times q as a reduced factor for multiplying with the $Z_{\alpha/2}^2$ and d^2 term.

For example, the following situation developed during the validation runs of AAPMOD:

Results for iteration # 2ØØ revealed p =.3Ø5 and q = .695. The confidence requested was 9Ø%, and the deviation from the true population probability was desired less than Ø.Ø5.

NCOMP computed a sample size of 23Ø, and program execution stopped on that iteration. The computation occurred as follows:

$$n = p * q * (Z_{\alpha/2}**2 / d**2)$$

$$n = (.3Ø5)*(.695)*(1.645/.Ø5)**2 = 23Ø$$

where: $Z_{\alpha/2} = 1.645$ $\quad d = Ø.Ø5$
$p = Ø.3Ø5$ $\quad q = Ø.695$

The concept can be explained as follows: the true probability of the defined attack closing the runway is unknown, but assumed to equal *p*. The sample size of 2ØØ revealed a sample mean of Ø.3Ø5. If drawn with two normal distributions, the situation resembles Figure 13.
The sample size, *n*, is then computed to satisfy the above

equation, and ensure the desired accuracy of p.

And recall that p, in this discussion, refers to the probability of closing any one TOL surface. Sample size reductions do not occur for elements other than the three TOL surfaces. Also, NCOMP considers the worst case probability (the p closest to Ø.5) from all three TOL surfaces.

AAPMOD Output. Earlier, when addressing inadequacies of the JMEM effectiveness method, a question was raised concerning application of force levels less than that required to cut a runway. The simple answer is to run the analysis again, and determine new results. However, when using AAPMOD, reanalysis may not be necessary. The output produced by AAPMOD offers the user many measures in addition to an overall probability of denying TOL operation at an airfield. The following discussion will relate the various output quantities of AAPMOD, to the various system responses of Chapter III.

The output of AAPMOD is obtained from simple data collection and storage during each iteration of sampling. PROGRAM--MAIN calls SUBROUTINE-RESULTS to process the data and print the output.

A sample of AAPMOD output appears in Appendix F. The reader should refer to Appendix F as the various elements of output are discussed.

The first part of the output consists of an echo of raw input. This simple procedure saves hours of trouble analyzing program output that may not have been expected. Most programmers know that computers do what you *tell* them to do, not what you *want* them to do. Well, in the same way, computer *programs* process the data that they *are* given, and not what they *should have* been given.

The reader may note, that in the sample of Appendix

F, by virtue of "Ø" preceding the default Z and ERROR terms of 1.645 and Ø.Ø5, sample size is not reduced. (ERROR is the program variable for the *d*, discussed earlier.) The user requested 25Ø iterations with intermediate output at the 1ØØth and 2ØØth iterations. Since output format is identical for any iteration, skip to the last set of values, found at NSAMP = 25Ø.

The first values that appear are confidence limits. The numbers represent one-half the width of an interval centered on the sample estimate for the expected number of hits on the target element listed. This element will be the target element in the complex with the highest number of expected hits, as reported in the line labeled, "EXP NO. HITS", found just below the confidence limits.

The value for the confidence interval is computed from the Student's *t* statistic, the S/D of the sample, and the number of the current iteration. For example, the sample reports 1.25 for the 99% level. This is computed from $t_{\alpha/2}$ =2.576, s=7.674, and n=25Ø, as follows:

$$1.25 = t_{1-\alpha/2} * \frac{S}{\sqrt{n}} = 2.576 * \frac{7.674}{\sqrt{25\emptyset}}$$

The 1.25 creates a 99% confidence interval that ranges from 19.9 to 22.4 for the main runway. The meaning of a confidence interval is that if the 25Ø iterations were repeated 1ØØ times, and a correct interval was computed for each replication, 99 of the 1ØØ intervals would include the true, expected number of hits, for the defined attack on the defined target.

After reporting the expected number of hits and its S/D, the output continues with "EXP AREA DAM". This value represents the expected, accumulated area of crater damage, per target element. And again, the expected value is accompanied its S/D.

Since total damage area calculations were not suppressed for the run, the values for damage area are reported. The price was paid, however, as the series of program runs, addressed in Chapter V, and illustrated in Appendix F, ranged from 66.5 seconds to 135 seconds of processing time.

The output format completes the individual target element information with a reminder of the group to which each target element belongs. Afterwards, the output turns to data concerning group damage. Area of group damage is simply the sum of the damaged area of the member elements of the group. The S/D for the group is computed separately, however. Each iteration contributes a squared term to a running sum of squares.

Next is data concerning TOL pavements and minor taxiways. For each of the up to three TOL capable surfaces, AAPMOD computes the probability of denying TOL operation from the strip, as well as the S/D of the probability estimator. Upon output, AAPMOD simplifies the label as "PROB CUT". However, if a TOL strip is so large to require two or more cuts, the output value is really the probability of denying a clear operations area and not only the probability of cutting the runway.

The demonstration experiment in Appendix F clearly illustrates the concept of cut versus closure. Runway-1 is 9,000' x 200'. Runway-2 is 6,000' x 100'. To close Runway-2 is to deny operations from Runway-2. To deny operations is to deny a minimum clear rectangle of 4,000' x 50'. Given the original dimensions of Runway-2, simply producing a cut more than 2000' from either end, denies the minimum clear area.

A runway cut is a chain of craters, across the runway, with a minimum width between craters of less than, in this case, 50'. However, one cut is not sufficient to deny TOL operations from Runway-1. At least two cuts must be made, and in the correct location to deny the clear, 4,000' length.

The next values are "EXP NO CRATERS" and "SIGMA". This is an interesting measure of the degree of damage in closing the runway. The label stands for the expected number of craters closing the easiest minimum clear strip to repair. This number represents the average number of craters denying a clear TOL surface, over the number of samples of the simulation. The number is computed by summing the integer number of craters closing the runway on each iteration. The expected number of craters closing the TOL area, given the runway is closed, can be figured by simply dividing the EXP NO CRATERS by the PROB CUT.

A related value and its S/D is "EXP AREA FILL". This number gives the area of disruption that must be repaired, to regain operational status. The area can be less than a full crater because AAPMOD sums the area of crater intersecting the easiest clear area to repair. A quarter of the area of two separate craters may need to be filled to regain the minimum clear TOL dimensions, so although EX AREA FILL correlates to EXP NO CRATERS, no direct mathematics relates the two.

The indefinite values found at "EXP NO FILLED" and its "SIGMA" occur because AAP originally repaired craters between attacks. Given the purposes of AAPMOD, although the call to compute the total area repaired was not changed, neither was its location in the loop, and with only one attack, was not excuted. Regardless, the subprograms to compute the repair data have been retained.

Finally, "EXP APPR NO CRATERS" and EXP APPR FILL" extend damage assessment to taxiing capability to either:

1) the clear strip that permits TOL operations, if the attack fails; or

2) the easiest TOL surface area to repair.

These values detail the damage that inhibits access to the clear, or nearly clear strip, from the end of the runway. The value is computed similar to above, where an integer number of craters is accumulated into a sum, and with floating point accuracy, is divided by the number of iterations. The same also occurs for the area computation.

The same data is presented for all combinations of major TOL surfaces. Had the example problem contained three TOL surfaces, the data presented on the line "1 & 2" would have been replicated for "2 & 3" "1 & 3", and "1 & 2 & 3". One notes the additional information of the "DISTRIBUTION MINIMUM CRATERS".

Finally, AAPMOD output details the damage to minor taxiways. Given that taxi operation requires only a minimum clear width, without associated length, "EXPECTED NUMBER OF CUTS" is alone a valid descriptor. However, a cut to deny taxi is not as simple as a cut to deny TOL. Recall the figure presented in Chapter III, Figure 9. AAPMOD considers meandering taxi. Figure 9, with distorted scale illustrates the difference between denying TOL and taxi operations. Recall also, the radius of disruption decreases for taxi, which clarifies the use of different sized craters in Figure 9. Very simply, if a chain of craters precludes taxi, the pavement is cut. Such cuts can occur anywhere, and in any number, along a taxiway. And each cut is considered an event, complete of itself. There is no requirement for exact location, because there is no requirement to deny a minimum length of minimum width.

This completes the discussion of implementation of AAPMOD. Chapter V, that follows, is the presentation of the results of an experiment run with AAPMOD.

V. Program Demonstration

This chapter will report the results of a simple, three factor, two level experiment run with AAPMOD. The results of the experiment are interesting in themselves; however, the significance of the experiment remains in the demonstration of the capabilities of AAPMOD.

Chapter V consists of two sections. The first is on experimental design, and discusses the experiment. The second addresses the sensitivity of the results of the experiment to changes in the inputs. Implicit with the discussion of these sections is further evidence of the validity and the veracity of AAPMOD.

Experimental Design

AAPMOD is a tool: a data processor. When inputs to the system of airfield attack are entered into AAPMOD, the program uses Monte Carlo sampling to process the inputs, and produce the system response. AAPMOD is only one tool of many. For example, an expensive alternative may be to actually fly out an attack, and examine the real world interactions of weapons on target. However, if money restricts the number of attacks available to be flown, confidence in results may be low.

Another method, tedious to run, is to perform a JMEM analysis. As the reader is aware, such a technique can only determine the probability of closing a runway.

All of these "tools" use inputs to form a response. And it is felt that the responses of AAPMOD make a significant contribution to a tactical, operational, weapons analysis.

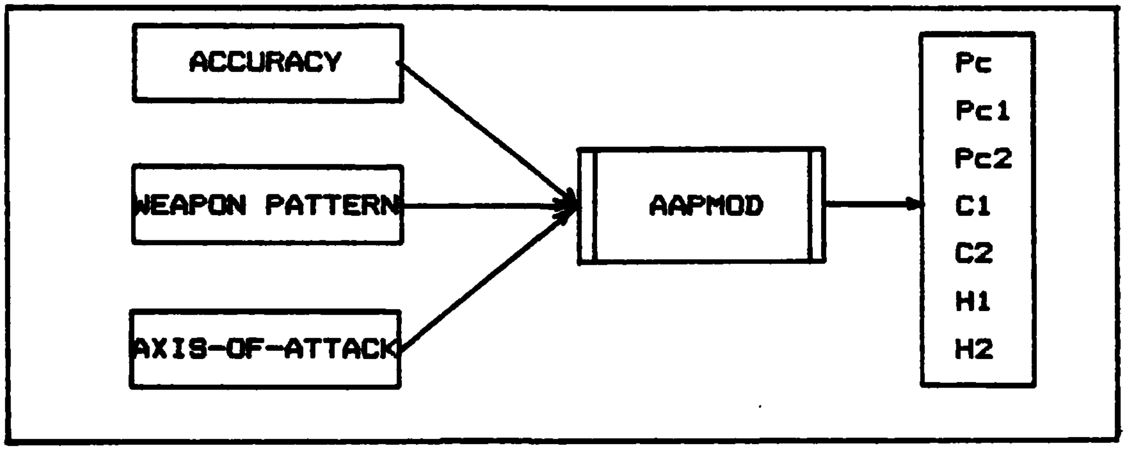


Figure 14 Airfield Attack Experiment

The Simulation. A practical experiment demonstrates the capabilities that AAPMOD offers to a weaponeer. The experiment depicted in Figure 14 investigates the effects of three input variables on seven output responses.

Experience suggests that the following factors will produce interesting effects:

1) Accuracy,
2) Pattern Definition, and
3) Axis-of-Attack

The effects of changing these input variables should be reflected in changes to the following seven responses:

1) Probability of closing both TOL surfaces, Pc.
2) Probability of closing Runway-1, Pc1.
3) Probability of closing Runway-2, Pc2.
4) The expected number of craters actually denying operations from the easiest rectangle to repair on Runway-1, C1.
5) The expected number of craters actually denying operations from the easiest rectangle to repair on Runway-2, C2.
6) The expected number of craters on Runway-1, H1.
7) The expected number of craters on Runway-2, H2.

But first, as with all analyses, the target, the attack, and the damage mechanisms must be defined. Again, the reader is referred to Appendix F for sketches and the full list of details. However, generalized elements of the experiment are discussed in this section.

<u>The Target</u>. The target complex consists of thirteen target elements. These elements belong to four target groups. Ten of the thirteen elements are pavements, and two of the ten pavements are TOL surfaces. Page F-1 is a sketch of the complex, and F-2 is the program produced echo of target data.

The complex was designed to simulate a small airfield. The main runway is 9,000' x 200'. Next to the main runway, separated by 100 yards, is a parallel taxiway, 8,500' x 100'. The right-most 6,000' of the parallel also has TOL capability. Taxiways, as indicated, join the TOL surfaces to the main parking ramp, element #12. Although the ramp is really a pavement, for the experiment, it was declared a structure. Therefore, AAPMOD checked element 12 for the area of crater damage, but did not search for cuts. Finally, elements 11 and 13 are structures representing the control tower and perhaps fuel trucks or piping facilities.

Although the design is simple, it is representative enough of an airfield to demonstrate AAPMOD's potential. Throughout the experiment, the target remained constant, as did the crater data, that follows.

<u>Crater Data</u>. Arbitrarily, three hardness codes were assigned to the various elements. But only one type of warhead code was used. This information is reflected at Column K and Row 43 of Page F-3. The information is stored in a two-dimension crater array, beginning at line 44. The

values for the crater array were available to the planner as circular radii. The radii for denying TOL, for surface-hardness codes 1 and 2 are 18' and 24' respectively. The structure, near-miss radius for surface code 3 is 36'. The radii of taxi-denial or direct hit damage, for the three hardnesses, are respectively: 12', 18', and 24'.

However, the reader will note peculiar numbers stored for the crater radii. The 15.9', for example, represents the 18' circular radius. The 18' has been transformed to one-half the length of a side of a square, having the same area as that of the original circular crater. The area of an 18' circle is 1018 sq ft and the area of a 31.8' square is 1011 sq ft, the difference due to roundoff by AAPIN.

<u>The Attack</u>. The input variables under study were parameters of the attack. In order to keep the research manageable, but at the cost of less than a practical experiment, only three variables were studied. The other input variables were held constant. To repeat, the three factors of interest were accuracy, weapon pattern definition, and axis-of-attack.

The experiment studied the effects of these variables set at two levels each, the high level and the low. A detailed discussion of the levels follows.

1) Delivery error: The input variable was the standard deviation of the normal error, in both range and deflection directions. For purposes of the experiment, the deflection error standard deviation (S/D) was defined to be one-half the range S/D. Therefore, by specifying either the range error probable (REP), or the S/D of range error (the two are related by S/D range = REP/.675), both parameters of the bi-variate error distribution were

specified.

For the experiment reported here, REP was chosen at 20' for high accuracy, and 250' for low accuracy. Therefore, the S/D for high accuracy was 30' and the S/D for low accuracy was 370'. The deflection S/D's were the respective one-half values: 15' and 185'.

2) Weapon Pattern: Recall the many factors affecting the shape of the weapons impact pattern. The single factor chosen for this experiment was mode, set at its only two levels, singles or pairs. High and low correlate to the impact density of the resulting pattern. The low level was defined at singles mode and the high level was defined with pairs release.

3) Axis-of-attack: In the real world, axis-of-attack can vary from 0° angle-off to 90° angle-off. Recall from Chapter II, that few impacts can occur on the runway with a 90° cut. Therefore, for practical, as well as operational considerations, (like defenses along the extended runway centerline), the high and low levels of axis-of-attack were chosen at 40° angle-off and 5° angle-off.

The rest of the attack plan should be evident from the echo beginning on page F-2. The attacks consisted of six passes with identical parameters, except for aimpoint. Three aircraft attack approximately the one-third point of the runway, and three attack the two-thirds point. However, the first three aim for the centerline of the runway, and the second three aim for the edge of the runway nearest the parallel taxiway. This 100' displacement offers better collateral damage to the parallel taxiway. However, as the experimental results reveal, the 100' offset was costly to the probability of Runway-1 closure.

In the experiment, accuracy will be termed factor *a*, density termed factor *b*, and axis-of-attack termed factor c. These labels will simplify later discussions. Furthermore, convenience suggests representing the high and low factor levels with 1's and Ø's, respectively.

The Experiment. The purpose of the experiment was to validate and demonstrate AAPMOD. The experiment was designed to display AAPMOD's capability to clarify the effects of levels of input factors on system response. The purpose in this case was not to optimize an attack plan, but to demonstrate that AAPMOD *can* help optimize attack planning.

Considering the demonstrative nature of the experiment, some of the factors that experience suggests affect weapons effectiveness were held constant. Specifically, the probability of aircraft survival was held constant at 1.Ø. (Different delivery tactics could affect survivability and indirectly, probability of closure.)

Also, weapon reliabilities were held at 1.Ø. Again, the relationships addressed in Chapter III suggest that there are subtle interactions between variables not addressed in this brief, three factor experiment. For example, altitude, speed, and dive angle all affect impact angle and therefore reliability. But to assess the interactions of such variables is beyond the scope of this effort. Nevertheless, AAPMOD can handle these interactions, and a larger scale experiment will provide a valuable data base for tactical decision making.

Given the decision to evaluate effects of three input variables, an experimental design was needed. Each attack plan could have been considered a variable itself. In the jargon of statistics, a plan could have been one policy.

An experiment of such design would entail a single factor analysis. However, one-factor policy analyses are weak. All the individual effects of factors, and combinations of factors are lost to the one factor, call it the plan.

Another traditional method is to hold two variables constant, and vary the third. This type of experiment requires replications simply to assess whether responses are due to factor effects or chance. A better design would allow all levels of a given factor to be combined with all levels of every other factor.

And so we have defined a third type of experimental design, a factorial analysis. A factorial design is used in the experiment reported here.

A factorial analysis is an efficient experimental design. When combined with the power of modern, computerized, statistics packages, it clearly describes the effects of not only the factors of interest, but also the effects of the interactions between the factors.

To quote from Hicks (Ref.8:88), some advantages of a factorial experiment include the following:

1) Better efficiency is possible than with one-factor-at-a-time experiments.

2) All data are used in computing all effects.

3) Information is available on the interactions between the factors.

In the three factor, two level experiment, called a 2^3 factorial, the combinations of levels can be visualized as the corners of a cube, as in Figure 15 on the next page.

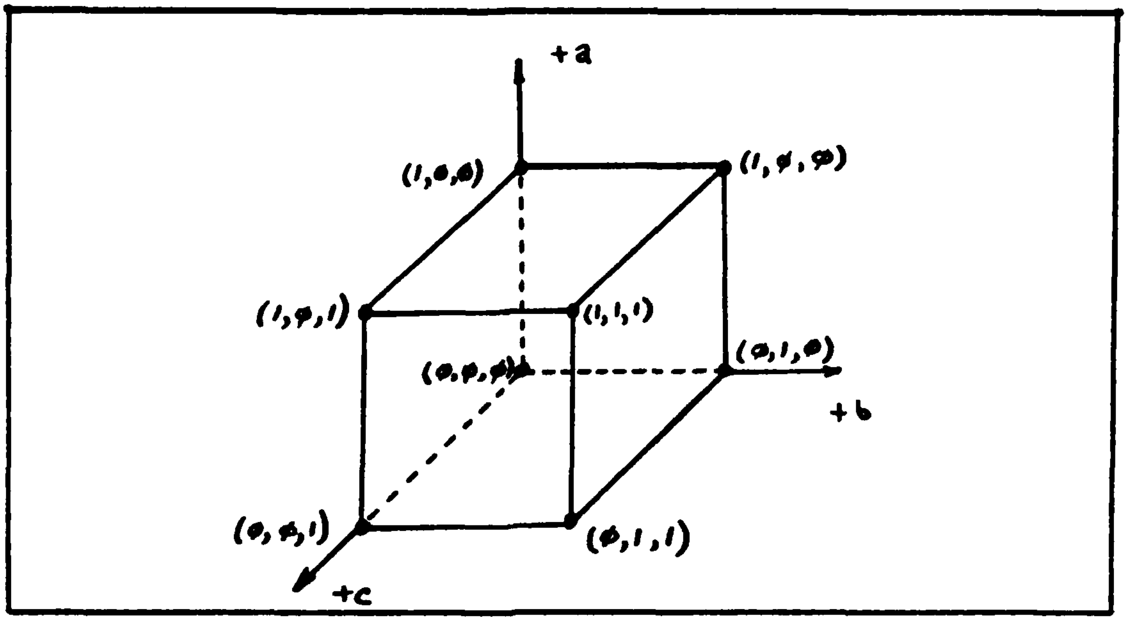


Figure 15 Three Factor, Two Level Experiment Represented as a Cube.

Although actual values were assigned to the levels of the experimental factors, the corners of the cube can be represented by use of 1 and Ø. As discussed above, the 1-Ø represent the high and low levels of the factors.

The Results. Table IV documents the results of running all combinations of factors through five replications. The five replications were chosen to determine if the random number seeds introduced any variability (error) into the experiment. This blocking of runs into groups will be adressed later. More important is the analysis of factor and interaction effects that follows.

Table IV

Experimental Results

Acr'y	Dns'y	Axis	Pc	Pc1	Pc2	C1	C2	H1	H2
0	0	0	.008	.160	.172	1.85	3.74	32.2	2.6
0	0	0	.016	.124	.120	2.22	3.47	32.0	1.0
0	0	0	.016	.116	.184	1.72	3.67	30.1	2.3
0	0	0	.008	.120	.184	1.80	3.45	29.7	2.7
0	0	0	.024	.096	.184	1.42	3.23	30.5	2.6
0	0	1	.176	.408	.520	1.43	2.39	21.1	5.0
0	0	1	.148	.424	.492	1.53	2.28	21.1	4.4
0	0	1	.160	.400	.536	1.50	2.28	21.3	4.8
0	0	1	.160	.384	.540	1.55	2.59	20.5	5.3
0	0	1	.140	.340	.528	1.43	2.49	21.2	4.9
0	1	0	.012	.144	.164	1.72	4.00	32.1	2.7
0	1	0	.012	.116	.112	2.10	3.79	32.1	1.9
0	1	0	.012	.104	.168	1.62	4.00	30.1	2.3
0	1	0	.008	.108	.164	1.74	3.73	29.9	2.7
0	1	0	.028	.092	.192	1.39	3.15	30.5	2.6
0	1	1	.084	.236	.404	1.64	2.68	21.8	4.5
0	1	1	.088	.284	.408	1.62	2.72	21.8	4.4
0	1	1	.104	.272	.440	1.64	2.81	22.2	4.5
0	1	1	.100	.264	.488	1.64	2.94	21.3	5.3
0	1	1	.084	.212	.444	1.70	2.85	22.0	4.8
1	0	0	.000	.000	.000	0.00	0.00	59.4	0.0
1	0	0	.000	.016	.000	1.00	0.00	59.8	0.0
1	0	0	.000	.008	.000	1.00	0.00	59.1	0.0
1	0	0	.000	.004	.000	1.00	0.00	59.0	0.0
1	0	0	.000	.000	.000	0.00	0.00	59.2	0.0
1	0	1	.004	.892	.004	2.50	1.00	44.2	.05
1	0	1	.008	.920	.008	2.70	1.00	43.7	.05
1	0	1	.004	.908	.004	2.43	1.00	43.4	.05
1	0	1	.004	.884	.004	2.50	1.00	43.8	.04
1	0	1	.004	.876	.004	2.53	1.00	44.0	.04
1	1	0	.000	.000	.000	0.00	0.00	59.6	0.0
1	1	0	.000	.008	.000	1.00	0.00	60.0	0.0
1	1	0	.000	.008	.000	1.00	0.00	59.5	0.0
1	1	0	.000	.000	.000	0.00	0.00	59.7	0.0
1	1	0	.000	.000	.000	0.00	0.00	59.7	0.0
1	1	1	.000	.568	.000	1.40	0.00	52.6	.01
1	1	1	.000	.552	.000	1.73	0.00	42.4	.01
1	1	1	.000	.608	.000	1.55	0.00	51.9	.00
1	1	1	.000	.568	.000	1.61	0.00	52.3	.00
1	1	1	.000	.560	.000	1.34	0.00	52.2	.00

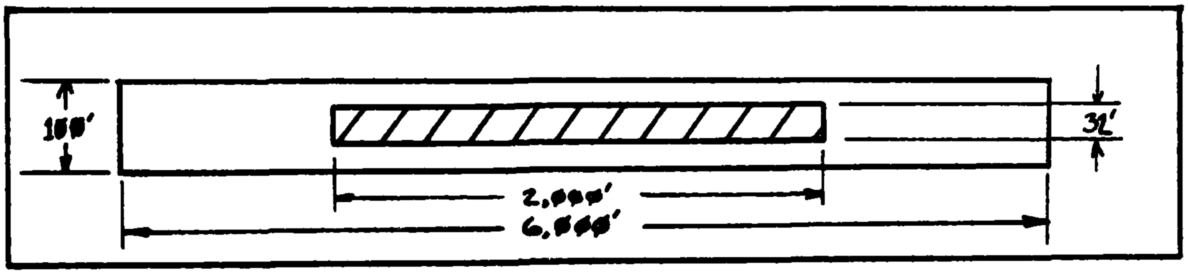


Figure 16 The Vulnerable Area of Runway-2

Main Effects: The main effect of accuracy initially surprised this analyst. As accuracy went up, Pc went down. But upon closer inspection, the result is fully plausible. The S/D of the two levels of accuracy were extreme: 30' and 370'. Therefore, when accuracy was set high, there was little chance of cutting Runway-2, separated by 300'. However, when accuracy was low, there was a good chance for damage to Runway-2.

Given that craters were almost 32' square, the original width of Runway-2 only 100', and the minimum width required only 50', closure came relatively easy. An impact anywhere beyond 2,000' from the Runway-2 ends, and beyond about 35' from the sides, will close the runway. This situation is depicted in Figure 16, although not drawn to scale.

The effect of accuracy on C2 can also be easily explained. Since aimpoints were at least 300' from Runway-2, when accuracy was high, few hits occurred on Runway-2.

The effect of density was also surprising, at least initially. But high density implied shorter stick length. Shorter stick length implied less chance of hitting Runway-2. And with less chance of closing Runway-2, the overall Pc of the field decreased.

The same reasoning holds for axis-of-attack. The lower axis-of-attack, the more craters on Runway-1 and

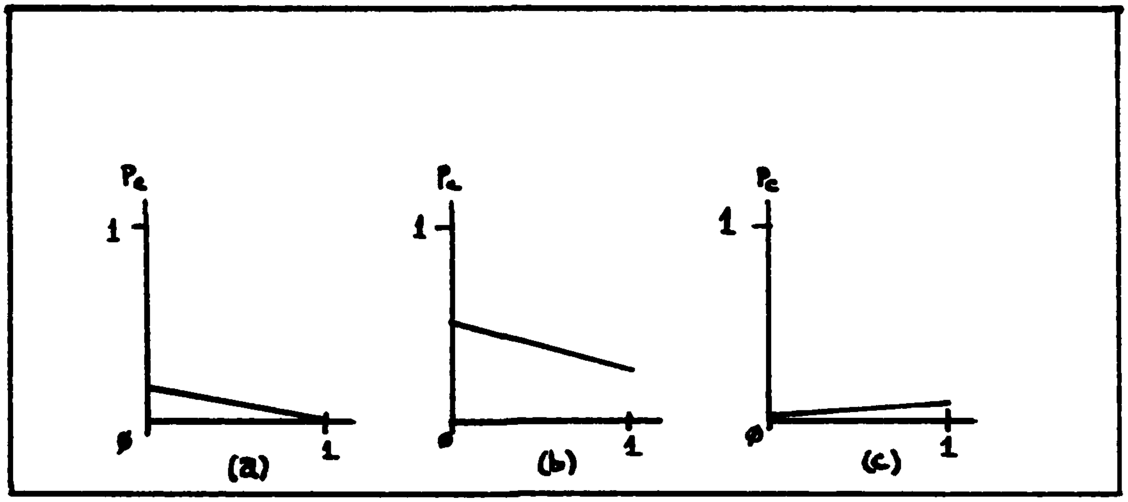


Figure 17 Plotted Effects on Pc by (a) Accuracy, (b) Density, and (c) Axis-of Attack.

therefore fewer on Runway-2, and thus less chance of closing Runway-2.

A summary of the main effects on each response appear in Tables V - VII, while Figure 17 graphs the three main effects on Pc.

Two-Way Interactions: Discussion of two-way interactions considers the combined effect of any two of the factors on the responses. Such study is one of the advantages of a factorial design for a statistical experiment. Each of the forty replications contributed information, that when processed, helped to detect the combined effects of two factors, as well as single factor effects.

The two way interactions in this experiment were not as dramatic as the single factor effects. Accuracy by axis produced the most obvious effects. Pc showed little interaction, but Pcr1, Pcr2, C1, C2, and H2 did display some factor interactions. Again, the interactions were weak, but they did exist. For example, at low angle-off, low accuracy exhibited a higher Pcr1 than did high accuracy.

Table V
Effect of Accuracy

	Level 0	Level 1
Pc	0.069	0.001
Pcr1	0.220	0.369
Pcr2	0.322	0.001
C1	1.663	1.275
C2	3.114	0.250
H1	26.175	53.775
H2	3.610	0.013

Table VI
Effect of Density

	Level 0	Level 1
Pc	0.044	0.027
Pcr1	0.354	0.235
Pcr2	0.174	0.149
C1	1.606	1.332
C2	1.730	1.634
H1	38.765	41.185
H2	1.837	1.786

Table VII
Effect of Axis-of-Attack

	Level 0	Level 1
Pc	0.007	0.063
Pcr1	0.061	0.528
Pcr2	0.082	0.241
C1	1.129	1.809
C2	1.812	1.552
H1	45.210	34.740
H2	1.215	2.408

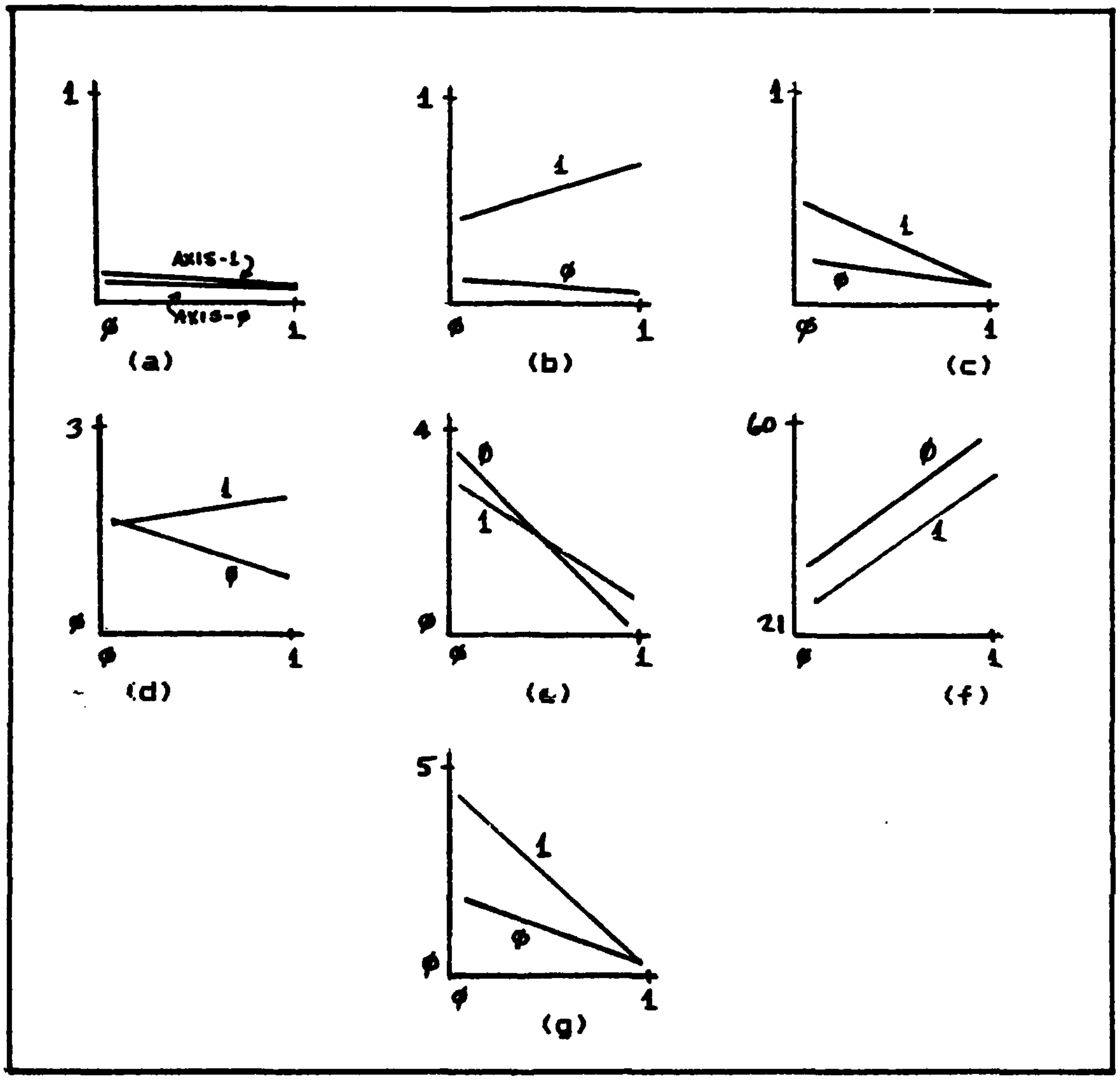


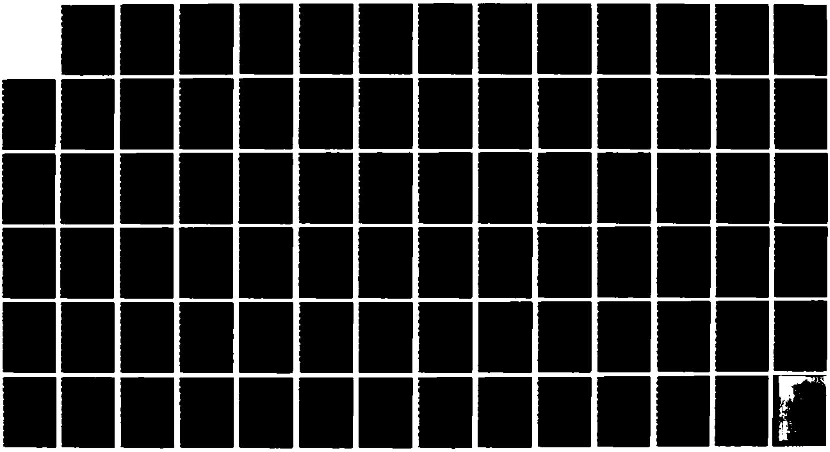
Figure 18 Plots of the Two-Way Interactions of Accuracy and Density for (a) Pc, (b) Pcr1, (c) Pcr2, (d) C1, (e) C2, (f) H1, (g) H2.

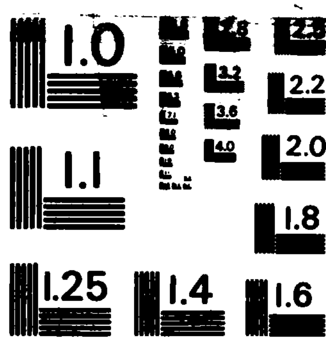
But at high angle-off, high accuracy produced better Pcr1. The interaction is easy to explain if one recalls that greater delivery error occurs in the range direction than in the deflection direction. With the low angle-off, this range error translated to errors along the runway. At high angle-off, most of the error was across the runway.

Recall the peculiar scenario of the attack. Aimpoint number two was the inside edge of Runway-1. When accuracy was high, and axis low, few weapons would fall low enough to deny 50 clear feet along the lower edge of the runway.

AD-A141 219 AAPMOD: AN INTERACTIVE COMPUTER MODEL FOR ANALYSIS OF CONVENTIONAL WEAPON..(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI.. R N MIGLIN 2/2

UNCLASSIFIED MAR 84 AFIT/GST/OS/84M-13 F/G 15/7 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

However, when axis-of-attack was high, high accuracy kept more impacts on the runway, and produced a better chance of closing it.

Three tables again present the effects of the two-way interactions. Table VIII has the combined effects of Accuracy and Density, Table IX has the combined effects of Accuracy and Axis-of-Attack, and Table X has the combined effects of Density and Axis-of-Attack. Figure 18 presents plots of the effects of Accuracy by Axis, which as described above showed the strongest two-way interaction.

Three-Way Interactions: Finally, the three-way interactions exhibited the least effect of all. Although most responses still had better than a 0.005 signifcance level, F-values were lower than for previous effects. The exception was H2. H2 was the number of collateral hits on Runway-2. Three-way effects had only a 0.5 significance level on this response. A plausible explanation follows.

While singly, accuracy and axis had significant effects on H2, density did not. Recall that accuracy levels were far enough apart to exhibit a clear effect on H2. Also, as angle-off moved high, there was a greater likelihood for impacts on Runway-2. But the difference in stick length due to density changes, alone, was not significant enough to affect H2.

Similarly, the two-way interactions were split: the two interactions with density had significance levels of only 0.6, while the the two-way between accuracy and axis was significant to 0.00001. The three-way interaction on H2 was therefore, not expected to show much significance. (Remember, H2 is a measure of collateral damage and was a collateral response.)

Table VIII

Effect of Accuracy by Density

	Accuracy —	Level Ø	Level 1
Pc	x Density — Ø x Density — 1	Ø.Ø86 Ø.Ø53	Ø.ØØ2 Ø.ØØØ
Pcr1	x Density — Ø x Density — 1	Ø.257 Ø.183	Ø.451 Ø.287
Pcr2	x Density — Ø x Density — 1	Ø.346 Ø.298	Ø.ØØ2 Ø.ØØØ
C1	x Density — Ø x Density — 1	1.645 1.681	1.566 Ø.983
C2	x Density — Ø x Density — 1	2.96Ø 3.267	Ø.5ØØ Ø.ØØØ
H1	x Density — Ø x Density — 1	25.97Ø 26.38Ø	51.56Ø 55.99Ø
H2	x Density — Ø x Density — 1	3.65Ø 3.57Ø	Ø.Ø23 Ø.ØØ2

Table IX

Effect of Accuracy by Axis

	Accuracy —	Level Ø	Level 1
Pc	x Axis — Ø x Axis — 1	Ø.Ø14 Ø.124	Ø.ØØØ Ø.ØØ2
Pcr1	x Axis — Ø x Axis — 1	Ø.118 Ø.322	Ø.ØØ4 Ø.734
Pcr2	x Axis — Ø x Axis — 1	Ø.164 Ø.48Ø	Ø.ØØØ Ø.ØØ2
C1	x Axis — Ø x Axis — 1	1.758 1.568	Ø.5ØØ 2.Ø49
C2	x Axis — Ø x Axis — 1	3.624 2.6Ø3	Ø.ØØØ Ø.5ØØ
H1	x Axis — Ø x Axis — 1	3Ø.92Ø 21.43Ø	59.5ØØ 48.Ø5Ø
H2	x Axis — Ø x Axis — 1	2.43Ø 4.79Ø	Ø.ØØØ Ø.Ø25

Table X
Effect of Density by Axis

	Density —	Level Ø	Level 1
Pc	x Axis — Ø x Axis — 1	Ø.ØØ7 Ø.Ø81	Ø.ØØ7 Ø.Ø46
Pcr1	x Axis — Ø x Axis — 1	Ø.Ø64 Ø.644	Ø.Ø58 Ø.412
Pcr2	x Axis — Ø x Axis — 1	Ø.Ø84 Ø.264	Ø.Ø8Ø Ø.218
C1	x Axis — Ø x Axis — 1	1.2Ø1 2.Ø1Ø	1.Ø57 1.6Ø7
C2	x Axis — Ø x Axis — 1	1.757 1.7Ø3	1.867 1.4ØØ
H1	x Axis — Ø x Axis — 1	45.1ØØ 32.43Ø	45.32Ø 37.Ø5Ø
H2	x Axis — Ø x Axis — 1	1.21Ø 2.463	1.22Ø 2.352

The above results point out the strong dependency of this experiment on the scenario. This analyst feels strongly that the 1ØØ' offset for aimpoint two skewed results from those initially expected.

Blocking Effect: The effect of the random number stream was also checked. Using an F test with 4,28 degrees of freedom (d.f.), the seed was not found to be significant to Pc or C2. However, blocking of the random number numbers was significant at better than a Ø.Ø1 level for Pcr1, C1, H1, and H2, and better than a Ø.Ø1Ø4 level for Pcr2. The F-test results for blocking can be found in Appendix F.

It is felt that the synchronization in the model caused the significance. There was no significance for Pc, the complex probability of closing both TOL surfaces. However, the same string of errors occurred to each aircraft, on each pass. The interaction between the two pavements disappeared when looking at Pcr1 and Pcr2. Thus the random numbers displayed greater significance in the response.

Sensitivity Analysis

Given the dependency of the results on the scenario, sensitivity analysis is crucial to fully understand the main effects as well as their interactions.

One of the primary factors to study is aimpoint. The attack consists of six aircraft. Apparently, it seems easy to close Runway-2. Even collateral damage from targeting Runway-1 closes Runway-2 with up to 25-35% probability. Rearranging aimpoints should improve the probability of closing the field.

This analyst expects significant interactions between aimpoint selection and accuracy. If accuracy is high, the weapons will be on the aimpoint, and achieve their goal. But if accuracy is low, collateral damage seems to yield as much damage as does the intended damage.

The reader is cautioned not to draw other conclusions from this experiment. 'The damage of six sorties seems high. However, the damage is due to 100% survivability of the aircraft, and 100% reliability of the weapon functioning. In fact, survivability will be less than 100%, and reliability can be as low as 15%.

VI. Project Summary

> Nothing will ever be attempted if all possible objections must be first overcome.
>
> Samuel Johnson

Summary

This thesis has contributed AAPMOD, a simple, fast, attack simulation model, to the number of tools available to operations planners. This model responds to the demonstrated need of Chapter I, to develop a method to accurately relate attack efforts to target damage. Whereas the parent program, AAP, is used by research and development agencies to produce better conventional weapons, AAPMOD can be used by aircrews and tactical planners to optimally employ the conventional weapons they have available to them today. Crews can use AAPMOD to optimally design attacks, and commanders can use AAPMOD to optimally assign weapon systems to targets.

Other works, as reported in Chapter II have made significant contributions to the study of conventional weapons effectiveness. AAPMOD draws on the best features of some of the previous works, and offers analysts a practical, flexible method to study weapons effects.

The system of interest, given the scope of this effort, has been tactical aviation attacking an airbase: specifically, the runway. Chapter III offered the reader a fundamental understanding of the interactions occurring in a modern, air-to-ground weapons delivery. Chapter III also related the various system inputs, through discussion of system interactions, to the primary response: probability of closure. Also, Chapter III discussed other responses such as number of impacts, or number of craters requiring repair before TOL capability is regained.

Chapter IV then presented a detailed discussion of AAPMOD. The Fortran source code is found in Appendix D. And while AAPMOD is the processor of the input variables, and generator of system responses, AAPIN helps the user quickly develop input data files for AAPMOD. AAPIN also reduces the size of AAPMOD by assuming some of the error trapping responsibility originally found in AAP. Use of AAPIN assures the user syntactically correct, and conceptually reasonable data input for AAPMOD. The source code for AAPIN is also available in the appendices, Section E.

Finally, although AAPMOD was verified throughout the conversion process, specific verification and validation occurred when a demonstrative, three factor, two level experiment was run. The results of the experiment are reported in Chapter V. The results suggest good credibility for AAPMOD.

Observations

The tactical experience of this analyst, in concert with the experience of this project's academic advisor, suggest that AAPMOD is an excellent contribution to the analysis techniques of assessing conventional munitions effectiveness.

The experiment reported in Chapter V, clearly demonstrates the statistical significance of some of the factors affecting conventional weapons effectiveness. However, different types of significance exist. And all types can influence the ultimate decisions. For example, personalities or politics may adjust values, so that although a given weapon system appears optimal after a rigorous analysis, some other system may be tasked for the mission. However, proper understanding of the results of AAPMOD may influence or counter personal prejudice or political concerns, so that weapons can, in fact, be optimally employed. Also, proper study with AAPMOD may provide the education necessary to eliminate *innocent* misconceptions, that nevertheless detract from optimal weapons employment.

AAPMOD does seem to possess the capability to educate, as well as assist in analyses. It is interactive and transportable. Perhaps, if aircrews and planners were to run AAPMOD often enough, they may develop a feel for planning an attack, and intuitively optimize the factors that contribute to attack success. Recall from Chapter III, the complex interactions affecting the probability of cutting a runway, or denying TOL operations from a base. A rigorous analysis of these tasks, requires a math and statistics background. Such a foundation is not always available in the educational background of aircrews or decision makers. The experience of these people rests in flying aircraft, and delivering the weapons under consideration. Therefore, AAPMOD, with its technique of simulation, offers these "educated laymen" the information and the methodology to relate their experience and training to an analysis of weapons delivery.

Recommendations

Recommendation-1: During conversion of AAP to AAPMOD, a type of target entry procedure was eliminated. The option allowed coordinate entry of the centers of opposite ends, and the element width. When AAPIN was developed, this option was eliminated as a superfluous luxury. However, entering the series of ten pavements defined in the sample experiment, suggests reinserting the option to AAPIN. Once the complex is drawn on graph paper, locating end points and defining widths can be easier than finding true center points, lengths, and angular orientations.

Recommendation-2: AAPIN should be prettied to further enhance useability. As an example, an algorithm developed by this analyst for an earlier project will input airspeed, release mode, release interval, dive angle, and several additional variables not addressed in this study, and output coordinates of the impacts in a stick delivery. This algorithm should be added to AAPIN to facilitate pattern descriptions.

Recommendation-3: Data input to AAPIN will be easier if accompanied by a series of worksheets. The user can study the target complex, complete the worksheets, and quickly enter the data to AAPIN.

Recommendation-4: The WRITE statements of AAPIN, and their associated FORMAT's should be reviewed and modified to prevent roundoff errors.

Recommendation-5: Change output of AAPMOD to reflect expected number of craters closing the easiest clear strip to repair, given the runway is closed. (See discussion in Chapter IV.)

Recommendation-6: Further reduce the loader requirements of AAPMOD to fit microcomputer RAM capability. Currently, AAPMOD requires about 17,000 words on the 60-bit CYBER. Noting that many of the values of AAPMOD are integers, the further conversion of AAPMOD to microcomputer Fortran is possible.

To demonstrate, this analyst compiled PROGRAM--MAIN, and generated an execution program for MAIN on his IBM Personal Computer, containing 256KB RAM. The binary execution program was only 57KB. However due to the large COMMON requirement, almost 8,000 words, or approximately 251KB, the loader could not properly function. Nevertheless, microcomputer implemen- tation seems reasonable. The PC uses 16-bit words, doubling the size for integers and reals. This produces the equiv- alent of a 32-bit machine. (Roundoff error could conceivably affect results, but given the algorithms of AAPMOD, and the low significance required of most variables, such error is expected to be negligible.) Converting 17,000 words to an average 30-32 bits each, requires a RAM of a little over a half-million bits or 544KB. In today's market, such capacity is well under $10,000.00, and closer to $5,000.00.

The Tactical Air Command has purchased 16-bit micro- computers, and USAFE purchased some high capability, 8-bit Cromenco microcomputers. Recommendation 2 is to investigate the feasibility of placing AAPMOD onto these small computers and further disseminate its planning utility.

Recommendation-7: The demonstration experiment of the previous chapter retained synchronization of the random numbers used in the simulation. There is, however, the potential to lose synchronization when reliabilities or survivabilities fail. (See disscussion in Chapter IV.) Efforts should be directed to enable AAPMOD to use separate random number streams to control accuracies and reliabilities.

Recommendation-8: Given the work of Hachida and Pemberton, combined with the capability afforded by AAPMOD, it may become possible to actually plan an airfield attack to maximize probability of TOL denial. Such a monumental planning tool would require a full factor screening of system inputs, to determine those with the most significance. Then loops can be placed in AAPMOD to change the factors, assess damage, and ultimately maximize results.

Recommendation-9: Possibly an alternative to heuristically looping AAPMOD, would be to use the techniques of response surface methodology (RSM). Since most input variables are continuous, quantitative variables, RSM seems a promising approach to optimize the damage resulting from an attack plan. (Ref. 19: 170)

Recommendation-10: AAPMOD should be studied to determine its suitability for assessing blast and fragment damage to structures, vehicles, or people. Application of AAPMOD to such types of analysis can then run the gamut of weapon effectiveness studies. One model may possibly take the place of two or three specific weapon analysis programs.

Recommendation-11: A study should determine the significance of the use of the bi-variate, *rectangular* normal error distributions, over bi-variate, *elliptical* normal error distribution. Minor error can be introduced into the analysis if the rectangular normal is used when in reality the true distribution more closely resembles an ellipse.

Recommendation-12: Although potential enemies possess hundreds of useable airfields, the number of high-value fields is limited. Reported in the November '82 issue of *Armed Forces Journal*, there are only about 72 Main Operating Bases within 800 km of the inner German border. Although preplanning attacks on all these fields commends a significant effort, to do so may equally improve mission success.

By preplanning is meant the development of a full attack, optimized for maximum damage to the airfield. Such optimization will include consideration of defenses, navigation accuracies, and collateral damage to adjacent target elements. One last mention: such optimization also requires utility or value assignments to target elements, relative to their contribution in support of combat sorties. While AAPMOD cannot emulate the entire decision process, AAPMOD can contribute the expected damage to the complex, due to the attack.

It is felt that AAPMOD satisfies the objective of providing a clear methodology to relate attack effort to target damage. The preceeding recommendations serve to further enhance the utility of AAPMOD.

Throughout this study, an implicit objective was to pursue research that had more than an academic significance. It is felt that exercising AAPMOD will positively affect the future effectiveness of tactical aviation. This improvement will reveal the ultimate, practical significance of this thesis.

Bibliography

1. ------. Attack Assessment Computer Program (AAP), Volume 1--User's Manual. 61JTCG/ME-80-3-1. Joint Technical Coordinating Group for Munitions Effectiveness, 1 Jun 80.

2. Bexfield, James N. Class Lecture, MA 4.41, Air Force Institute of Technology, Wright-Patterson AFB, OH, Winter, 1983.

3. Brodie, Bernard and Fawn M. From Crossbow to H-Bomb. Bloomington, IN: Indiana University Press, 1973.

4. David, H.A. Order Statistics. New York: John Wiley and Sons, Inc., 1970

5. Goodson, Winfred L., B/Gen, USAFE/XO, Consultation with, Guest Lecturer, Operational Sciences Department, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 21 April 83.

6. "Classified Document: Qualified requestors may obtain this reference from AFIT/ENS, Wright-Patterson AFB OH 45433."

7. Hachida, Howard M. A Computer Model to Aid the Planning of Runway Attacks. Unpublished. MS Thesis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 82.

8. Hicks, Charles R. Fundamental Concepts in the Design of Experiments. New York, NY: Holt, Rinehart and Winston, 1982.

9. Hillier, Frederick S. and Gerald J. Lieberman Introduction to Operations Research. San Francisco, CA: Holden-Day, Inc., 1980.

10. Hoeber, Francis P. Military Applications of Modeling. Ex Libris Maj J. Coakley. Dept. of Operational Sciences, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH.

11. "Classified Document: Qualified requestors may obtain this reference from AFIT/ENS, Wright-Patterson AFB OH 45433."

12. Law, Averill M. and W. David Kelton. Simulation Modeling and Analysis. New York, NY: McGraw-Hill Book Company, 1982.

13. Lewis, T.G. and B.J. Smith Computer Principles of Modeling and Simulation. Boston, MA: Houghton Mifflin Company, 1979.

14. Meyer, Deborah G. and Benjamin F. Schemmer. Interview with General Wilbur L. Creech. Armed Forces Journal. January, 1983.

15. -----. NATO Study of the Balance of Military Power. Belgium: Supreme Headquarters Allied Powers, Europe. May, 1982.

16. Pemberton, John C. A Generalized Computer Model for the Targeting of Conventional Weapons to Destroy a Runway. Unpublished. MS Thesis. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 80.

17. -----. Report to the Congress of the United States--Models, Data, and War: A Critique of the Foundation for Defense Analyses. Office of the Comptroller General, 12 March 80.

18. Schemmer, Benjamin F. "NATO's New Strategy: Defend Forward, But Strike Deep". Armed Forces Journal. November, 1983.

19. Shannon, Robert E. Systems Simulation: The Art and Science. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1975.

20. Wikner, Dr. N.F. (Fred). "Interdicting Fixed Targets with Conventional Weapons". Armed Forces Journal. March, 1983.

Vita

Robert N. Miglin was born on January 11, 1953. He grew up in Sayreville, NJ, attending Our Lady of Victories R.C. Grammer School, and Sayreville War Memorial High
..footc5898
School. He was the high school valedictorian in June, 1971, and entered the United States Air Force Academy in July. In 1975, he proudly accepted a regular commission as a Second Lieutenant, USAF, and received a Bachelor of Science degree in Civil Engineering.

Captain Miglin has extensive air to ground weapons delivery experience, as well as experience in high-speed, low-level aerial navigation. He attended navigator training at Mather AFB, CA, and earned his nav wings in 1976. His first assignment was to FB-111A's: SAC's medium range, strategic bomber. This was followed with an assignment to the F-111E's of USAFE, at RAF Upper Heyford, England. His combined time in F/FB-111's is over 1200 hours.

In May, 1981, Captain Miglin attended an advanced tactics school, sponsored by the Allied Air Forces Central Europe. The school was the month-long, Tactical Leadership Program at Jever AB, Germany. There, he flew 16 sorties as part of a NATO, multi-national training exercise, and attended 3 weeks of academics on alliance operations.

In August, 1982, he entered the Graduate Program, Strategic and Tactical Sciences, Department of Operational Sciences, School of Engineering, Air Force Institute of Technology, Wright Patterson AFB, OH. Graduation is 16 March 1984. His directed-duty assignment is to the office of the Deputy Chief of Staff for Armament and Avionics, Tactical Air Warfare Center, Eglin AFB, FL. AV 872-3935.

Permanent Address: 124 Dolan Street
Sayreville, NJ 08872

Eglin Address: 210 Dominica Circle, East
Niceville, FL 32578

Appendix A

GLOSSARY OF FREQUENTLY USED TERMS

Like any field, tactical aviation has its own lingo. Presented here are definitions of some of the terms used in this thesis.

Aimpoint--the point on the ground where the pilot desires his weapons to impact.

Area Weapon--typically a CBU. By design, the numerous submunitions within a single dispenser will cause many small craters over a large area, called the weapon footprint.

B--suffix to number to indicate an octal value, frequently used when discussing computer hardware requirements.

Bomb--generic for a unitary weapon, released from an aircraft, and that falls without additional propulsion.

Cluster Bomb Unit (CBU)--a single dispenser, released from an aircaraft. The dispenser opens prior to ground impact, and releases numerous submunitions.

DMPI--desired mean point of impact. Aimpoint of an attack pass, when several weapons are released: the desired mass center of the "stick" of weapons.

DPI--desired point of impact. Aimpoint of an attack pass when only one weapon is released.

Footprint--the ground coverage of uniformly distributed impacts from a cluster weapon.

Minimum Clear Length (MCL)--when considering a take-off and land capable surface (asphalt, concrete, or even sod), the minimum distance, of sufficient width, to enable aircraft operations from the surface.

Minimum Clear Width (MCW)--similar to MCL, but refers to width. MCW is typically wider than taxi-width, due to less precision of the high-speed conditions.

Mission--the total effort expended to damage a target complex. The "mission" can include several attack phases, each composed of several attack passes, each composed of the release of one or more weapons.

MPI- mean point of impact. The actual mass center of the impacts resulting from a stick release.

Multiple Release--more than one weapon released on a single pass over the target. A multiple release results in a "stick" of weapon impacts.

Point-Impact Weapon--unitary weapon such as general purpose (GP) bombs or precision guided munitions (PGM).

Stick--a ground pattern of craters, resulting from a multiple release delivery pass. The pattern is defined by release conditions discussed in Chapter III.

Submunitions--small warheads packed into a dispenser. Typically, a CBU is considered as one *weapon* that contains numerous submunitions. Each submunition is capable of a limited size crater, but due to the numbers of craters, damage is distributed over a large area.

Taxi-Width--minimum width of surface required to permit aircraft taxi operations. Usually a function of gear width. Implies slow speed and, possibly, ground marshallers.

Unitary Weapon--a weapon that contains only one cratering device.

Void Area--the area within a CBU footprint that may be void of impacts due to dispenser functioning or design.

Warhead--the part of a weapon that causes a crater upon impact.

Weapon--a generic term for a bomb, CBU, missile, or rocket, whole and complete of itself.

Weapons Delivery Pass--a flight maneuver involving the release of a weapon or weapons from an aircraft, in attempt to damage a target.

Appendix B

Discussion of Ballistic CEP

A reasonable value for the intrinsic ballistic errors in a high-drag bomb is about 5 mils. The error is due to minor differences in fin alignment, CG location, bomb-rack ejection velocity, ejection yaw angle, ejection angle-of-attack, and several other factors. (A "mil" is one-milliradian or 0.001 radians, a dimensionless measure of an angle.)

Also, for say a 1,500' level release, a high-drag bomb has a ground range of about 3,730', and a slant range of 4,020' Figure B.1 depicts the geometry of the situation.

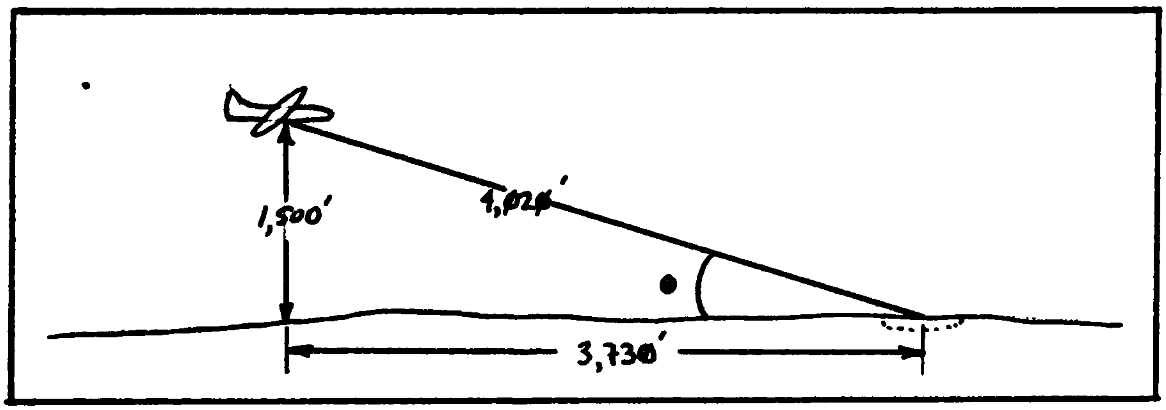


Figure B.1 Geometry of a Weapons Release.

Given so many factors contributing to ballistic dispersion of the weapon, a circular normal error distribution is a reasonable choice to characterize the error. Therefore, the plan view of the release of a single weapon can be depicted by Figure B.2. The aircraft

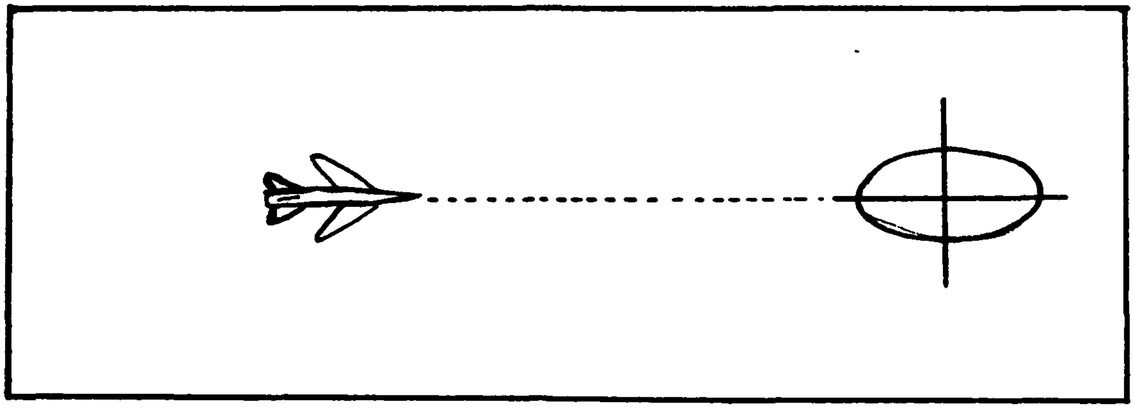


Figure B.2 Plan View of a Weapons Release.

releases the weapon, and as it drops, it continues forward, with its bomb-range, and impacts with an error drawn from the normal, ballistic error distribution. On the ground, the error relates to an ellipse.

The deflection component (the component of the error transverse to the flight direction of the aircraft) translates to distance simply with:

DISTANCE = THETA * SLANT RANGE

where slant range is as described above, and theta is the angular displacement.

So in this example:

DISTANCE = 0.005 * 4020' = 20.1'

But since the 20' is error probable, it must be converted to a normal distribution's standard deviation (S/D). Again, refer to Hachida or Pemberton for the derivation, but the conversion for range or deflection is simply:

0.675 S/D = Error Probable

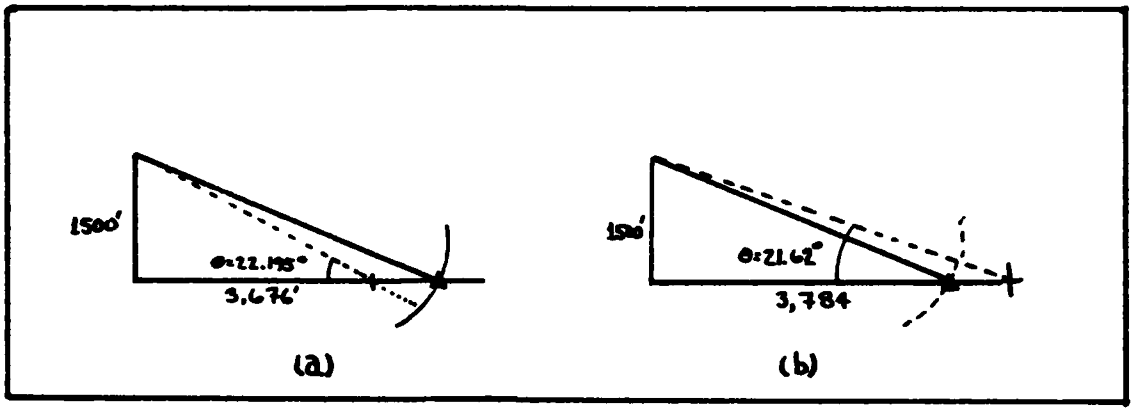


Figure B.3 Ballistic Dispersion Error Applied to Release.

The elliptical pattern is due to the slant between the weapons trajectory and the ground. Circular error in the plane normal to the trajectory of the weapon is a circle. But the same error in a plane perhaps 68.1 off the normal, will be an ellipse. Using the same example, Figure B.3 (a) depicts the case where the 5 mils add extra depression to the trajectory, and produce a short impact, and (b) depicts the case where the ballistic dispersion reduces the depression, and the bomb goes long. Note the differences in ground range.

It is error in the ground plane that misses the target, not errors in some hypothetical plane normal to the weapon trajectory. But again, converting 54' to a S/D yield a sigma of 80'. Thus, this Appendix has briefly shown how the 80' REP, and 30' DEP, used in the demonstration experiment, were chosen.

Appendix C

Program Variable List

NSAMPT: Write to output every NSAMPT

NSAMP: total # Monte Carlo iterations

TIME: estimated CP time (TXXX on job control card)

LUNITO: Input/output option

NFLAG3: = Ø: nothing
= 1: and NSAMP > 2ØØ, will reduce sample size, if appropriate

ZALPH: Normal Z for one-half required confidence

ERROR: Tolerable difference in probability between sample and true

NELT: # targets (max 112)

NTGPS: # target groups (max 15)

APPRCW: min taxi width; also flag: = Ø.Ø, then suppresses search for taxi approach to clear strip

NAREA: Flag to compute damage area of TOL
= Ø: Yes; = 1: No (skips OVLAP)

TGT(I,1): X-coord, center of I target
(I,2): Y-coord, center of I target
(I,3): orientation angle, degrees
(I,4): length
(I,5): width

ITGT(I,1): target type code = 1: surface
= Ø: not surface
(I,2): surface code for crater radius table (max 11)
(I,3): target group, with which I is associated)

CRIT(I,1): Flags type of surface capability
(Min Clear Length for TOL)
MCL=0.0 indicates taxi only (w/meandering course)
MCL=length indicates length of clear strip required to permit TOL

(I,2): Min Clear Width required...
for TOL if MCL = 0.0
for taxi if MCL = 0.0

NPATT: number of weapon patterns (max 12)

M: # different surface hardnesses (max 11)

N: # of warhead codes, (max of 6)

NSQCR: Crater Code = 0: craters input square
= 0: craters input round, converted to eqv't sq area

CRTAB(I,J,K): the crater-size storage array where subscript
I = surface type 1-11
J = weapon type 1-6
K = type of encounter
=1: for Bldgs--near miss size
for Pavement--TOL crater size
=2: for Bldgs--direct hit crater size
for Pavement--taxi crater size

NATT: # of attacks (max 10)

MXPTCH: # patches resources will allow

IREPR: airbase established priority for repair of craters (see program list, Appendix E)

NPATCH: # of patches time will allow after attack

PASS(I,1): X-coordinate of aimpoint, pass number I
(I,2): Y-coordinate of aimpoint
(I,3): axis-of-attack
(I,4): Pr a/c reaches target
(I,5): Pr a/c can reattack

IPASS(I,1): Pattern # from PATT array
(I,2): Next pass number that this a/c is responsible for.

Variable Name	for General Purpose:	when Cluster:	when a Guided Munition:
IPAT(I,1)	# weapons in I patt (max 12)	same	same
(I,2)	=1 General Purpose Bombs	= # bomblets/dispenser	= 1 Guided Bomb
(I,3)	Weapon Code (Crater Tab Index)	same	same
(I,4)	Not Used (N/U)	PattShape:1=Rect,2=Elps,3=N/A	PattShape 3=guided bomb
PATT(I,1)	S/D aimpt--range	same	CEP1 converted to S/D rng
(I,2)	S/D aimpt--deflection	same	CEP1 converted to S/D def
(I,3)	S/D Bd--range	same	CEP2 converted to S/D rng
(I,4)	S/D Bd--deflection	same	CEP2 converted to S/D def
(I,5)	N/U	1/2 pattern length, range	range for GE
(I,6)	N/U	1/2 pattern width, deflection	deflection for GE
(I,7)	N/U	1/2 void length, range	Pr (CEP1)
(I,8)	N/U	1/2 void width, deflection	Pr (CEP1 or CEP2)
(I,9)	fuze reliability	disp fuze reliability	fuze reliability
(I,10)	N/U	bomblet fuze reliability	N/U
(I,J+10)	range of 1st weapon from aimpt	same	
(I,J+11)	deflection of 1st weapon	same	
(I,J+12)	range of 2nd weapon	same (stick)	
(I,J+13)	deflection of 2nd weapon	same (definition)	
(I,J+14)	range of 3rd weapon	same	
(I,J+....)	ect.	same	

Appendix D

```
*************************************************************************
* LAST UPDATE 01/1200 MAR 84          FILE:INPUT.AAP
*************************************************************************
      PROGRAM AAPIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
*
      REAL TGT(112,5),CRIT(112,2),PATT(12,34),CRTAB(11,6,2),PASS(32,6)
      INTEGER ITGT(112,3),IPAT(12,4),IPASS(32,2),OPT(32,2)
      CHARACTER FNAME*6,YESNO*1
*
      PRINT*
      PRINT*
      PRINT 900
900   FORMAT(
     11X,'THIS PROGRAM WILL CREATE A LAUNDERED INPUT TAPE FOR THE',/,
     21X,'MODIFIED ATTACK ASSESSMENT PROGRAM--AAPMOD.  YOUR OPTIONS',/,
     31X,'ARE TWO:',/,
     41X,'          0:  CREATE A NEW TAPE',/,
     51X,'          1:  MODIFY OR CHECK AN EXISTING TAPE')
      PRINT*,'ENTER CHOICE, 0 OR 1==> '
      READ*,ICHC
      IF (ICHC.EQ.0) THEN
        PRINT*
        PRINT*
        PRINT*,'BE SURE TO CHOOSE A NEW, UNUSED FILE NAME...'
        PRINT*,'ENTER FILE NAME FOR THIS TAPE (MAX 6 CHARACTERS)==> '
        READ 935,FNAME
        OPEN(UNIT=12,FILE=FNAME,STATUS='NEW')
        PRINT*
        PRINT*
        PRINT*,'ENTER PROGRAM CONTROLS:'
1       PRINT*,'SEED (MAX 10 DIGIT INTEGER)==> '
        READ*,ISEED
        IF ((ISEED.LT.1).OR.(ISEED.GT.9999999999)) GO TO 1
        PRINT*,'NUMBER OF MONTE CARLO ITERATIONS==> '
        READ*,NSAMP
        ERROR=0.05
        ZALPH=1.645
        NFLAG3=0
        IF (NSAMP.GT.200) THEN
          PRINT*,'DO YOU WANT TO REDUCE SAMPLE SIZE, IF PRACTICAL?'
          PRINT*,'Y/N==> '
          READ 934,YESNO
          IF (YESNO.EQ.'Y') THEN
            NFLAG3=1
            PRINT*,'DEFAULT IS 0.05 ERROR FROM TRUE PROBABILITY.'
```

```
      PRINT*,'WITH A TGT HARDNESS-CODE, DETERMINES THE CRATER SIZE.'
      PRINT*,'NOTE...  TWO DISPENSERS WITH THE SAME SUBMUNITION'
      PRINT*,'BUT IN DIFFERENT NUMBERS OF SUBMUNITIONS, DOES NOT,'
      PRINT*,'REPEAT, DOES NOT IMPLY DIFFERENT WEAPONS AT THIS POINT.'
6     PRINT*,'(MAX 6)==> '
      READ*,NWPN
      IF (NWPN.GT.6) GO TO 6
      PRINT*,'**** READ CAREFULLY ****'
      FRINT*
      PRINT*,'FOR ANY COMBINATION OF SURFACE TYPE AND WEAPON TYPE,'
      PRINT*,'AAPMOD USES TWO DIFFERENT CRATER SIZES.  THE SIZE USED'
      FRINT*,'DEPENDS ON WHETHER THE IMPACT WAS AGAINST A PAVEMENT,'
      PRINT*,'OR A NON-PAVEMENT (A BUILDING).'
      FRINT*
      PRINT*,'IF THE TARGET TYPE-CODE WAS ASSIGNED TO A PAVEMENT,'
      PRINT*,'FIRST, ENTER THE SIZE OF THE DISRUPTION SEVERE ENOUGH'
      PRINT*,'TO DENY HI-SPEED TOL OPERATIONS, AND THEN THE SIZE OF'
      PRINT*,'DISRUPTION SEVERE ENOUGH TO DENY TAXI OPERATIONS.'
      PRINT*
      PRINT*,'IF THE TARGET WAS A BUILDING, FIRST ENTER THE CRATER '
      PRINT*,'RADIUS RESULTING FROM A NEAR-MISS, THEN THE RADIUS'
      FRINT*,'RESULTING FROM A DIRECT HIT.'
      PRINT*
      PRINT*,'ALSO, AAFMOD USES SQUARE CRATERS.  CHOOSE ENTRY MODE:'
      PRINT*,'     0:  INPUT AS HALF-LENGTH OF SIDE OF SQUARE CRATER'
      FRINT*,'     1:  INPUT AS RADIUS OF CIRCULAR CRATER'
      PRINT*,'CHOICE==> '
      READ*,NSQR
      PRINT*,'THIS TAPE PROGRAM WILL LOOF SLOWEST ON INTERACTIONS.'
      PRINT*,'THEN HARDNESS TYPES, AND FASTEST ON WARHEAD TYPE.'
      FRINT*,'ENTER CRATER SIZES AS INSTRUCTED:'
      PRINT*,'GOOD LUCK...'
      FRINT*
      IF (NSQR.EQ.0) THEN
        PRINT*,'USE 1/2 THE LENGTH OF A SIDE...'
        PRINT*
      ELSE
        PRINT*,'USE THE RADIUS OF A CIRCULAR CRATER...'
        PRINT*
      ENDIF
      DO 301 I=1,NSFC
       DO 301 J=1,NWPN
        IF (I.LE.NTP) THEN
         PRINT*,'SFC TYPE ',I,', WPN TYPE ',J,' DENY-TOL SIZE==> '
        ELSE
         PRINT*,'BLDG TYPE ',I,', WPN TYPE ',J,' NEAR-MISS SIZE==> '
        ENDIF
        READ*,CRTAB(I,J,1)
301   CONTINUE
      PRINT*
      PRINT*
```

```
      IF (NSQR.EQ.0) THEN
        PRINT*,'USE 1/2 THE LENGTH OF A SIDE...'
        PRINT*
      ELSE
        PRINT*,'USE THE RADIUS OF A CIRCULAR CRATER...'
        PRINT*
      ENDIF
      DO 302 I=1,NSFC
       DO 302 J=1,NWPN
        IF (I.LE.NTP) THEN
         PRINT*,'SFC TYPE ',I,', WPN TYPE ',J,' DENY-TAXI SIZE==> '
        ELSE
         PRINT*,'BLDG TYPE ',I,', WPN TYPE ',J,' DIRECT-HIT SIZE==> '
        ENDIF
        READ*,CRTAB(I,J,2)
302   CONTINUE
      IF (NSQR.EQ.1) THEN
        DO 310 I=1,NSFC
         DO 310 J=1,NWPN
          DO 310 K=1,2
           CRTAB(I,J,K)=CRTAB(I,J,K)*0.886
310     CONTINUE
      ENDIF
*
      PRINT*,'DESCRIBE THE TARGET COMPLEX:'
2     PRINT*,'NUMBER OF TARGET ELEMENTS (MAX 112)==> '
      READ*,NELT
      IF ((NELT.LT.1).OR.(NELT.GT.112)) GO TO 2
3     PRINT*,'NUMBER OF TARGET GROUPS (MAX 15)==> '
      READ*,NTGPS
      IF ((NTGPS.LT.1).OR.(NTGPS.GT.15)) GO TO 3
      PRINT*,'MIN WIDTH FOR TAXI OPS==> '
      READ*,APPRCW
4     PRINT*,'NUMBER OF TOL CAPABLE SURFACES (MAX 3)==> '
      READ*,NCP
      IF (NCP.GT.3) GO TO 4
      PRINT*,'NUMBER OF PAVEMENTS FOR TAXI ONLY (MAX ',30-NCP,')==> '
      READ*,LV
      IF ((LV+NCP).GT.30) THEN
        PRINT*,'TOO MANY SURFACES, MAX IS 30.'
        GO TO 4
      ENDIF
      PRINT*,'NUMBER OF NON-PAVEMENTS (MAX ',NELT-LV-NCP,')==> '
      READ*,NBLDG
      IF ((LV+NCP+NBLDG).GT.112) THEN
        PRINT*,'TOO MANY TARGETS, MAX IS 112.'
        GO TO 4
      ENDIF
      PRINT 905
905   FORMAT(1X,//,
     1 1X,'YOU MUST NOW DEFINE EACH TARGET.',/,
```

```
     2 1X,'ESTABLISH AN X-Y COORDINATE SYSTEM FOR THE COMPLEX.',/,
     3 1X,'THE POSITIVE X-AXIS BECOMES 0 DEGREES ANGLE-OFF.',/,
     4 1X,'RECOMMEND...  MAIN RUNWAY CENTER AND ORIENTATION DEFINE',/,
     5 1X,'THE COORDINATE SYSTEM.  LATER, ATTACK PASSES ALSO',/,
     6 1X,'HAVE AIMPOINTS AND ANGLE-OFF DEFINED BY SAME SYSTEM.')
       PRINT 910
910    FORMAT(1X,/,
     1 1X,'INPUT PROMPTS ARE DEFINED AS FOLLOWS:',//,
     2 1X,'X-COORD:  X-COORDINATE OF THE CENTER OF THE TARGET,',/,
     3 1X,'          REFERENCED TO COORDINATE SYSTEM OF COMPLEX.',/,
     4 1X,'Y-COORD:  TYPICAL TO X-COORD.',/,
     5 1X,'   AXIS:  ORIENTATION OF TARGET CENTERLINE, MEASURED CCW',/,
     6 1X,'          FROM +X-AXIS OF COORD SYSTEM OF THE COMPLEX.',/,
     7 1X,' LENGTH:  BOTH LENGTH AND WIDTH ARE SELF-EXPLANATORY...,',/,
     8 1X,'  WIDTH:                AND MAY BE CHOSEN ARBITRARILY.',/,
     9 1X,'TYPCODE:  TYPE OF TARGET    1 = PAVEMENT (TAXI OR TOL)',/,
     + 1X,'                            0 = NON-PAVEMENT (BUILDING)',/,
     1 1X,'SFCCODE:  HARDNESS OF TARGET   (FOR CRATER TABLE LOOKUP)',/,
     2 1X,' TGTGRP:  TARGET GROUP THE TARGET BELONGS TO',//,
     3 1X,'       ** ENTER  0  TO CONTINUE **')
       READ*,WAIT
       PRINT 915
915    FORMAT(1X,/,
     1 1X,'TWO OTHER PROMPTS:',/,
     2 1X,'  MINCL:  MINIMUM CLEAR LENGTH REQUIRED FOR TOL OPS',/,
     3 1X,'          ENTER ZERO FOR TAXI-ONLY PAVEMENTS.',/,
     4 1X,'  MINCW:  MINIMUM CLEAR WIDTH REQUIRED FOR...',/,
     5 1X,'            TAXI:  IF MINCL = 0.0   (TAXI-ONLY)',/,
     6 1X,'            TAKE-OFF/LAND:  IF MINCL .NE. 0.0.',//,
     7 1X,'ENTER PAVEMENTS IN ORDER OF PRIORITIZED IMPORTANCE,',/,
     8 1X,'MOST IMPORTANT, FIRST.',///)
*
       NTOL=0
       NPAV=0
*
       DO 100 I=1,NELT
104     CRIT(I,1)=0.
        CRIT(I,2)=0.
        PRINT*,'FOR TARGET NUMBER ',I,', ENTER:'
        PRINT*,'X-COORD==> '
        READ*,TGT(I,1)
        PRINT*,'Y-COORD==> '
        READ*,TGT(I,2)
        PRINT*,'   AXIS==> '
        READ*,TGT(I,3)
        IF (TGT(I,3).GE.180.0) TGT(I,3)=TGT(I,3)-180.
        TGT(I,3)=TGT(I,3)*0.01745
        PRINT*,' LENGTH==> '
        READ*,TGT(I,4)
        PRINT*,'  WIDTH==> '
        READ*,TGT(I,5)
```

```
      IF ((I.LT.(NCP+LV)).AND.(TGT(I,5).GT.899.0)) THEN
      PRINT*,'CODE TO ACCUMULATE TOTAL PAVEMENT AREA DAMAGED IS'
      PRINT*,'DESELECTED, SINCE THIS TARGET IS TOO WIDE FOR ROUTINE.'
      NAREA=1
      ENDIF
101   PRINT*,'CHOICE OF TYPES:  0 = BLDG  /  1 = PAVEMENT'
      PRINT*,'TYPCODE==> '
      READ*,ITGT(I,1)
      IF ((ITGT(I,1).NE.0).AND.(ITGT(I,1).NE.1)) GO TO 101
102   PRINT*,'SFCCODE==> '
      READ*,ITGT(I,2)
      IF (ITGT(I,2).GT.11) THEN
        PRINT*,'TOO MANY.  MAX IS 11.  RE-ENTER.'
        GO TO 102
      ENDIF
      IF (((ITGT(I,1).EQ.1).AND.(ITGT(I,2).GT.NTP)).OR.
     1    ((ITGT(I,1).EQ.0).AND.(ITGT(I,2).LE.NTP))) THEN
        PRINT*,'*** MISMATCH WITH SFC CODE AND TGT TYPE ***'
        PRINT*,'  IF UNRECONCILABLE AT THIS INPUT POINT,'
        PRINT*,'  YOU MUST TERMINATE PROGRAM WITH <%A>,'
        PRINT*,'  AND RESTART WHEN YOU CLEAN UP THE ERROR.'
        PRINT*
        GO TO 102
      ENDIF
103   PRINT*,' TGTGRP==> '
      READ*,ITGT(I,3)
      IF (ITGT(I,3).GT.15) THEN
        PRINT*,'TOO MANY.  MAX IS 15.  RE-ENTER.'
        GO TO 103
      ENDIF
      IF (ITGT(I,1).EQ.1) THEN
        NPAV=NPAV+1
        IF (NPAV.GT.NCP+LV) THEN
          PRINT*,'NUMBER OF PAVEMENTS EXCEEDS ',NCP+LV,'.  TOO MANY.'
          NPAV=NPAV-1
          GO TO 104
        ENDIF
        PRINT*,'TARGET ',I,' IS A PAVEMENT.  ENTER MINCL FOR TOL.'
        PRINT*,'(0 IMPLIES TAXI ONLY)  MINCL==> '
        READ*,CRIT(I,1)
        IF (CRIT(I,1).LT.1.0) THEN
          PRINT*,'PAVEMENT ',I,' IS FOR TAXI ONLY.'
          PRINT*,'ENTER MINCW FOR TAXI OPS==> '
        ELSE
          NTOL=NTOL+1
          IF (NTOL.GT.NCP) THEN
            PRINT*,'NUMBER OF TOL SFCS EXCEEDS ',NCP,'.  TOO MANY.'
            NPAV=NPAV-1
            NTOL=NTOL-1
            GO TO 104
          ENDIF
```

```
            PRINT*,'PAVEMENT ',I,' SUPPORTS TOL OPS.'
            PRINT*,'ENTER MINCW FOR TOL OPS==> '
          ENDIF
          READ*,CRIT(I,2)
        ENDIF
100   CONTINUE
*
      DO 920 I=1,6
        PRINT*
920   CONTINUE
*
201   PRINT*,'ENTER THE NUMBER OF DIFFERENT WEAPON PATTERNS'
      PRINT*,'USED IN THE ATTACK (MAX 12)==> '
      READ*,NPATT
      IF (NPATT.GT.12) GO TO 201
      PRINT*
      PRINT*
*
      DO 200 I=1,NPATT
       PRINT*,'DESCRIBE WEAPONS DELIVERY PATTERN NUMBER ',I,'.'
       PRINT*,'ENTER:'
202    PRINT*,'NUMBER OF WEAPONS IN THE PATTERN (MAX 12)==> '
       READ*,IPAT(I,1)
       IF ((IPAT(I,1).GT.12).OR.(IPAT(I,1).LT.1)) GO TO 202
       PRINT*,'WEAPON/CANISTER FUZE RELIABILITY==> '
       READ*,PATT(I,9)
       PRINT*,'NUMBER OF CRATERS PER WEAPON.'
       PRINT*,'  ENTER 1 FOR GP OR GUIDED MUNITIONS,'
       PRINT*,'  OR THE NUMBER OF BOMBLETS FOR CLUSTERED UNITS.'
       PRINT*,'  ENTER NUMBER==> '
       READ*,IPAT(I,2)
203    PRINT*,'WEAPON CODE FOR CRATER TABLE==> '
       READ*,IPAT(I,3)
       IF (IPAT(I,3).GT.NWPN) THEN
         PRINT*,'ONLY STORED ',NWPN,' WEAPON TYPES.'
         PRINT*,'CORRECT OR TERMINATE AND FIGURE IT OUT.'
         GO TO 203
       ENDIF
204    PRINT*,'INDIVIDUAL WEAPON TRAJECTORY CODE:'
       PRINT*,'    0: DUMB, GENERAL PURPOSE WEAPONS'
       PRINT*,'    1: CBU, RECTANGULAR PATTERN (DOES NOT ALLOW VOIDS)'
       PRINT*,'    2: CBU, ELLIPTICAL PATTERN (ALLOWS VOID AREA)'
       PRINT*,'    3: GUIDED MUNTION'
       PRINT*,'ENTER CODE==> '
       READ*,IPAT(I,4)
       IF ((IPAT(I,4).LT.0).OR.(IPAT(I,4).GT.3)) GO TO 204
       PATT(I,10)=0.
       IF (IPAT(I,4).LT.3) THEN
         PRINT*,'WEAPON PATTERN NUMBER ',I,' USES DUMB BOMBS.'
         PRINT*,'DESCRIBE ERROR DISTRIBUTIONS WITH STD DEVS.  ENTER:'
         PRINT*,'AIMPOINT, RANGE SIGMA==> '
```

```
      READ*,PATT(I,1)
      PRINT*,'     DEFLECTION SIGMA==> '
      READ*,PATT(I,2)
      PRINT*,'INDVDL WPN BALLISTIC DISPERSION, RANGE SIGMA==> '
      READ*,PATT(I,3)
      PRINT*,'                                DEFLECTION SIGMA==> '
      READ*,PATT(I,4)
      IF ((PAT(I,1).GT.1) THEN
        PRINT*,'DESCRIBE THE STICK.  FOR EACH WEAPON, ENTER ITS'
        PRINT*,'SIGNED (+/-) RANGE AND DEFLECTION POSITION '
        PRINT*,'WITHIN THE STICK, REFERENCED TO THE AIMPOINT.'
        LASTJ=IPAT(I,1)
        DO 220 LJ=1,LASTJ
         JRNG=2*LJ+9
         JDEF=2*LJ+10
         PRINT*,'WPN ',LJ,' RNG COORD==> '
         READ*,PATT(I,JRNG)
         PRINT*,'WPN ',LJ,' DEF COORD==> '
         READ*,PATT(I,JDEF)
         PRINT*
220     CONTINUE
      ELSE
        PATT(I,11)=0.
        PATT(I,12)=0.
      ENDIF
      IF (IPAT(I,4).GT.0) THEN
        PRINT*,'DESCRIBE CBU BOMBLET DISTRIBUTION.  ENTER:'
        PRINT*,'BOMBLET FUZE RELIABILITY==> '
        READ*,PATT(I,10)
        PRINT*,'DISPENSER GROUND COVERAGE, LENGTH (RANGE)==> '
        READ*,AL
        PATT(I,5)=0.5*AL
        PRINT*,'DISPENSER GROUND COVERAGE, WIDTH (DEFLECTION)==> '
        READ*,AW
        PATT(I,6)=0.5*AW
        IF (IPAT(I,4).GT.1) THEN
          PRINT*,'VOID LENGTH (RANGE)==> '
          READ*,VL
          PATT(I,7)=0.5*VL
          PRINT*,'VOID WIDTH (DEFLECTION)==> '
          READ*,VW
          PATT(I,8)=0.5*VW
        ELSE
          PATT(I,7)=0.
          PATT(I,8)=0.
        ENDIF
      ELSE
        PATT(I,5)=0.
        PATT(I,6)=0.
        PATT(I,7)=0.
        PATT(I,8)=0.
```

```
        ENDIF
      ELSE
        PRINT*,'WEAPON PATTERN NUMBER ',I,' USES GUIDED MUNITIONS.'
205     PRINT*,'DESCRIBE ERROR DISTRIBUTION WITH STD DEV OR CEPS:'
        PRINT*,'          1: ENTRY AS CEP'
        PRINT*,'          2: ENTRY AS STD DEV (SIGMA)'
        PRINT*,'ENTER CHOICE==> '
        READ*,ICH
        IF ((ICH.NE.1).AND.(ICH.NE.2)) GO TO 205
        PRINT*,'ENTER:'
        IF (ICH.EQ.1) THEN
          PRINT*,'OPTIMAL GUIDANCE CEP==> '
          READ*,CEP1
          PRINT*,'NEAR-MISS CEP==> '
          READ*,CEP2
          PATT(I,1)=CEP1/0.675
          PATT(I,2)=CEP1/0.675
          PATT(I,3)=CEP2/0.675
          PATT(I,4)=CEP2/0.675
        ELSE
          PRINT*,'OPTIMAL GUIDANCE RANGE SIGMA==> '
          READ*,PATT(I,1)
          PRINT*,'OPTIMAL GUIDANCE DEFLECTION SIGMA==> '
          READ*,PATT(I,2)
          PRINT*,'NEAR-MISS RANGE SIGMA==> '
          READ*,PATT(I,3)
          PRINT*,'NEAR-MISS DEFLECTION SIGMA==> '
          READ*,PATT(I,4)
        ENDIF
        PRINT*,'GROSS ERROR RANGE SIGMA==> '
        READ*,PATT(I,5)
        PRINT*,'GROSS ERROR DEFLECTION SIGMA==> '
        READ*,PATT(I,6)
        PRINT*,'PROBABILITY OF OPTIMAL GUIDANCE==> '
        READ*,PATT(I,7)
        PRINT*,'PROBABILITY OF NEAR-MISS GUIDANCE==> '
        READ*,PATT(I,8)
      ENDIF
200   CONTINUE
      PRINT*
      PRINT*
      FRINT*,'HOW MANY PATCHES WILL RESOURCES ALLOW?==>'
      READ*,MXPTCH
      FRINT 925
925   FORMAT(
     1 1X,'SELECT CRATER REPAIR PRIORITY:',//,
     2 1X,'   0:  ALL TOL STRIPS IN ORDER OF TARGET NUMBER.',/,
     3 1X,'   1:  EASIEST TOL STRIP FIRST, REST IN ORDER.',/,
     4 1X,'   2:  REPAIR ONLY THE EASIEST TOL STRIP.',/,
     5 1X,'  10:  ALL PAVEMENTS IN ORDER OF TARGET NUMBER.',/,
     6 1X,'  11:  ALL APPROACHES AND EASIEST TOL STRIP FIRST,',/,
```

```
     7 1X,'       FOLLOWED BY OTHERS IN TARGET ORDER.',/,
     8 1X,'  12:  ALL APPROACHES AND ONLY EASIEST TOL STRIP.',//,
     9 1X,'CHOICE==> ')
      READ*,IREPR
      NPATCH=99999
      PRINT*
      PRINT*
      PRINT*,'ALMOST DONE.  DEFINE THE ATTACK.'
      PRINT*,'ENTER THE FOLLOWING:'
7     PRINT*,'NUMBER OF PASSES OVER THE COMPLEX (MAX 32)==> '
      READ*,NPASS
      IF (NPASS.GT.32) GO TO 7
8     PRINT*,'EACH AIRCRAFT MAY REATTACK ONE TIME.'
      PRINT*,'NUMBER OF AIRCRAFT PARTICIPATING IN THE ATTACK==> '
      READ*,NAC
      RAT=REAL(NPASS)/REAL(NAC)
      IF (RAT.GT.2.0) THEN
        PRINT*,'INSUFFICIENT A/C TO ACCOMPLISH ATTACK.'
        GO TO 8
      ENDIF
      KAC=0
      DO 401 I=1,NPASS
       PASS(I,5)=99.
       DO 401 J=1,2
        OPT(I,J)=0
401   CONTINUE
      DO 400 I=1,NPASS
       PRINT*,'PASS NUMBER ',I,'.'
       IF (OPT(I,1).EQ.0) THEN
         KAC=KAC+1
         IF (KAC.GT.NAC) THEN
           PRINT*,'DISCREPANCY IN NUMBER OF A/C.  RE-ENTER.'
           GO TO 7
         ENDIF
         PRINT 930,KAC
       ELSE
         PRINT 930,OPT(I,2)
       ENDIF
930    FORMAT(1X,'FLOWN BY A/C NUMBER ',I2,':')
       PRINT*,'AIMPOINT--X-COORD==> '
       READ*,PASS(I,1)
       PRINT*,'          Y-COORD==> '
       READ*,PASS(I,2)
       PRINT*,'ATTACK DIRECTION (REFERENCED CCW FROM +X-AXIS)==> '
       READ*,PASS(I,3)
       PASS(I,3)=PASS(I,3)*0.01745
9      PRINT*,'WEAPON PATTERN CODE, (ONE YOU DEFINED EARLIER)==> '
       READ*,IPASS(I,1)
       IF (IPASS(I,1).GT.NPATT) THEN
         PRINT*,'UNDEFINED PATTERN.  IF IRRECONCILABLE AT THIS'
         PRINT*,'INPUT POINT, YOU MUST TERMINATE, AND RESTART.'
```

```
      GO TO 9
    ENDIF
    IF (OPT(I,1).EQ.0) THEN
      PRINT*,'PROBABILITY A/C SURVIVES ENROUTE ATTRITION==> '
      READ*,PASS(I,4)
      PRINT*,'NUMBER OF NEXT PASS FOR THIS A/C==> '
      READ*,IPASS(I,2)
      IF (IPASS(I,2).GT.1) THEN
        PRINT*,'PROBABILITY A/C SLRVIVES TARGET AREA ATTRITION==> '
        READ*,PASS(I,5)
        OPT(IPASS(I,2),1)=1
        OPT(IPASS(I,2),2)=KAC
      ENDIF
    ELSE
      PASS(I,4)=1.
      IPASS(I,2)=0
      PASS(I,5)=88.
    ENDIF
400   CONTINUE
      PRINT*
      FRINT*
      PRINT*,'         *** DATA INPUT COMPLETE ***'
      PRINT*
      PRINT*
      WRITE(12,950)ISEED
      WRITE(12,950)NSAMP,NSAMPT
      WRITE(12,975)NFLAG3,ERROR,ZALPH
      WRITE(12,970)NELT,NTGPS,APPRCW,NAREA
      DO 500 I=1,NELT
       WRITE(12,955)(TGT(I,J),J=1,5),(ITGT(I,J),J=1,3)
       IF (ITGT(I,1).EQ.1) WRITE(12,955)CRIT(I,1),CRIT(I,2)
500   CONTINUE
      WRITE(12,950)NCP,LV
      WRITE(12,950)NPATT
      DO 510 I=1,NPATT
       WRITE(12,960)(IPAT(I,J),J=1,4),(PATT(I,J),J=1,10)
       LASTJ=IPAT(I,1)
       DO 510 LJ=1,LASTJ
        JRNG=2*LJ+9
        JDEF=2*LJ+10
        WRITE(12,955)PATT(I,JRNG),PATT(I,JDEF)
510   CONTINUE
      WRITE(12,950)NSFC,NWPN
      DO 521 I=1,NSFC
       WRITE(12,965)(CRTAB(I,J,1),J=1,NWPN)
521   CONTINUE
      DO 522 I=1,NSFC
       WRITE(12,965)(CRTAB(I,J,2),J=1,NWPN)
522   CONTINUE
      WRITE(12,950)MXPTCH,IREPR,NPASS
      DO 530 I=1,NPASS
```

```
      WRITE(12,955)(PASS(I,J),J=1,5),(IPASS(I,J),J=1,2)
530   CONTINUE
      CLOSE(UNIT=12)
      ENDIF
934   FORMAT(A1)
935   FORMAT(A6)
950   FORMAT(4I10)
955   FORMAT(5(F15.4,1X),3I10)
960   FORMAT(4I6,10(F9.3,1X))
965   FORMAT(F14.3,F12.3)
970   FORMAT(2I10,F10.1,I10)
975   FORMAT(I10,2F10.4)
      END
```

Appendix E

```
*********************************************************************
* LAST UPDATE 01/1200 MAR 84          FILE:MAIN.AAP                *
*********************************************************************
      PROGRAM AAPMOD
*********************************************************************
*                    AIRFIELD ATTACK PROGRAM                        *
*                                                                   *
*--USED AT EGLIN AFB, FL, AND 50-60 CONTRACTOR LOCATIONS.           *
*  DEVELOPED AT OKLAHOMA STATE UNIVERSITY,                          *
*  UNDER CONTRACT F08635-79-C-0255, FOR THE JOINT TECHNICAL COORDINATING
*  GROUP FOR MUNITIONS EFFECTIVENESS.                               *
*  MODIFIED BY CAPTAIN ROBERT N. MIGLIN TO PROVIDE INTERACTIVE      *
*  CAPABILITY FOR TACTICS ASSESSMENT.                               *
*                                                                   *
*--NON-ANSI.  ANSI=0 REQUIRED FOR CDC 6600.                         *
*                                                                   *
*********************************************************************
*
      CHARACTER FNAME*6,FNAME2*6
      INTEGER NPX(32)
*
      COMMON
     1 ADM(112)       ,GPHT(15)       ,MXPTCH         ,SIGADM(112),
     2 AMIN(3)        ,GPHTAC(15)                     ,SIGARP(3),
     3 APRA(3)        ,GPHTS(15)                      ,SIGASP(3),
     4 APRMIN(3)      ,LNHITS(112)    ,NSAMP1         ,SIGCRT(3),
     5 AREP(3)        ,ICRAT(4)       ,PASS(0:32,6)   ,SIGCTS(27),
     6 ASTP(3)        ,ICUT(4,3)      ,PATT(13,34)    ,SIGFIL(27),
     7 COUNTR(112)    ,IHIT(3)        ,RAPF(112)      ,SIGHTS(112),
     8 CRIT(112,2)    ,IPASS(32,2)    ,RCUT(112)      ,SIGNAF(112),
     9 CRTAB(11,6,2)  ,IPAT(12,4)     ,RHIT(112)      ,SMINA(4),
     & DECAR(112)     ,IPCUT(3)       ,SAPR(4)        ,SNAPFL(3),
     1 DSTR(3)                        ,SAPRA(4)       ,TGT(112,5),
     2 ENAPFL(3)      ,IPL(40)        ,SAVE(800,3)    ,XC(3),
     3 GPADAC(15)     ,ISAV(800)      ,SGAPR(4)       ,YC(3),
     4 GPADM(15)      ,ITGT(112,3)    ,SGAPRA(4),
     5 GPADMS(15)     ,I2CUT(4)       ,SGCRAT(4),
     6 GPAREA(15)     ,KH(3)          ,SGMINA(4)
*
      COMMON/RAY/TWOPI
*
      COMMON/RAY2/SQUARE(900),CRMAX
*
      COMMON/END/NSAMP2,NELT,NTGPS,NCP,CRMIN,APPRCW,NAREA
*
```

```
      COMMON/CATA/NUMAPR(4,2),STEMP(303)
*
      COMMON/JOHN/NFLAG1,NFLAG2,NMAX,NSAMPR,ZALPH,ERROR,NSAMP,NFLAG3
*
*-----INPUT/INITIALIZE
*
      TWCPI=6.28318530718
      SRPI04=0.8862269254528
      POV180=0.01745329252
*
      PRINT*,'NAME OF INPUT FILE==> '
      READ 901,FNAME
      OPEN(UNIT=12,FILE=FNAME,STATUS='OLD')
      REWIND 12
      PRIN~*,'NAME OF OUTPUT FILE==> '
      READ 901,FNAME2
      OPEN(UNIT=13,FILE=FNAME2,STATUS='NEW')
*
10    READ(12,991)ISEED
      CALL RANSET(ISEED)
      ITDI=1
      READ(12,*)NSAMP,NSAMPT
      READ(12,*)NFLAG3,ERROR,ZALPH
*
*-----READ TARGET DESCRIPTION
*
      READ(12,*)NELT,NTGPS,APPRCW,NAREA
      DO 30 I=1,NELT
       READ(12,*)(TGT(I,J),J=1,5),(ITGT(I,J),J=1,3)
       CRIT(I,1)=0.
       CRIT(I,2)=0.
       IF (ITGT(I,1).EQ.1) THEN
         READ(12,*)CRIT(I,1),CRIT(I,2)
       ENDIF
30    CONTINUE
      READ(12,*)NCP,LV
*
*-----READ PATTERN DESCRIPTIONS
*
      READ(12,*)NPATT
      DO 40 I=1,NPATT
       READ(12,*)(IPAT(I,J),J=1,4),(PATT(I,J),J=1,10)
       NVALS=IPAT(I,1)
       DO 40 IJ=1,NVALS
        JR=2*IJ+9
        JD=2*IJ+10
        READ(12,*)PATT(I,JR),PATT(I,JD)
40    CONTINUE
*
*-----READ CRATERING TABLE
*
```

```
      READ(12,*)M,N
      DO 51 I=1,M
       READ(12,*)(CRTAB(I,J,1),J=1,N)
51    CONTINUE
      DO 52 I=1,M
       READ(12,*)(CRTAB(I,J,2),J=1,N)
52    CONTINUE
      CRMIN=1.0E10
      CRMAX=0.
*
*-----READ MISSION DESCRIPTION
*  IREPR... TELLS WHAT TYPE OF REPAIRS ARE TO BE MADE
*                  = 0--ALL MAJOR PAVEMENTS (CRIT(L,1)>0)
*                       ARE REPAIRED IN ORDER INPUT
*                  = 1--EASIEST STRIP TO REPAIR FIXED FIRST,
*                       THEN REST WITH (CRIT(L,1)>0) IN ORDER INPUT
*                  = 2--ONLY EASIEST STRIP TO REPAIR IS DONE
*                  =1X--REPAIR STRIP AND APPROACH IN ORDER OF
*                      "X" ABOVE, I.E., 11 => APPROACHES AND 1.
*
      READ(12,*)MXPTCH,IREPR,NPASS
      DO 70 I=1,NPASS
       READ(12,*)(PASS(I,J),J=1,5),(IPASS(I,J),J=1,2),NPX(I)
       PASS(I,6)=PASS(I,5)
70    CONTINUE
      CLOSE(UNIT=12)
*
      PRINT*,'DO YOU WANT AN OUTPUT ECHO OF INPUT? 1=YES, 0=NO ==>'
      READ*,OECHO
      IF (OECHO.EQ.1) THEN
        WRITE(13,1970)
        WRITE(13,1950)ISEED
        WRITE(13,1950)NSAMP,NSAMPT
        WRITE(13,1980)NFLAG3,ERROR,ZALPH
        WRITE(13,1975)NELT,NTGPS,APPRCW,NAREA
        DO 1500 I=1,NELT
         WRITE(13,1955)(TGT(I,J),J=1,5),(ITGT(I,J),J=1,3)
         IF (ITGT(I,1).EQ.1) WRITE(13,1955)CRIT(I,1),CRIT(I,2)
1500    CONTINUE
        WRITE(13,1950)NCP,LV
        WRITE(13,1950)NPATT
        DO 1510 I=1,NPATT
         WRITE(13,1960)(IPAT(I,J),J=1,4),(PATT(I,J),J=1,10)
         LASTJ=IPAT(I,1)
         DO 1510 LJ=1,LASTJ
          JRNG=2*LJ+9
          JDEF=2*LJ+10
          WRITE(13,1955)PATT(I,JRNG),PATT(I,JDEF)
1510    CONTINUE
        WRITE(13,1950)M,N
        DO 1521 I=1,M
```

```
         WRITE(13,1965)(CRTAB(I,J,1),J=1,N)
1521     CONTINUE
         DO 1522 I=1,M
          WRITE(13,1965)(CRTAB(I,J,2),J=1,N)
1522     CONTINUE
         WRITE(13,1950)MXPTCH,IREPR,NPASS
         DO 1530 I=1,NPASS
          WRITE(13,1955)(PASS(I,J),J=1,5),(IPASS(I,J),J=1,2)
1530     CONTINUE
      ENDIF
*
*-----INITIALIZE FOR MONTE CARLO
*
      WRITE(13,905)FNAME,FNAME2
      NSAMPR=1
      DO 80 I=1,NELT
       ITGTTP=ITGT(I,2)
       DO 80 J=1,NPASS
        NPTRN=IPASS(J,1)
        JWPNTP=IPAT(NPTRN,3)
        IF (ITGT(I,1).EQ.1) THEN
          TBHLD1=CRTAB(ITGTTP,JWPNTP,1)
          TBHLD2=CRTAB(ITGTTP,JWPNTP,2)
          CRMIN=AMIN1(CRMIN,TBHLD1,TBHLD2)
          CRMAX=AMAX1(CRMAX,TBHLD1,TBHLD2)
        ENDIF
80    CONTINUE
      CALL INITL(NELT,NTGPS,NCP,LV)
      NMAX=0
*
*--TEST TO SEE IF LIMITING MONTE CARLO LOOPS IS BOTH DESIRED (NFLAG3=1)
*  AND APPROPRIATE (NSAMP>200).  IF SO, SET FLAGS AND SET INITIAL
*  MONTE CARLO LOOP LIMIT.
*
      IF ((NFLAG3.EQ.1).AND.(NSAMP.GE.200)) THEN
        NFLAG1=0
        NFLAG2=0
        NMAX=NSAMP
        NSAMP=200
      ENDIF
*
*
*-----MONTE CARLO LOOP  --  820 ON (IT)
*
E5    DO 820 IT=NSAMPR,NSAMP
*
*-----INITIALIZE VARIABLES WHICH GET RESET EACH MONTE CARLO REP
*
       NSAMP2=IT
100    DO 110 L=1,NELT
        DECAR(L)=TGT(L,4)*TGT(L,5)
```

```
110   CONTINUE
      DO 120 L=1,3
       IPCUT(L)=0
       IHIT(L)=0
       AMIN(L)=0.
       APRMIN(L)=0.
       APRA(L)=0.
120   CONTINUE
      N=0
      M0=0
      KZ=0
*
*------SET NUMBER OF HITS PER TARGET EQUAL TO ZERO
*
      DO 130 L=1,NELT
       LNHITS(L)=0
130   CONTINUE
*
*------COMPUTE IMPACT POINTS OF WEAPONS
*
200   DO 370 I=1,NPASS
*
*-------SEE IF A/C SURVIVED. IF YES, CHANGE NEXT PASS PS TO REATTACK PS
*                            IF NOT, CHANGE NEXT PASS PS TO 0.0,
*                                    AND LOG NO HITS FOR THIS PASS
*
       NXTP=IPASS(I,2)
       CRAZYN=RANF()
       IF (CRAZYN.GT.PASS(I,4)) THEN
         PASS(NXTP,4)=0.
         GO TO 370
       ELSE
         PASS(NXTP,4)=PASS(I,5)
       ENDIF
*
       NPTRN=IPASS(I,1)
       NWEP=IPAT(NPTRN,1)
       NBOM=IPAT(NPTRN,2)
       RMAJ=PATT(NPTRN,5)
       RMIN=PATT(NPTRN,6)
       VMAJ=PATT(NPTRN,7)
       VMIN=PATT(NPTRN,8)
       KODE=IPAT(NPTRN,4)
*
*-------LOCATE STICK PATTERN CENTER
*
       PASSXT=PASS(I,1)
       PASSYT=PASS(I,2)
*
*-------IF A TOL SURFACE, DISPLACE AIMPOINT FOR AIMPOINT ERROR
*
```

```
          IF (NPX(I).LE.NCP) THEN
            NTTT=NPX(I)
            CALL TRISUB(DAP)
            PASSXT=PASSXT+DAP*COS(TGT(NTTT,3))
            PASSYT=PASSYT+DAP*SIN(TGT(NTTT,3))
          ENDIF
          SINP=SIN(PASS(I,3))
          COSP=COS(PASS(I,3))
210       IF (KODE.EQ.3) THEN
*- - - - -GUIDED MUNITIONS...
            CRAZYN=RANF()
            IF (CRAZYN.LE.PATT(NPTRN,7)) THEN
              CALL NORAN (R,PATT(NPTRN,1),D,PATT(NPTRN,2))
            ELSE
              IF (CRAZYN.LE.PATT(NPTRN,8)) THEN
                CALL NORAN (R,PATT(NPTRN,3),D,PATT(NPTRN,4))
              ELSE
                CALL NORAN (R,PATT(NPTRN,5),D,PATT(NPTRN,6))
              ENDIF
            ENDIF
            X=PASSXT+R*COSP+D*SINP
            Y=PASSYT+R*SINP-D*COSP
          ELSE
*- - - - -DUMB BOMBS...
            CALL NORAN (R,PATT(NPTRN,1),D,PATT(NPTRN,2))
            XCTR=PASSXT+R*COSP+D*SINP
            YCTR=PASSYT+R*SINP-D*COSP
          ENDIF
*
*-------LOCATE WEAPON IMPACT OR CENTER OF DISPENSER PATTERN
*
          DO 360 K=1,NWEP
           CRAZYN=RANF()
           IF (CRAZYN.GT.PATT(NPTRN,9)) GO TO 360
           IF (KODE.LT.3) THEN
             CALL NORAN (R,PATT(NPTRN,3),D,PATT(NPTRN,4))
             K2=2*K+9
             XIWOD=XCTR+(PATT(NPTRN,K2)+R)*COSP+(PATT(NPTRN,K2+1)+D)*SINP
             YIWOD=YCTR+(PATT(NPTRN,K2)+R)*SINP-(PATT(NPTRN,K2+1)+D)*COSP
           ENDIF
*
*--------LOCATE IMPACTS  (NBOM = 1 OR NMBR BOMBLETS/CBL SHELL)
*
270        DO 350 M1=1,NBOM
            IF (KODE.LT.3) THEN
              X=XIWOD
              Y=YIWOD
              IF (NBOM.GT.1) THEN
                CRAZYN=RANF()
                IF (CRAZYN.GT.PATT(NPTRN,10)) GO TO 350
280             CRAZYN=RANF()
```

```
              X1=2.*RMAJ*CRAZYN-RMAJ
              CRAZYN=RANF()
              Y1=2.*RMIN*CRAZYN-RMIN
              IF (KODE.EQ.2) THEN
                X1Y1OL=(X1**2/RMAJ**2)+(Y1**2/RMIN**2)
                IF (X1Y1OL.GT.1.) GO TO 280
                IF ((VMAJ.GT.0.).AND.(VMIN.GT.0.)) THEN
                  X1Y1IL=(X1**2/VMAJ**2)+(Y1**2/VMIN**2)
                  IF (X1Y1IL.LT.1.) GO TO 280
                ENDIF
              ENDIF
290           X=X+X1*COSP+Y1*SINP
              Y=Y+X1*SINP-Y1*COSP
            ENDIF
          ENDIF
*
*---------CHECK FOR ANY HIT OR NEAR-MISS
*
300       DO 340 L=1,NELT
           SINT=SIN(TGT(L,3))
           COST=COS(TGT(L,3))
           XP=X-TGT(L,1)
           YP=Y-TGT(L,2)
           T1=XP*COST+YP*SINT
           XP=YP*COST-XP*SINT
           ITGTTP=ITGT(L,2)
           JWPNTP=IPAT(NPTRN,3)
           IF ((L.GT.NCP).AND.(L.LE.(LV+NCP))) THEN
            IF(ABS(T1)-CRTAB(ITGTTP,JWPNTP,2).GE..5*TGT(L,4)) GO TO 340
            IF(ABS(XP)-CRTAB(ITGTTP,JWPNTP,2).GE..5*TGT(L,5)) GO TO 340
           ELSE
            IF(ABS(T1)-CRTAB(ITGTTP,JWPNTP,1).GE..5*TGT(L,4)) GO TO 340
            IF(ABS(XP)-CRTAB(ITGTTP,JWPNTP,1).GE..5*TGT(L,5)) GO TO 340
           ENDIF
330        M=M+1
           IF (M.LE.800) THEN
             SAVE(M,1)=T1+.5*TGT(L,4)
             SAVE(M,2)=XP+.5*TGT(L,5)
             SAVE(M,3)=FLOAT(L)
             ISAV(M)=IPAT(NPTRN,3)
             COUNTR(L)=COUNTR(L)+1.
             LNHITS(L)=LNHITS(L)+1.
           ENDIF
340       CONTINUE
          IF (M.GT.800) WRITE(13,1200)I,M
          M=MIN0(M,800)
350      CONTINUE
360     CONTINUE
370    CONTINUE
       K1END=0
       IF (M.EQ.0) THEN
```

```
          ELSE
*
*---------TGT L IS A PAVEMENT
*
            IF (N.LT.1) THEN
              IF (L.LE.NCP) THEN
                XC(L)=.5*(TGT(L,4)+CRIT(L,1))
                YC(L)=.5*(TGT(L,5)-CRIT(L,2))
              ENDIF
              GO TO 730
            ENDIF
520         CALL SORT(N,SAVE(K0,1),SAVE(K0,2),SAVE(K0,3),ISAV(K0))
            IF (NAREA.EQ.0) CALL OVLAP
    1          (SAVE(K0,1),SAVE(K0,2),CRTAB,ITGT(L,2),ISAV(K0),0.,0.,
    2           IFIX(TGT(L,4)),IFIX(TGT(L,5)),N,SUMRUN)
*
*-----------TAXIWAYS (MINOR PAVEMENTS)
*           FIND WANDERING PATH ONLY FOR TAXI-ONLY TARGETS  (CRIT(L,1)=0.)
*
            IF (CRIT(L,1).LT.1.0) THEN
              CALL MINCW (CRMAX,N,SAVE(K0,1),SAVE(K0,2),
    1                     CRTAB(1,1,2),ITGT(L,2),ISAV(K0),
    2                     CRIT(L,2),TGT(L,5),NFILL,CUTS,ARFILL)
              ARFILS=ARFILL
              FILL=FLOAT(NFILL)
710           RHIT(L)=RHIT(L)+FILL
              NTXWY=L-NCP
              SIGFIL(NTXWY)=SIGFIL(NTXWY)+FILL*FILL
              RCUT(L)=RCUT(L)+CUTS
              SIGCTS(NTXWY)=SIGCTS(NTXWY)+CUTS*CUTS
            ELSE
*
*-----------RUNWAYS (MAJOR PAVEMENTS)
*    SEARCH FOR A CLEAR STRIP (LENGTH=CRIT(L,1) .X. WIDTH=CRIT(L,2))
*
              M0=K-1
              IF ((K.EQ.M).AND.(SAVE(M,3).EQ.FLOAT(L))) M0=M0+1
              IPCUT(L)=0
              IHIT(L)=0
              AMIN(L)=0.
              APRMIN(L)=0.
              APRA(L)=0.
              DO 540 KK=1,4
               DO 540 KK2=1,2
                NUMAPR(KK,KK2)=0
540           CONTINUE
              CALL CLSTRP(CRMAX,N,SAVE(K0,1),SAVE(K0,2),CRTAB,
    1                     ITGT(L,2),ISAV(K0),TGT(L,4),TGT(L,5),
    2                     CRIT(L,1),CRIT(L,2),XC(L),YC(L),NMIN)
              IF (NMIN.GT.0) THEN
                RCUT(L)=RCUT(L)+1.
```

```
          IPCUT(L)=1
        ENDIF
550     RHIT(L)=RHIT(L)+FLOAT(NMIN)
        IHIT(L)=NMIN
        SUMSTP=0.
        KM1=K-1
        IF ((K.EQ.M).AND.(SAVE(K,3).EQ.FLOAT(L))) KM1=K
        KA=1
        MFLAG=0
        XS1=XC(L)
        YS1=YC(L)
        XS2=XC(L)-CRIT(L,1)
        YS2=YC(L)+CRIT(L,2)
        KH(L)=KZ
        KP1=K0
560     CONTINUE
        IF ((KP1.LE.M).AND.(KM1.LE.M)) THEN
          ITGTTP=ITGT(L,2)
          DO 580 KW=KP1,KM1
           JWPNTP=ISAV(KW)
           IF(SAVE(KW,1)+CRTAB(ITGTTP,JWPNTP,KA).LE.XS2) GO TO 580
           IF(SAVE(KW,1)-CRMAX.GE.XS1) GO TO 590
           IF(SAVE(KW,1)-CRTAB(ITGTTP,JWPNTP,KA).GE.XS1) GO TO 580
           IF(SAVE(KW,2)+CRTAB(ITGTTP,JWPNTP,KA).LE.YS1) GO TO 580
           IF(SAVE(KW,2)-CRTAB(ITGTTP,JWPNTP,KA).GE.YS2) GO TO 580
           KZ=KZ+1
           IF (KW.NE.KZ) THEN
             S1=SAVE(KW,1)
             S2=SAVE(KW,2)
             S3=SAVE(KW,3)
             ITT=ISAV(KW)
             KZP=KZ+1
             DO 570 KB=KZP,KW
              KK=KW-KB+KZP
              SAVE(KK,1)=SAVE(KK-1,1)
              SAVE(KK,2)=SAVE(KK-1,2)
              SAVE(KK,3)=SAVE(KK-1,3)
              ISAV(KK)=ISAV(KK-1)
570          CONTINUE
             SAVE(KZ,1)=S1
             SAVE(KZ,2)=S2
             SAVE(KZ,3)=S3
             ISAV(KZ)=ITT
           ENDIF
580       CONTINUE
        ENDIF
590     IF (MFLAG.EQ.0) THEN
          KZT=0
          IF (KZ.NE.KH(L)) THEN
            KZT=KZ-KH(L)
            KK=KH(L)+1
```

```
              KH(L)=KZ
              IF (NAREA.LE.0) CALL OVLAP
     1          (SAVE(KK,1),SAVE(KK,2),CRTAB,ITGT(L,2),
     2           ISAV(KK),XC(L)-CRIT(L,1),YC(L),IFIX(CRIT(L,1)),
     3           IFIX(CRIT(L,2)),KZT,SUMSTP)
600           ASTP(L)=ASTP(L)+SUMSTP
              SIGASP(L)=SIGASP(L)+SUMSTP*SUMSTP
              AMIN(L)=SUMSTP
            ENDIF
610         MFLAG=1
            KA=2
            KP1=KP1+KZT
            KZ=KP1-1
            KZ1=KZ
            XS1=XC(L)-CRIT(L,1)
            IF (XS1.GE.CRIT(L,2)) THEN
              XS2=CRIT(L,2)
              GO TO 560
            ELSE
              GO TO 640
            ENDIF
          ENDIF
620       KZT=0
          NFILL=0
          IF (KZ.NE.KZ1) THEN
            KZT=KZ-KZ1
            KK=KZ1+1
            IF (MFLAG.GE.3) CALL SCRT
     1        (KZT,SAVE(KK,2),SAVE(KK,1),SAVE(KK,3),ISAV(KK))
            DO 892 II=1,KZT
             SAVE(KK+II-1,2)=SAVE(KK+II-1,2)-YS1
             SAVE(KK+II-1,1)=SAVE(KK+II-1,1)-XS2
892         CONTINUE
*
            IF (MFLAG.LE.2)
     1        CALL MINCW(CRMAX,KZT,SAVE(KK,1),SAVE(KK,2),
     2        CRTAB(1,1,2),ITGT(L,2),ISAV(KK),APPRCW,
     3        CRIT(L,2),NFILL,CUTS,ARFILL)
*
            IF (MFLAG.GE.3)
     1        CALL MINCW(CRMAX,KZT,SAVE(KK,2),SAVE(KK,1),
     2        CRTAB(1,1,2),ITGT(L,2),ISAV(KK),APPRCW,
     3        CRIT(L,2),NFILL,CUTS,ARFILL)
*
            DO 893 II=1,KZT
             SAVE(KK+II-1,2)=SAVE(KK+II-1,2)+YS1
             SAVE(KK+II-1,1)=SAVE(KK+II-1,1)+XS2
893         CONTINUE
            ARFILS=ARFILS+ARFILL
          ENDIF
630       NUMAPR(MFLAG,2)=KZT
```

```
              KZ=KZ1+NFILL
              KZ1=KZ
              NUMAPR(MFLAG,1)=NFILL
              FILL=FILL+FLOAT(NFILL)
              IF (KZ.EQ.KM1) GO TO 670
              GO TO (640,650,660,670),MFLAG
*
              PRINT*,'ERR GOTO ORIGINAL LINE NUMBER 733'
*
640           MFLAG=2
              NFILL=0
              KP1=KP1+KZT
              XS1=TGT(L,4)-CRIT(L,2)
              IF (XC(L)+CRIT(L,2).LE.TGT(L,4)) THEN
                XS2=XC(L)
                GO TO 560
              ENDIF
*
650           MFLAG=3
              KP1=KP1-NUMAPR(1,2)+NUMAPR(1,1)+NFILL
              CALL SORT(K-KP1,SAVE(KP1,1),SAVE(KP1,2),
     1                    SAVE(KP1,3),ISAV(KP1))
              XS1=CRIT(L,2)
              YS1=0.
              XS2=0.
              YS2=YC(L)+CRIT(L,2)
              GO TO 560
*
660           MFLAG=4
              KP1=KP1+KZT
              XS1=TGT(L,4)
              XS2=TGT(L,4)-CRIT(L,2)
              GO TO 560
*
670           KZ=KH(L)
              IF ((IREPR.GE.10).AND.(FILL.GT.0.)) THEN
                WRITE(13,890)L,KH(L),K0,FILL,
     1              (SAVE(KK,1),SAVE(KK,2),SAVE(KK,3),KK=1,M)
                KZ=KH(L)+IFIX(FILL+.01)
                IF (L.GT.1) THEN
                  K8=IFIX(FILL+.01)
                  DO 690 KZ1=1,K8
                   KK=K0+KZ1-1
                   S1=SAVE(KK,1)
                   S2=SAVE(KK,2)
                   S3=SAVE(KK,3)
                   IS=ISAV(KK)
                   KZP=KH(L)+KZ1+1
                   DO 680 KW=KZP,KK
                    KW1=KK-KW+KZP
                    SAVE(KW1,1)=SAVE(KW1-1,1)
```

```
                 SAVE(KW1,2)=SAVE(KW1-1,2)
                 SAVE(KW1,3)=SAVE(KW1-1,3)
                 ISAV(KW1)=ISAV(KW1-1)
680             CONTINUE
                KZP=KZP-1
                SAVE(KZP,1)=S1
                SAVE(KZP,2)=S2
                SAVE(KZP,3)=S3
                ISAV(KZP)=IS
690            CONTINUE
             ENDIF
           ENDIF
700        SIGCRT(L)=SIGCRT(L)+FLOAT(NMIN)**2
           ENAPFL(L)=ENAPFL(L)+FILL
           SNAPFL(L)=SNAPFL(L)+FILL**2
           APRMIN(L)=FILL
           GO TO 720
         ENDIF
720      ADM(L)=ADM(L)+SUMRUN
         ITGTGP=ITGT(L,3)
         GPADAC(ITGTGP)=GPADAC(ITGTGP)+SUMRUN
         SIGADM(L)=SIGADM(L)+SUMRUN*SUMRUN
         RAPF(L)=RAPF(L)+ARFILS
         SIGNAF(L)=SIGNAF(L)+ARFILS*ARFILS
         IF (CRIT(L,1).GT.0.) APRA(L)=ARFILS
       ENDIF
730    CONTINUE
       L=L+1
       K0=K
       FILL=0.
       ARFILL=0.
       ARFILS=0.
       CUTS=0.
       SUMRUN=0.
       IF (SAVE(K,3).GT.FLOAT(L)) GO TO 430
       IF ((K.EQ.M).AND.(SAVE(K,3).EQ.FLOAT(L))) GO TO 430
       IF ((L.LE.NELT).AND.(K.EQ.M)) GO TO 430
     ENDIF
740  CONTINUE
     DO 750 J=1,NTGPS
      GPADMS(J)=GPADMS(J)+GPADAC(J)**2
750  CONTINUE
     II3=1
*
*------COMPUTE COMBINED PROBABILITIES FOR RUNWAY, TAXIWAY,AND SOD
*
     IF (NCP.GT.1) THEN
       I3=0
       KJ=1
       IFIN=0
       DO 790 JJ=1,2
```

```
        DO 790 JK=1,NCP
*
*----------ONLY INTERESTED IN 1&2 (KJ=1), 1&3 (KJ=2), 2&3 (KJ=3)
*
         IF (JJ.GE.JK) GO TO 790
         IF (IPCUT(II3).EQ.0) GO TO 760
         IF (IPCUT(JJ).NE.1) II3=JJ
         IF (IPCUT(JK).NE.1) II3=JK
760      IF ((IPCUT(JJ).NE.1).OR.(IPCUT(JK).NE.1)) GO TO 780
*
*----------BOTH SURFACES ARE CUT
*
         I3=I3+1
*
*-----II INDICATES WHICH SURFACE HAS THE MINIMUM NUMBER OF CRATERS TO
*     REPAIR FOR COMBINATIONS OF 2 SURFACES AND II3 FOR ALL 3 SURFACES
*
         II=JJ
         IF (IHIT(JJ).GT.IHIT(JK)) II=JK
         IF (IHIT(II3).GT.IHIT(JK)) II3=JK
*
*----------DISTRIBUTION OF MINIMUM NUMBER OF CRATERS
*
770      ICUT(KJ,II)=ICUT(KJ,II)+1
         I2CUT(KJ)=I2CUT(KJ)+1
         SGCRAT(KJ)=SGCRAT(KJ)+FLOAT(IHIT(II))**2
*
*----------MINIMUM NUMBER OF CRATERS
*
         ICRAT(KJ)=ICRAT(KJ)+IHIT(II)
*
*----------AREA OF CRATERS
*
         SMINA(KJ)=SMINA(KJ)+AMIN(II)
         SGMINA(KJ)=SGMINA(KJ)+AMIN(II)**2
*
*----------MINIMUM NUMBER OF CRATERS ON APPROACH TO OPERATIONAL STRIP
*
         SAPR(KJ)=SAPR(KJ)+APRMIN(II)
         SGAPR(KJ)=SGAPR(KJ)+APRMIN(II)**2
*
*----------AREA OF CRATERS ON APPROACH
*
         SAPRA(KJ)=SAPRA(KJ)+APRA(II)
         SGAPRA(KJ)=SGAPRA(KJ)+APRA(II)**2
         IF (IFIN.EQ.1) GO TO 800
780      KJ=KJ+1
         IF ((JJ.NE.2).OR.(Jk.NE.3)) GO TO 790
*
*-----ALL COMBINATIONS OF 2 SURFACES HAVE BEEN LOOKED AT. IF ALL 3
*     SURFACES HAVE BEEN CUT (I3=3) COMPUTE STATISTICS FOR ALL 3 & EXIT
```

```
*     LOOP (IFIN=1).
*
          IF (I3.NE.3) GO TO 800
          KJ=4
          II=II3
          IFIN=1
          GO TO 770
790     CONTINUE
      ENDIF
800   CALL REPAIR(NXPTCH,KZ,M0,IREPR,CRMAX,II3,NAREA,NCP)
      M=M0
      M0=0
      KZ=0
      I1=0
810   IF (IT.GT.1) THEN
        IF (MOD(IT,NSAMPT).EQ.0) CALL RESLTS
      ENDIF
E20  CONTINUE
*
*-----TEST TO SEE IF LIMITING MONTE CARLO LOOP WAS DESIRED
*     AND APPROPRIATE. IF NOT, AVOID SUBROUTINE "NCOMP".
*
     IF ((NFLAG3.EQ.1).AND.(NSAMP.GE.200)) THEN
*
*-----TESTS ON FLAGS SET INSIDE SUBROUTINE "NCOMP" TO DIRECT
*     EITHER RETURN TO MONTE CARLO LOOP OR PASS ON, BASED ON
*     ESTIMATE OF ITERATIONS REQUIRED.
*
       IF (NFLAG2.EQ.0) CALL NCOMP
825    IF (NFLAG1.EQ.0) THEN
         NFLAG1=1
         GO TO 85
       ENDIF
     ENDIF
*
*-----CALCULATE AND PRINT STATISTICS
*
E30  IF (MOD((IT-1),NSAMPT).NE.0) CALL RESLTS
     CLOSE(UNIT=13)
*
840  FORMAT (1X,'NO HITS DURING ATTACK, MONTE CARLO ITERATION: ',I4)
890  FORMAT (8H TARGET ,I3,9H KH(L) = ,I4,6H K0 = ,I4,8H FILL = ,F7.0,8
    100(/1X,3F10.2))
901  FORMAT(A6)
905  FORMAT('1       INPUT FILE: ',A6,'     OUTPUT FILE: ',A6,//)
991  FORMAT(I10)
1200 FORMAT (1H0,37HMORE THAN 800 HITS WERE FOUND IN PASS,I4,1H./1X,20H
    1EXCESS WERE IGNORED.)
1934 FORMAT(A1)
1935 FORMAT(A6)
1950 FORMAT(4I10)
```

```
1955 FORMAT(5(F15.4,1X),3I10)
1960 FORMAT(4I6,10(F9.3,1X))
1965 FORMAT(6F12.1)
1970 FORMAT('1',T20,'*** DATA INPUT ECHO ***',//)
1975 FORMAT(2I10,F10.1,I10)
1980 FORMAT(I10,2F10.4)
     END
*********************************************************************
* LAST UPDATE 24/2300 FEB 84          FILE:SUBS1.AAP
*********************************************************************
     SUBROUTINE TRISUB(RV)
     U=RANF()
     X=SQRT(2.0*U)
     RV=1000.0*X-1000.0
     RETURN
     END
*--------------------------------------------------------------------
     SUBROUTINE NORAN(R,SR,D,SD)
*
     COMMON/RAY/TWOPI
*
     X=RANF()
     A=SQRT(-2.*ALOG(X))
     X=RANF()
     X=TWOPI*X
     R=A*SR*SIN(X)
     D=A*SD*COS(X)
     RETURN
     END
*--------------------------------------------------------------------
     SUBROUTINE INITL(NELT,NTGPS,NCP,LV)
*
     COMMON
    1 ADM(112)       ,GPHT(15)      ,NXPTCH        ,SIGADM(112),
    2 AMIN(3)        ,GPHTAC(15)                   ,SIGARP(3),
    3 APRA(3)        ,GPHTS(15)                    ,SIGASP(3),
    4 APRMIN(3)      ,LNHITS(112)   ,NSAMP1        ,SIGCRT(3),
    5 AREP(3)        ,ICRAT(4)      ,PASS(0:32,6)  ,SIGCTS(27),
    6 ASTP(3)        ,ICUT(4,3)     ,PATT(13,34)   ,SIGFIL(27),
    7 COUNTR(112)    ,IHIT(3)       ,RAPF(112)     ,SIGHTS(112),
    8 CRIT(112,2)    ,IPASS(32,2)   ,RCUT(112)     ,SIGNAF(112),
    9 CRTAB(11,6,2)  ,IPAT(12,4)    ,RHIT(112)     ,SMINA(4),
    & DECAR(112)     ,IPCUT(3)      ,SAPR(4)       ,SNAPFL(3),
    1 DSTR(3)                       ,SAPRA(4)      ,TGT(112,5),
    2 ENAPFL(3)      ,IPL(40)       ,SAVE(800,3)   ,XC(3),
    3 GPADAC(15)     ,ISAV(800)     ,SGAPR(4)      ,YC(3),
    4 GPADM(15)      ,ITGT(112,3)   ,SGAPRA(4),
    5 GPADMS(15)     ,I2CUT(4)      ,SGCRAT(4),
    6 GPAREA(15)     ,KH(3)         ,SGMINA(4)
*
     DO 10 I=1,NELT
```

```
      COUNTR(I)=0.
      SIGHTS(I)=0.
      ADM(I)=0.
      SIGADM(I)=0.
10    CONTINUE
      DO 20 J=1,NTGPS
      GPHTS(J)=0.
      GPADMS(J)=0.
20    CONTINUE
      IPAV=LV+NCP
      DO 30 K=1,IPAV
      RAPF(K)=0.
      SIGNAF(K)=0.
      RCUT(K)=0.
      RHIT(K)=0.
      SIGCTS(K)=0.
      SIGFIL(K)=0.
30    CONTINUE
      DO 40 L=1,NCP
      SIGCRT(L)=0.
      ASTP(L)=0.
      SIGASP(L)=0.
      AREP(L)=0.
      ENAPFL(L)=0.
      SNAPFL(L)=0.
      IHIT(L)=0
      IPCUT(L)=0
      AMIN(L)=0.
      APRMIN(L)=0.
      APRA(L)=0.
      DSTR(L)=0.
      SIGARP(L)=0.
40    CONTINUE
      N1=NCP+1
      DO 50 I=1,N1
      I2CUT(I)=0
      ICRAT(I)=0
      SGCRAT(I)=0.
      SMINA(I)=0.
      SGMINA(I)=0.
      SAPR(I)=0.
      SGAPR(I)=0.
      SAPRA(I)=0.
      SGAPRA(I)=0.
      DO 50 J=1,NCP
       ICUT(I,J)=0
50    CONTINUE
      RETURN
      END
*------------------------------------------------------------------------
      SUBROUTINE SORT(N,XI,YI,ZI,IX)
```

```
*
      DIMENSION IX(N),ZI(N),XI(N),YI(N)
*
      EQUIVALENCE (IT,T)
*
      JO=0
10    JO=JO+JO+1
      IF (JO.LT.N) GO TO 10
20    JO=JO/2
      IF (JO.LE.0) RETURN
      KO=N-JO
      DO 40 LO=1,KO
       MO=LO
30     NO=MO+JO
       IF (XI(MO).GT.XI(NO)) THEN
         T=XI(MO)
         XI(MO)=XI(NO)
         XI(NO)=T
         T=YI(MO)
         YI(MO)=YI(NO)
         YI(NO)=T
         T=ZI(MO)
         ZI(MO)=ZI(NO)
         ZI(NO)=T
         IT=IX(MO)
         IX(MO)=IX(NO)
         IX(NO)=IT
         MO=MO-JO
         IF (MO.GT.0) GO TO 30
       ENDIF
40    CONTINUE
      GO TO 20
      END
*-----------------------------------------------------------------------
      SUBROUTINE BLDG(XI,YI,CRTAB,L,NP,N,TL,TW,AREA)
*
      DIMENSION XI(N),YI(N),CRTAB(11,6,2),NP(N)
*
*-----ASSESS AREA REMAINING UNDAMAGED AFTER ALL HITS ARE
*     EVALUATED FOR THIS ATTACK
*
      RATIO=TL/TW
      DO 10 J=1,N
       DW=SQRT(AREA/RATIO)
       DL=DW*RATIO
       XH=.5*(TL-DL)
       YH=.5*(TW-DW)
       XOC=.5*TL-XH
       YOC=.5*TW-YH
       XCEN=XI(J)-XH
       YCEN=YI(J)-YH
```

```
         D1=ABS(YCEN-YOC)
         D2=ABS(XCEN-XOC)
         NPJ=NP(J)
         IF ((D1.LT.(CRTAB(L,NPJ,1)+0.5*DW)).AND.
     1       (D2.LT.(CRTAB(L,NPJ,1)+0.5*DL))) THEN
           KA=1
           IF ((D1.LE.(0.5*TW)).AND.(D2.LE.(0.5*TL))) KA=2
           OWDTH=AMIN1(DW,YCEN+CRTAB(L,NPJ,KA))
           OWDTH=OWDTH-AMAX1(0.,YCEN-CRTAB(L,NPJ,KA))
           OLNGTH=AMIN1(DL,XCEN+CRTAB(L,NPJ,KA))
           OLNGTH=OLNGTH-AMAX1(0.,XCEN-CRTAB(L,NPJ,KA))
           OAREA=OLNGTH*OWDTH
           AREA=AREA-OAREA
           IF (AREA.LE.0.) RETURN
         ENDIF
10      CONTINUE
        RETURN
        END
*-----------------------------------------------------------------
******************************************************************
* LAST UPDATE 16/2300 JAN 84              FILE:SUBS2.AAP
******************************************************************
        SUBROUTINE CLSTRP(CRMAX,N,XI,YI,CRTAB,LT,NP,TL,TW,CL,CW,XSTAR,
     1                   YSTAR,ICSTAR)
        DIMENSION XI(N),YI(N),CRTAB(11,6),AREA(800),ISORT(800),JSORT(800),
     1            NP(N)
        XC=0.0
        YC=0.0
        TSXU=CL
        TSYU=CW
        CSTAR=10.0E15
        ICSTAR=N
*
*-----DEFINE AREA(J)=DIFFICULTY OF REPAIRING CRATER J
*     CHANGED 28 OCT 81 TO COMPUTE AREA OF SQUARE CRATERS
*
        DO 24 J=1,N
         AREA(J)=4.0*CRTAB(LT,NP(J))**2
24      CONTINUE
*
*-----SET UP FOR SWEEP
*
25      NMIN=0
        ISTART=0
        SWEP=10.E15
        DO 11 J=1,N
         IF ((YI(J)+CRTAB(LT,NP(J)).GT.YC).AND.
     1       (YI(J)-CRTAB(LT,NP(J)).LT.TSYU)) THEN
*
14         IF (NMIN.EQ.0) THEN
             NMIN=1
```

```
            ISORT(1)=J
            JSORT(1)=J
            GO TO 11
          ENDIF
*
[illegible]          IT=NMIN
          NMIN=NMIN+1
17        JZ=ISORT(IT)
*
          IF ((XI(J)+CRTAB(LT,NP(J))).LT.(XI(JZ)+CRTAB(LT,NP(JZ)))) THEN
            ISORT(IT+1)=ISORT(IT)
            IT=IT-1
            IF (IT.GT.0) GO TO 17
            ISORT(1)=J
          ELSE
18          ISORT(IT+1)=J
          ENDIF
*
116       IT=NMIN-1
117       JR=JSORT(IT)
*
          IF ((YI(J)+CRTAB(LT,NP(J))).LT.(YI(JR)+CRTAB(LT,NP(JR)))) THEN
            JSORT(IT+1)=JSORT(IT)
            IT=IT-1
            IF (IT.GT.0) GO TO 117
            JSORT(1)=J
          ELSE
118         JSORT(IT+1)=J
          ENDIF
        ENDIF
11    CONTINUE
*
*-----EXECUTE SWEEP
*     DETERMINE DIFFICULTY OF REPAIRING CRATERS TOUCHING FRAME
*
10    IX=ISTART+1
      AICC=0.0
      ICC=0
30    IF (IX.LE.NMIN) THEN
        JM=ISORT(IX)
        IF ((XI(JM)-CRTAB(LT,NP(JM))).LT.TSXU) THEN
          AICC=AICC+AREA(JM)
          ICC=ICC+1
31        IX=IX+1
          GO TO 30
        ELSE
32        IF ((XI(JM)-CRMAX).LT.TSXU) THEN
            IX=IX+1
            GO TO 30
          ENDIF
        ENDIF
```

```
      ENDIF
*
*-----COMPARE REPAIR DIFFICULTY FOR FRAME
*
6     IF (CSTAR.GT.AICC) THEN
        CSTAR=AICC
        ICSTAR=ICC
        XSTAR=XC
        YSTAR=YC
        IF (CSTAR.LE.0.0000001) THEN
          XSTAR=XSTAR+CL
          RETURN
        ENDIF
      ENDIF
*
*-------MOVE FRAME
*
16    TEMP=AICC-CSTAR
41    ISTART=ISTART+1
      IF (ISTART.LE.NMIN) THEN
        IS=ISORT(ISTART)
        IF (TEMP.GT.AREA(IS)) THEN
          TEMP=TEMP-AREA(IS)
          GO TO 41
        ENDIF
998     IF (SWEP.GT.AICC) SWEP=AICC
        TSXU=XI(IS)+CRTAB(LT,NP(IS))+CL+0.000000001
        IF (TSXU.LE.TL) THEN
          XC=TSXU-CL
          GO TO 10
        ENDIF
      ENDIF
*
*-----SWEEP FINISHED
*
20    TEMP=SWEP-CSTAR
      JDP=0
46    JDP=JDP+1
      IF (JDP.GT.NMIN) THEN
        XSTAR=XSTAR+CL
        RETURN
      ENDIF
      IS=JSORT(JDP)
      IF (TEMP.GT.AREA(IS)) THEN
        TEMP=TEMP-AREA(IS)
        EO TO 46
      ENDIF
45    TSYU=YI(IS)+CRTAB(LT,NP(IS))+CW+0.000000001
      IF (TSYU.GT.TW) THEN
        XSTAR=XSTAR+CL
        RETURN
```

```
      ENDIF
      YC=TSYU-CW
      XC=0.0
      TSXU=CL
      GO TO 25
      END
*-------------------------------------------------------------------
      SUBROUTINE MINCW(CRMAXX,N,X,Y,CR,LT,KP,W,WW,NREP,CUTS,ATOTAL)
*
*-----HARNETT'S TAXIWAY PROGRAM INSERTED TO REPLACE MINCW 1 OCT 81
*     LATEST VERSION OF TAXIWAY  23 APRIL 1982
*           NC = MAX NUMBER OF CRATERS IN A SUBPROBLEM
*           NSUB = MAX NUMBER OF SUBPROBLEMS TO BE SOLVED
*           N = NUMBER OF CRATERS IN ENTIRE PROBLEM
*
       DIMENSION ISTART(1001),A(100),X(N),Y(N),CR(11,6),
    1            LIST1(50),LIST2(50),IT(50),WX(50),WY(50),WR(50),
    2            IREP(50),KP(N),IPSOL(50),ICOMP(50),IBEAS(50)
*
      COMMON/TAXI/NFM,NF,NL
*
      CRMAX=0.0
      IF (N.GT.50) THEN
        WRITE(6,799)N
        CALL EXIT
      ENDIF
*
*------CHANGED TO COMPUTE AREA OF SQUARE CRATERS 28 OCT 81
*
750   DO 100 J=1,N
       IF (CRMAX.LT.CR(LT,KP(J))) CRMAX=CR(LT,KP(J))
       A(J)=4.0*CR(LT,KP(J))**2
100   CONTINUE
*
      NREP=0
      ATOTAL=0.0
*
*-----SEARCH FOR SUBPROBLEMS
*
      ISTART(1)=1
      NSUB=1
      NNM=N-1
      DO 110 J=1,NNM
       JP=J+1
       JM=J
       EL=X(J)+CR(LT,KP(J))
       EU=X(JP)-CR(LT,KP(JP))
       IF ((EL+W).LE.EU) THEN
*
10:      JM=JM-1
         IF (JM.GE.1) THEN
```

```
        IF ((X(JM)+CR(LT,KP(JM))).GT.EL) EL=X(JM)+CR(LT,KP(JM))
        IF ((X(JM)+CRMAX).GT.EL) GO TO 101
      ENDIF
*
103   JP=JP+1
      IF (JP.LE.N) THEN
        IF (EU.GT.(X(JP)-CR(LT,KP(JP)))) EU=X(JP)-CR(LT,KP(JP))
        IF (EU.GT.(X(JP)-CRMAX)) GO TO 103
      ENDIF
*
105   IF ((EL+W).LE.EU) THEN
        NSUB=NSUB+1
        IF (NSUB.GT.1000) THEN
          WRITE(6,798)
          CALL EXIT
        ENDIF
760     ISTART(NSUB)=J+1
      ENDIF
     ENDIF
110  CONTINUE
     ISTART(NSUB+1)=N+1
*
*------SOLVE SUBPROBLEMS
*
     DO 230 JS=1,NSUB
      NF=ISTART(JS)
      NL=ISTART(JS+1)-1
      NFM=NF-1
      CRMAX=0.0
      DO 5 J=NF,NL
       IF (CRMAX.LT.CR(LT,KP(J))) CRMAX=CR(LT,KP(J))
5     CONTINUE
      NC=NL-NFM
      IF (NC.GT.50) THEN
        WRITE(6,797)NC
        CALL EXIT
      ENDIF
770   IF (NC.LE.2) THEN
        BFEAS=0.0
        NP=NF+1
        IF (Y(NF)+CR(LT,KP(NF)).GT.WW-W) THEN
          IF (Y(NF)-CR(LT,KP(NF)).GE.W) GO TO 122
          BFEAS=BFEAS+A(NF)
          NREP=NREP+1
          IREP(NREP)=NF
          ATOTAL=ATOTAL+A(NF)
          IF (NC.LE.1) GO TO 230
          IF (Y(NP)+CR(LT,KP(NP)).LE.WW-W) GO TO 230
          IF (Y(NP)-CR(LT,KP(NP)).GE.W) GO TO 230
          BFEAS=BFEAS+A(NP)
          NREP=NREP+1
```

```
            IREP(NREP)=NP
            ATOTAL=ATOTAL+A(NP)
            GO TO 230
          ENDIF
112       IF (NC.LE.1) GO TO 230
          IF (Y(NP)+CR(LT,KP(NP)).LE.WW-W) GO TO 230
          IF (Y(NP)-CR(LT,KP(NP)).GE.W) GO TO 114
113       ATOTAL=ATOTAL+A(NP)
          BFEAS=BFEAS+A(NP)
          NREP=NREP+1
          IREP(NREP)=NP
          GO TO 230
114       XD=X(NF)-X(NP)
          YD=Y(NF)-Y(NP)
          DIST=SQRT(XD**2+YD**2)-2.0*CR(LT,KP(NP))
          IF (DIST.GE.W) GO TO 230
          IF (((Y(NF)-CR(LT,KP(NF))).GE.W).AND.
     1        ((Y(NP)-CR(LT,KP(NP))).GE.W)) GO TO 230
          AMIN=A(NF)
          ISAVE=NF
          IF (A(NF).GT.A(NP)) ISAVE=NP
          IF (A(NF).GT.A(NP)) AMIN=A(NP)
          ATOTAL=ATOTAL+AMIN
          NREP=NREP+1
          IREP(NREP)=ISAVE
          BFEAS=BFEAS+AMIN
          GO TO 230
122       IF (NC.LE.1) GO TO 230
          IF (Y(NP)-CR(LT,KP(NP)).GE.W) GO TO 230
          IF (Y(NP)+CR(LT,KP(NP)).LE.(WW-W)) GO TO 114
          GO TO 113
        ENDIF
*
*------CHECK CLEAR PATH
*
1       DO 2 J = 1,NC
         IPSOL(J)=0
2       CONTINUE
        CALL CHECK(IPSOL,IFLAG,X,Y,CR,WX,WY,WR,NC,LIST1,LIST2,IT,
     1            LT,KP,CRMAX,WW,W)
        IF (IFLAG.LE.0) GO TO 6000
        BFEAS=0.0
        GO TO 200
*
*------INITIALIZATION FOR IMPLICIT ENUMERATION
*
6000    DO 7500 K=1,NC
         IBEAS(K)=0
         ICOMP(K)=1
7500    CONTINUE
*
```

```
      JLAST=0
      ITER=0
      NREPC=0
      REP=0
      BFEAS=10.E20
*
*------FORWARD MOVE
*
7000  JLAST=JLAST+1
      IUNDER=JLAST
      IPSOL(JLAST)=1
      REP=REP+A(NFM+JLAST)
*
*------TEST 2
*
      IF (REP.GE.BFEAS) GO TO 7020
*
*------TEST 1
*
      CALL CHECK(IPSOL,IFLAG,X,Y,CR,WX,WY,WR,NC,LIST1,LIST2,IT,
     1          LT,KP,CRMAX,WW,W)
      IF (IFLAG.LE.0) GO TO 7010
      BFEAS=REP
      DO 7030 K=1,NC
       IBEAS(K) = IPSOL(K)
7030  CONTINUE
*
*------TEST 6
*
7020  IF (NREPC.EQ.JLAST) GO TO 70
*
*------BACKWARD MOVE
*
      NREPC=NREPC+IUNDER-JLAST+1
      IPSOL(IUNDER)=0
      JLAST=IUNDER
      REP=REP-A(NFM+JLAST)
      IF (JLAST.LE.1) GO TO 7010
      M=IUNDER-1
      DO 7040 K=1,M
       L=IUNDER-K
       IF (IPSOL(L).EQ.1) THEN
         IUNDER=IUNDER-K
         GO TO 7010
       ENDIF
7040  CONTINUE
7010  IF (JLAST.EQ.NC) GO TO 7050
      M=JLAST+1
      RMIN=10000.0
      DO 7060 K=M,NC
       IF (A(NFM+K).LT.RMIN) RMIN=A(NFM+K)
```

```
7060   CONTINUE
7050   BND=REP+RMIN
*
*------TEST 3
*
       IF ((BND.GE.BFEAS).OR.(JLAST.EQ.NC)) GO TO 7020
*
*------TEST 4
*
       IF (IPSOL(JLAST).EQ.1) GO TO 7000
       DO 7070 K=1,JLAST
        ICOMP(K)=IPSOL(K)
7070   CONTINUE
       CALL CHECK(ICOMP,IFLAG,X,Y,CR,WX,WY,WR,NC,LIST1,LIST2,IT,
     1            LT,KP,CRMAX,WW,W)
*
*------TEST 5
*
       IF (IUNDER.NE.JLAST) THEN
         M=IUNDER+1
         DO 7080 K=M,NC
          ICOMP(K) = 1
7080     CONTINUE
       ENDIF
7081   IF (IFLAG.LE.0) GO TO 7020
       GO TO 7000
70     ATOTAL=ATOTAL+BFEAS
200    CONTINUE
       IF (BFEAS.GT.0.0) THEN
         DO 201 I=1,NC
          IF (IBEAS(I).GT.0) THEN
            NREP=NREP+1
            IREP(NREP)=NFM+I
          ENDIF
201      CONTINUE
       ENDIF
230   CONTINUE
      CUTS=0.
      IF (NREP.NE.0) CUTS=FLOAT(NSUB)
      RETURN
797   FORMAT(1H0,10X,49HNUMBER OF CRATERS IN SUBPROGRAM EXCEEDS 50, NC=
     1 ,I5)
798   FORMAT(1H0,10X,23HSUBPROBLEMS EXCEED 1000)
799   FORMAT (1H0,10X,33HNUMBER OF CRATERS EXCEEDS  50, N= ,I5)
      END
*-----------------------------------------------------------------
```

```
*************************************************************************
* LAST UPDATE 16/2300 JAN 84            FILE:SUBS3.AAP
*************************************************************************
      SUBROUTINE CHECK(IN,IFLAG,X,Y,CR,WX,WY,WR,NC,LIST1,LIST2,IT,
     1                 LT,KP,CRMAX,WW,W)
*
      DIMENSION IN(NC),IT(NC),WX(NC),WY(NC),WR(NC),X(NC),Y(NC),
     1          CR(11,6),LIST1(NC),LIST2(NC),KP(NC)
*
      COMMON/TAXI/NFM,NF,NL
*
      IFLAG=1
      JT=0
      DO 6 JX=1,NC
       IF (IN(JX).LT.1) THEN
        JT=JT+1
        JJ=NFM+JX
        WX(JT)=X(JJ)
        WY(JT)=Y(JJ)
        WR(JT)=CR(LT,KP(JJ))
       ENDIF
6     CONTINUE
      IF (JT.LE.0) RETURN
      IT(1)=-1
      IF ((WY(1)-WR(1)).GE.W) IT(1)=0
      IF ((WY(1)+WR(1)).LE.(WW-W)) IT(1)=1
      IF (IT(1).LT.0) THEN
        IFLAG=0
        RETURN
      ENDIF
      JX=1
10    JX=JX+1
      JXM=JX-1
      IF (JX.GT.JT) RETURN
*
*-----CAN WE GET OVER JX?
*
      IF ((WY(JX)+WR(JX)).LE.(WW-W)) THEN
        IF (IT(JXM).GT.0) THEN
*
*---------DO AN 'OVER-OVER'
*
          XMIN=WX(JX)-WR(JX)-CRMAX-W
*
*---------CHECK BACK
*
          JTEMP=JXM
13        JTEMP=JTEMP-1
          IF (JTEMP.GT.0) THEN
*
*-----------DOES AN 'UNDER' IMPINGE UPON JX?
```

```
*
*-----TRY FOR 'OVER-UNDER'
*
14    JFLAG=2
      CALL BETWN(JXM,JX,JT,JFLAG,WX,WY,WR,LIST1,LIST2,CRMAX,WW,W)
      IF (JFLAG.GT.0) THEN
        IT(JX)=0
        GO TO 10
      ENDIF
*
*-----BACKTRACK
*
500   JI=JX
501   JI=JI-1
      IF (JI.GE.1) THEN
        IF (IT(JI).LE.0) GO TO 501
        JX=JI
        JXM=JX-1
        IF (JXM.GT.0) THEN
          IF ((WY(JX)-WR(JX)).GE.W) GO TO 20
          GO TO 500
        ENDIF
502     IF ((WY(1)-WR(1)).GE.W) THEN
          IT(1)=0
          GO TO 10
        ENDIF
      ENDIF
999   IFLAG=0
      RETURN
      END
*------------------------------------------------------------------
      SUBROUTINE BETWN(JXM,JX,JT,JFLAG,WX,WY,WR,LIST1,LIST2,CRMAX,WW,W)
*
      DIMENSION WX(JT),WY(JT),WR(JT),LIST1(JT),LIST2(JT)
*
      COMMON /TAXI/NFM,NF,NL
*
*-----(JFLAG.LE.1) IMPLIES 'UNDER-OVER'
*     (JFLAG.GE.2) IMPLIES 'OVER-UNDER'
*
      KFLAG=1
      NL1=1
      LIST1(1)=JX
      NLT=1
      K=JX
      XMIN=WX(JX)-WR(JX)-CRMAX-W
*
*-----CONSTRUCT 'LIST1' OF CRATERS BEHIND JX IMPINGING
*     DIRECTLY OR INDIRECTLY UPON IT
*
1     KM=JXM
```

```
*
*-----DETERMINE IF KM IMPINGES UPON K
*
2     IF (WX(KM).GE.XMIN) THEN
        DO 13 IX=1,NL1
         IF (KM.EQ.LIST1(IX)) GO TO 3
13      CONTINUE
        XD=WX(K)-WX(KM)
        YD=WY(K)-WY(KM)
        DIS=SQRT(XD**2+YD**2)-WR(KM)-WR(K)
        IF (DIS.LT.W) THEN
          IF ((JFLAG.LE.1).AND.((WY(KM)+WR(KM)).GT.(WW-W))) GO TO 999
          IF ((JFLAG.GE.2).AND.((WY(KM)-WR(KM)).LT.W)) GO TO 999
*
*---------DETERMINE IF KM IMPINGES UPON JXM
*
          XD=WX(KM)-WX(JXM)
          YD=WY(KM)-WY(JXM)
          DIS=SQRT(XD**2+YD**2)-WR(KM)-WR(JXM)
          IF (DIS.LT.W) GO TO 999
          TEMP=WX(KM)-WR(KM)-CRMAX-W
          IF (XMIN.GT.TEMP) XMIN=TEMP
          NL1=NL1+1
          LIST1(NL1)=KM
        ENDIF
3       KM=KM-1
        IF (KM.GT.0) GO TO 2
      ENDIF
4     NLT=NLT+1
      IF (NLT.LE.NL1) THEN
        K=LIST1(NLT)
        GO TO 1
      ENDIF
*
*-----CONSTRUCT 'LIST2' OF CRATERS AHEAD OF JXM IMPINGING
*     DIRECTLY OR INDIRECTLY UPON IT
*
5     NL2=1
      LIST2(1)=JXM
      NLT=1
      K=JXM
      XMAX=WX(K)+WR(K)+CRMAX+W
*
*-----DETERMINE IF KP IMPINGES UPON K
*
7     KP=JX
8     IF (WX(KP).LE.XMAX) THEN
        DO 19 IX=1,NL2
         IF (KP.EQ.LIST2(IX)) GO TO 9
19      CONTINUE
        XD=WX(K)-WX(KP)
```

```
          YD=WY(K)-WY(KP)
          DIS=SQRT(XD**2+YD**2)-WR(KP)-WR(K)
          IF (DIS.LT.W) THEN
            IF ((JFLAG.LE.1).AND.((WY(KP)-WR(KP)).LT.W)) GO TO 999
            IF ((JFLAG.GE.2).AND.((WY(KP)+WR(KP)).GT.(WW-W))) GO TO 999
*
*---------DETERMINE IF KP IMPINGES UPON JX
*
            XD=WX(KP)-WX(JX)
            YD=WY(KP)-WY(JX)
            DIS=SQRT(XD**2+YD**2)-WR(KP)-WR(JX)
            IF (DIS.LT.W) GO TO 999
            TEMP=WX(KP)+WR(KP)+CRMAX+W
            IF (XMAX.LT.TEMP) XMAX=TEMP
            NL2=NL2+1
            LIST2(NL2)=KP
          ENDIF
9         KP=KP+1
          IF (KP.LE.JT) GO TO 8
        ENDIF
10      NLT=NLT+1
        IF (NLT.LE.NL2) THEN
          K=LIST2(NLT)
          GO TO 7
        ENDIF
*
*-----DETERMINE IF LIST1 IMPINGES UPON LIST2
*
1000    DO 30 K1=1,NL1
         L1=LIST1(K1)
         DO 30 K2=1,NL2
          L2=LIST2(K2)
          DX=WX(L1)-WX(L2)
          DY=WY(L1)-WY(L2)
          DIS=SQRT(DX**2+DY**2)-WR(L1)-WR(L2)
          IF (DIS.LT.W) GO TO 999
30      CONTINUE
        GO TO 2000
999     KFLAG=0
2000    JFLAG=KFLAG
        RETURN
        END
*-----------------------------------------------------------------------
        SUBROUTINE OVLAP(X,Y,CRTAB,LT,NP,X0,Y0,ITL,ITW,KZ,SUM)
*
        COMMON/RAY2/SQUARE(900),CRMAX
*
        DIMENSION X(KZ),Y(KZ),CRTAB(11,6),NP(KZ)
*
*-----INITIALIZE
*
```

```
      DO 10 I=1,ITW
       SQUARE(I)=0.
10    CONTINUE
      SUM=0.
      SUMP=0.
*
*-----FIND FIRST AND LAST VALUES OF X TO CONSIDER
*
      L3=MAX1(1.,(X(1)-CRMAX+1.-X0))
      L2=MIN1(FLOAT(ITL),(X(KZ)+CRMAX+1.-X0))
      J6=1
      M=0
20    L1=L3
*
*-----LOOP-ONE SQUARE AT A TIME IN X
*     L=X VALUE AT TOP OF SQUARE
*
      DO 120 L=L1,L2
       DXP=0.
       J6=J6+M
       M=0
       IF(J6.GT.KZ) RETURN
*
*------IF ALL CRATERS HAVE BEEN CONSIDERED, RETURN
*      LOOP-CRATER BY CRATER...CONSIDER ALL CRATERS WHICH
*      COULD POSSIBLY INTERSECT IN X
*
       DO 90 I=J6,KZ
*
*-------LOCATE LEFT HAND EDGE OF CRATER
*
        NPI=NP(I)
        X1=X(I)-CRTAB(LT,NPI)-X0
        IF (X1.LT.FLOAT(L-1)) GO TO 30
        X2=X(I)-CRMAX-X0
        IF (X2.GE.FLOAT(L)) GO TO 100
        IF (X1.GE.FLOAT(L)) GO TO 90
*
*-------LEFT-HAND EDGE OF CRATER LIES INSIDE LTH SQUARE
*
        DXP=FLOAT(L)-X1
        GO TO 60
*
*-------LEFT HAND EDGE OF CRATER IS BELOW X-SQUARE
*       LOCATE RIGHT HAND EDGE OF CRATER
*
30      X1=X(I)+CRTAB(LT,NPI)-X0
        IF (X1.LE.FLOAT(L-1)) GO TO 40
        IF (X1.GE.FLOAT(L)) GO TO 50
*
*-------RIGHT HAND EDGE OF CRATER LIES INSIDE LTH SQUARE
```

```
*
      DXP=X1-FLOAT(L)+1.
      GO TO 60
*
*-------CRATER I LIES ENTIRELY LEFT OF X-SQUARE...NO NEED TO CONSIDER
*       THIS CRATER ANY MORE
*
40    X3=X(I)+CRMAX-X0
      IF (X3.LE.FLOAT(L-1)) M=M+1
      GO TO 90
50    DXP=1.
*
*-------CRATER INTERSECTS X-SQUARE...CHECK INTERSECTIONS IN Y
*
60    Y1=Y(I)-CRTAB(LT,NPI)-Y0
*
*-------K1=INDEX OF Y-SQUARE CONTAINING LOWER EDGE OF CRATER I
*
      K1=MAX1(1.,Y1+1.)
*
*-------D1=% OF Y-SQUARE OCCUPIED BY CRATER
*
      D1=AMIN1(1.,FLOAT(K1)-Y1)
      SQUARE(K1)=D1*DXP+SQUARE(K1)
      IF (K1.EQ.ITW) GO TO 90
      K1=K1+1
      Y1=Y(I)+CRTAB(LT,NPI)-Y0
      K2=MIN0(ITW,IFIX(Y1))
      IF (K2.EQ.ITW) GO TO 70
      D1=Y1-FLOAT(K2)
*
*-------LOAD SQUARE CONTAINING TOP EDGE OF CRATER I
*
      SQUARE(K2+1)=D1*DXP+SQUARE(K2+1)
*
*-------LOAD INTERMEDIATE Y-SQUARES...D1=1.
*
70    DO 80 J=K1,K2
       SQUARE(J)=SQUARE(J)+DXP
80    CONTINUE
90    CONTINUE
*
*-------COUNT SQUARES THAT ARE AT LEAST HALF-FILLED
*
100   DO 110 J=1,ITW
      IF (SQUARE(J).GE.0.5) SUMP=SUMP+1.
      SQUARE(J)=0.
110   CONTINUE
      SUM=SUM+SUMP
*
*------IF THERE IS A GAP IN X-VALUES, SKIP TO NEXT X-VALUE NEEDED
```

```
*
      IF (DXP.LE.0.) THEN
        IF (M.NE.0) THEN
          J6PM=J6+M
          IF (J6PM.GT.KZ) RETURN
          L3=IFIX(X(J6PM)-CRMAX-X0)+1
          IF (L3.GT.L) GO TO 20
          L3=L+1
          GO TO 20
        ENDIF
      ENDIF
      SUMP=0.
120   CONTINUE
      RETURN
      END
*-----------------------------------------------------------------------
************************************************************************
* LAST UPDATE 14/2200 JAN 84                FILE:SUBS4.AAP
************************************************************************
      SUBROUTINE REPAIR(MXP,KZ,M0,IREPR,CRMAX,II3,NAREA,NCP)
*
      COMMON
     1 ADM(112)        ,GPHT(15)        ,MXPTCH         ,SIGADM(112),
     2 AMIN(3)         ,GPHTAC(15)                      ,SIGARP(3),
     3 APRA(3)         ,GPHTS(15)                       ,SIGASP(3),
     4 APRMIN(3)       ,LNHITS(112)     ,NSAMP1         ,SIGCRT(3),
     5 AREP(3)         ,ICRAT(4)        ,PASS(0:32,6)   ,SIGCTS(27),
     6 AETP(3)         ,ICUT(4,3)       ,PATT(13,34)    ,SIGFIL(27),
     7 COUNTR(112)     ,IHIT(3)         ,RAPF(112)      ,SIGHTS(112),
     8 CRIT(112,2)     ,IPASS(32,2)     ,RCUT(112)      ,SIGNAF(112),
     9 CRTAB(11,6,2)   ,IPAT(12,4)      ,RHIT(112)      ,SMINA(4),
     & DECAR(112)      ,IPCUT(3)        ,SAPR(4)        ,SNAPFL(3),
     1 DSTR(3)                          ,SAPRA(4)       ,TGT(112,5),
     2 ENAPFL(3)       ,IPL(40)         ,SAVE(800,3)    ,XC(3),
     3 GPADAC(15)      ,ISAV(800)       ,SGAPR(4)       ,YC(3),
     4 GPADM(15)       ,ITGT(112,3)     ,SGAPRA(4),
     5 GPADMS(15)      ,I2CUT(4)        ,SGCRAT(4),
     6 GPAREA(15)      ,KH(3)           ,SGMINA(4)
*
      NREP=MIN0(KZ,MXP)
      IF (NREP.EQ.0) RETURN
      K1=0
      K9=KZ
      KTYP=MOD(IREPR,10)
      IF (KTYP.GT.0) THEN
        IF ((SAVE(1,3).LT.FLOAT(II3)).OR.(KTYP.EQ.2)) THEN
          IF ((SAVE(1,3).GT.FLOAT(II3)).AND.(KTYP.EQ.2)) RETURN
          DO 10 J=1,KZ
            IF (SAVE(J,3).GT.FLOAT(II3)) GO TO 20
            IF (SAVE(J,3).LT.FLOAT(II3)) K1=J
10        CONTINUE
```

```
20        K9=J-1
        ENDIF
      ENDIF
30    K9=MIN0(K9,NREP+K1)
      K1=K1+1
      IF (K9.LT.K1) THEN
        IF (KTYP.EQ.2) RETURN
        K1=0
        K9=KZ
        GO TO 30
      ENDIF
40    L=IFIX(SAVE(K1,3)+.01)
      SUMR=KH(L)-K1+1
      IF (NAREA.EQ.0) SUMR=AMIN(L)
      IF (K9.LT.KH(L)) THEN
        SUMR=K9-K1+1
        IF (NAREA.EQ.0) THEN
          IF (SUMR.LE.FLOAT(KH(L)-K9)) THEN
            SUMR=0.
            CALL OVLAP(SAVE(K1,1),SAVE(K1,2),CRTAB,ITGT(L,2),ISAV(K1),
     1                 XC(L)-CRIT(L,1),YC(L),IFIX(CRIT(L,1)),
     2                 IFIX(CRIT(L,2)),K9-K1+1,SUMR)
            GO TO 60
          ENDIF
50        J=K9+1
          SUMR=0.
          CALL OVLAP (SAVE(J,1),SAVE(J,2),CRTAB,ITGT(L,2),ISAV(J),
     1                XC(L)-CRIT(L,1)-2.*CRMAX,YC(L)-2.*CRMAX,
     2                IFIX(CRIT(L,1)+4.*CRMAX),IFIX(CRIT(L,2)+4.*CRMAX),
     3                KH(L)-K9,SUMR)
          SUMR=AMIN(L)-SUMR
        ENDIF
      ENDIF
60    AREP(L)=AREP(L)+SUMR
      SIGARP(L)=SIGARP(L)+SUMR**2
      K5=MIN0(K9,KH(L))+1
      DO 70 J=K5,M0
       J1=K1+J-K5
       SAVE(J1,1)=SAVE(J,1)
       SAVE(J1,2)=SAVE(J,2)
       SAVE(J1,3)=SAVE(J,3)
       ISAV(J1)=ISAV(J)
70    CONTINUE
      K5=K5-K1
      NREP=NREP-K5
      MXP=MXP-K5
      KZ=KZ-K5
      M0=M0-K5
      DO 80 J=L,NCP
       KH(J)=KH(J)-K5
80    CONTINUE
```

```
      IF ((NREP.EQ.0).OR.(KZ.EQ.0)) RETURN
      IF (SAVE(K1,3).NE.FLOAT(L)) THEN
        IF (KTYP.EQ.2) RETURN
        K1=0
        K9=KZ
        GO TO 30
      ENDIF
*
*-----REPAIR HITS ON APPROACH FOR LTH TARGET -- IF APPROPRIATE
*
50    DO 100 J=K1,KZ
       IF (SAVE(J,3).NE.FLOAT(L)) GO TO 110
       IF(J-K1+1.GT.NREP) GO TO 110
       ITGTTP=ITGT(L,2)
       JWPNTP=ISAV(J)
       IF (NAREA.EQ.0) SUMR=SUMR+4.*CRTAB(ITGTTP,JWPNTP,1)**2
100   CONTINUE
110   K5=J-K1
      WRITE(13,150)K1,KZ,M0,J,(SAVE(KK,1),SAVE(KK,2),SAVE(KK,3),KK=1,M0)
      DO 120 J1=J,M0
       KK=K1+J1-J
       SAVE(KK,1)=SAVE(J1,1)
       SAVE(KK,2)=SAVE(J1,2)
       SAVE(KK,3)=SAVE(J1,3)
       ISAV(KK)=ISAV(J1)
120   CONTINUE
      IF (NAREA.EQ.1) SUMR=K5
      WRITE(13,160)K5
      NREP=NREP-K5
      MXF=MXP-K5
      KZ=KZ-K5
      M0=M0-K5
      L=L+1
      IF (L.LE.NCP) THEN
        DO 130 J=L,NCP
         KH(J)=KH(J)-K5
130     CONTINUE
      ENDIF
140   IF ((NREP.EQ.0).OR.(KZ.EQ.0)) RETURN
      IF (KTYP.EQ.2) RETURN
      K1=0
      K9=KZ
      GO TO 30
150   FORMAT (6H K1 = ,I3,6H KZ = ,I3,6H M0 = ,I4,5H J = ,I4.300(/1X,3F:
     12.2))
160   FORMAT (40H NUMBER OF CRATERS FILLED ON APPROACH = ,I6)
      END
*----------------------------------------------------------------------
      SUBROUTINE RESLTS
*
      CHARACTER NAME*4
```

```
*
      DIMENSION PR1(15),PR2(15),PR3(15),PR4(15),PR5(15),PR6(15)
*
      COMMON
     1 ADM(112)       ,GPHT(15)       ,MXPTCH        ,SIGADM(112),
     2 AMIN(3)        ,GPHTAC(15)                    ,SIGARP(3),
     3 APRA(3)        ,GPHTS(15)                     ,SIGASP(3),
     4 APRMIN(3)      ,LNHITS(112)    ,NSAMP1        ,SIGCRT(3),
     5 AREP(3)        ,ICRAT(4)       ,PASS(0:32,6)  ,SIGCTS(27),
     6 ASTP(3)        ,ICUT(4,3)      ,PATT(13,34)   ,SIGFIL(27),
     7 COUNTR(112)    ,IHIT(3)        ,RAPF(112)     ,SIGHTS(112),
     8 CRIT(112,2)    ,IPASS(32,2)    ,RCUT(112)     ,SIGNAF(112),
     9 CRTAB(11,6,2)  ,IPAT(12,4)     ,RHIT(112)     ,SMINA(4),
     & DECAR(112)     ,IPCUT(3)       ,SAPR(4)       ,SNAPFL(3),
     1 DSTR(3)                        ,SAPRA(4)      ,TGT(112,5),
     2 ENAPFL(3)      ,IPL(40)        ,SAVE(800,3)   ,XC(3),
     3 GPADAC(15)     ,ISAV(800)      ,SGAPR(4)      ,YC(3),
     4 GPADM(15)      ,ITGT(112,3)    ,SGAPRA(4),
     5 GPADMS(15)     ,I2CUT(4)       ,SGCRAT(4),
     6 GPAREA(15)     ,KH(3)          ,SGMINA(4)
*
      COMMON/END/NSAMP,NELT,NTGPS,NCP,CRMIN,APPRCW,NAREA
*
      COMMON/JOHN/NFLAG1,NFLAG2,NMAX,NSAMPR,ZALPH,ERROR,NSAMP2,NFLAG3
*
      NAME='  NO'
      SAMPL=1./FLOAT(NSAMP)
      SAMPO=FLOAT(NSAMP-1)
      DO 10 I=1,NTGPS
       GPAREA(I)=0.
       GPADM(I)=0.
       GPHT(I)=0.
10    CONTINUE
      CT=0.
      DO 30 L=1,NELT
       IF (COUNTR(L).GT.CT) THEN
         LCOUNT=L
         CT=COUNTR(L)
       ENDIF
       ITGTGP=ITGT(L,3)
       GPHT(ITGTGP)=GPHT(ITGTGP)+COUNTR(L)
       GPADM(ITGTGP)=GPADM(ITGTGP)+ADM(L)
       GPAREA(ITGTGP)=GPAREA(ITGTGP)+TGT(L,4)*TGT(L,5)
30    CONTINUE
      CONF90=SIGHTS(LCOUNT)-SAMPL*COUNTR(LCOUNT)**2
      CONF90=SQRT(CONF90/SAMPO)
      CONF90=2.576*CONF90*SQRT(SAMPL)
      WRITE(13,240)NSAMP,CONF90,LCOUNT
      CONF90=1.645*CONF90/2.576
      WRITE(13,250)CONF90
      IF (NFLAG3.EQ.1.AND.NSAMP.GE.200) WRITE(13,450)
```

```
      IB=0
40    IA=IB+1
      IB=MIN0(IA+14,NELT)
      KM=IB-IA+1
      WRITE(13,260)(K,K=IA,IB)
      DO 50 K=1,KM
       L=K+IA-1
       PR1(K)=SAMPL*COUNTR(L)
       PR2(K)=SIGHTS(L)-SAMPL*COUNTR(L)**2
       PR2(K)=SQRT(PR2(K)/SAMPO)
       PR3(K)=SAMPL*ADM(L)
       PR4(K)=SIGADM(L)-SAMPL*ADM(L)**2
       PR4(K)=SQRT(PR4(K)/SAMPO)
50    CONTINUE
      WRITE(13,270)(PR1(K),K=1,KM)
      WRITE(13,280)(PR2(K),K=1,KM)
      IF (NAREA.EQ.0) WRITE(13,290)(PR3(K),K=1,KM)
      IF (NAREA.EQ.0) WRITE(13,300)(PR4(K),K=1,KM)
      WRITE(13,310)(ITGT(K,3),K=IA,IB)
      IF (IB.LT.NELT) GO TO 40
      WRITE(13,320)
      IB=0
60    IA=IB+1
      IB=MIN0(IA+14,NTGPS)
      KM=IB-IA+1
      WRITE(13,330)(K,K=IA,IB)
      DO 70 K=1,KM
       L=K+IA-1
       PR1(K)=GPHTS(L)-SAMPL*GPHT(L)**2
       PR1(K)=SQRT(PR1(K)/SAMPO)
       GPHT(L)=SAMPL*GPHT(L)
       PR2(K)=GPADMS(L)-SAMPL*GPADM(L)**2
       PR2(K)=SQRT(PR2(K)/SAMPO)
       GPADM(L)=SAMPL*GPADM(L)
       GPAREA(L)=GPADM(L)/GPAREA(L)
70    CONTINUE
      WRITE(13,270)(GPHT(K),K=IA,IB)
      WRITE(13,280)(PR1(K),K=1,KM)
      IF (NAREA.EQ.0) THEN
        WRITE(13,290)(GPADM(K),K=IA,IB)
        WRITE(13,300)(PR2(K),K=1,KM)
        WRITE(13,340)(GPAREA(K),K=IA,IB)
      ENDIF
80    IF (IB.LT.NTGPS) GO TO 60
      IF (NCP.GT.0) THEN
        WRITE(13,350)NAME
        DO 120 L=1,NCP
         PR1(1)=SAMPL*RCUT(L)
         PR1(2)=SQRT((PR1(1)-PR1(1)**2)*SAMPL)
         PR1(4)=SIGCRT(L)-SAMPL*RHIT(L)**2
         PR1(4)=SQRT(PR1(4)/SAMPO)
```

```
      PR1(3)=SAMPL*RHIT(L)
      PR1(5)=SAMPL*ASTP(L)
      PR1(6)=SIGASP(L)-SAMPL*ASTP(L)**2
      PR1(6)=SQRT(PR1(6)/SAMPO)
      PR1(7)=SAMPL*AREP(L)
      PR1(8)=SIGARP(L)-SAMPL*AREP(L)**2
      PR1(8)=SQRT(PR1(8)/SAMPO)
      PR1(12)=SIGNAF(L)-SAMPL*RAPF(L)**2
      PR1(12)=SQRT(PR1(12)/SAMPO)
      PR1(11)=SAMPL*RAPF(L)
      PR1(10)=SNAPFL(L)-SAMPL*ENAPFL(L)**2
      PR1(10)=SQRT(PR1(10)/SAMPO)
      PR1(9)=SAMPL*ENAPFL(L)
      IF (NAREA.EQ.1) THEN
        PR1(5)=1.E20
        PR1(6)=1.E20
      ENDIF
90    IF (MXPTCH.EQ.0) THEN
        PR1(7)=1.E20
        PR1(8)=1.E20
      ENDIF
100   IF (APPRCW.LT.1.) THEN
        PR1(9)=1.E20
        PR1(10)=1.E20
        PR1(11)=1.E20
        PR1(12)=1.E20
      ENDIF
110   WRITE(13,360)L,CRIT(L,1),CRIT(L,2),(PR1(K),K=1,12)
120  CONTINUE
     IF (NCP.GT.1) THEN
       WRITE(13,370)
       IEL1=1
       IEL2=2
       NCP1=NCP+1
       DO 170 KJ=1,NCP1
        KK=4-KJ
        DO 130 L=1,3
         DSTR(L)=SAMPL*FLOAT(ICUT(KJ,L))
         IF (KK.GT.0) DSTR(KK)=1.E20
130     CONTINUE
        PR1(1)=SAMPL*FLOAT(I2CUT(KJ))
        PR1(2)=SQRT(SAMPL*(PR1(1)-PR1(1)**2))
        PR1(4)=FLOAT(ICRAT(KJ))
        PR1(3)=SAMPL*PR1(4)
        PR1(4)=SGCRAT(KJ)-SAMPL*PR1(4)**2
        PR1(4)=SQRT(PR1(4)/SAMPO)
        PR1(5)=SAMPL*SMINA(KJ)
        PR1(6)=SGMINA(KJ)-SAMPL*SMINA(KJ)**2
        PR1(6)=SQRT(PR1(6)/SAMPO)
        PR1(7)=SAMPL*SAPR(KJ)
        PR1(8)=SGAPR(KJ)-SAMPL*SAPR(KJ)**2
```

```
      PR1(8)=SQRT(PR1(8)/SAMPO)
      PR1(9)=SAMPL*SAPRA(KJ)
      PR1(10)=SGAPRA(KJ)-SAMPL*SAPRA(KJ)**2
      PR1(10)=SQRT(PR1(10)/SAMPO)
      IF (NAREA.EQ.1) THEN
        PR1(5)=1.E20
        PR1(6)=1.E20
      ENDIF
140   IF (APPRCH.LT.1.) THEN
        PR1(7)=1.E20
        PR1(8)=1.E20
        PR1(9)=1.E20
        PR1(10)=1.E20
      ENDIF
150   IF (KJ.LT.4) THEN
        IF (KJ.EQ.2) IEL2=3
        IF (KJ.EQ.3) IEL1=2
        WRITE(13,480)IEL1,IEL2,CRIT(IEL1,1),CRIT(IEL2,2),(PR1(K),
     1                K=1,6),(DSTR(K),K=1,3),(PR1(K),K=7,10)
        IF (NCP.EQ.3) GO TO 170
        GO TO 180
      ENDIF
160   WRITE(13,380)CRIT(1,1),CRIT(1,2),(PR1(K),K=1,6),(DSTR(K),
     1                K=1,3),(PR1(K),K=7,10)
170     CONTINUE
      ENDIF
    ENDIF
180 IF (LV.GT.0) WRITE(13,390)
    LV=0
    DO 190 L=1,NELT
     IF ((ITGT(L,1).EQ.1).AND.(CRIT(L,1).LT.1.)) THEN
       LV=LV+1
       IPL(LV)=L
     ENDIF
190 CONTINUE
    IF (LV.GT.0) THEN
      IB=0
200   IA=IB+1
      IB=MIN0(IA+14,LV)
      KM=IB-IA+1
      WRITE(13,400)(IPL(K),K=IA,IB)
*-------NON-ANSI STANDARD SUBSCRIPTS MAY REQUIRE ADJUSTMENT.
      WRITE(13,410)(TGT(IPL(K),5),K=IA,IB)
      WRITE(13,420)(CRIT(IPL(K),2),K=IA,IB)
      DO 210 K=1,KM
       L=K+IA-1
       IPLL=IPL(L)
       PR1(K)=SAMPL*RCUT(IPLL)
       PR2(K)=SIGCTS(L)-SAMPL*RCUT(IPLL)**2
       PR2(K)=SQRT(PR2(K)/SAMPO)
       PR3(K)=SAMPL*RHIT(IPLL)
```

```
         PR4(K)=SIGFIL(L)-SAMPL*RHIT(IPLL)**2
         PR4(K)=SQRT(PR4(K)/SAMPO)
         PR6(K)=SAMPL*RAPF(IPLL)
         PR5(K)=SIGNAF(IPLL)-SAMPL*RAPF(IPLL)**2
         PR5(K)=SQRT(PR5(K)/SAMPO)
210     CONTINUE
        WRITE(13,430)(PR1(K),K=1,KM)
        WRITE(13,440)(PR2(K),K=1,KM)
        WRITE(13,450)(PR3(K),K=1,KM)
        WRITE(13,440)(PR4(K),K=1,KM)
        IF (NAREA.EQ.0) THEN
          WRITE(13,460)(PR6(K),K=1,KM)
          WRITE(13,470)(PR5(K),K=1,KM)
        ENDIF
220     IF (IB.LT.LV) GO TO 200
      ENDIF
      RETURN
240   FORMAT(1X,'NSAMP =',I5,5X,'CONF INTERVAL FOR 99% LEVEL =',F7.3,
     12X,'FOR TGT ELT =',I5)
250   FORMAT(18X,29HCONF INTERVAL FOR 90% LEVEL =,F7.3)
260   FORMAT(1H0,1X,11HTGT ELEMENT,15I8)
270   FORMAT(1X,12HEXP NO. HITS,15F8.3)
280   FORMAT(8X,5HSIGMA,15F8.3)
290   FORMAT(1X,12HEXP AREA DAM,15F8.0)
300   FORMAT(8X,5HSIGMA,15F8.0)
310   FORMAT(2X,11HTGT GP. NO.,15I8)
320   FORMAT(1H0,13HTARGET GROUPS)
330   FORMAT(1H0,1X,11HTGT GP. NO.,15I8)
340   FORMAT(1X,12HEXP PER. DAM,15F8.3)
350   FORMAT (1H0,4X,30HFOR RUNWAYS AND MAJOR TAXIWAYS,/8X,3HTGT,4X,3HMC
     1L,2X,3HMCW,3X,4HPROB,2X,5HSIGMA,2X,6HEXP NO,3X,5HSIGMA,3X,8HEXP AR
     2EA,3X,5HSIGMA,3X,4HEXP ,A4,3X,5HSIGMA,3X,8HEXP APPR,3X,5HSIGMA,3X,
     38HEXP APPR,3X,5HSIGMA,/8X,3HELT,16X,3HCUT,8X,7HCRATERS,15X,4HFILL,
     413X,6HFILLED,12X,7HNO CRAT,15X,4HFILL)
360   FORMAT(8X,I3,F7.0,F5.0,2F7.3,2F8.3,4X,F7.0,1X,F7.0,4X,F7.0,1X,F7.
     10,3X,F8.3,1X,F8.3,3X,F8.0,1X,F8.0)
370   FORMAT (1H0,4X,29HCOMBINED PROBABILITIES OF CUT,/77X,12HDISTRIBUTI
     1ON,/75X,16HMINIMUM  CRATERS,/8X,3HTGT,4X,3HMCL,2X,3HMCW,3X,4HPROB,
     22X,5HSIGMA,2X,6HEXP NO,3X,5HSIGMA,3X,8HEXP AREA,3X,5HSIGMA,4X,3(3H
     3ELT,3X),8HEXP APPR,3X,5HSIGMA,3X,8HEXP APPR,3X,5HSIGMA,/7X,4HELTS,
     416X,3HCUT,8X,7HCRATERS,15X,4HFILL,13X,1H1,5X,1H2,5X,1H3,5X,7HNO CR
     5AT,15X,4HFILL)
380   FORMAT(6X,5H1&2&3,F7.0,F5.0,2F7.3,2F8.3,4X,F7.0,1X,F7.0,3X,3(F5.3
     1,1X),1X,2F8.3,3X,2F8.0)
390   FORMAT(1H0,4X,18HFOR MINOR TAXIWAYS)
400   FORMAT(1H0,13X,14HTARGET ELEMENT,15I7)
410   FORMAT(16X,12HTARGET WIDTH,15F7.0)
420   FORMAT(9X,19HMINIMUM CLEAR WIDTH,15F7.0)
430   FORMAT(5X,23HEXPECTED NUMBER OF CUTS,15F7.3)
440   FORMAT(23X,5HSIGMA,15F7.3)
450   FORMAT(4X,24HEXPECTED CRATERS TO FILL,15F7.3)
```

```
460   FORMAT(7X,21HEXPECTED AREA TO FILL,15F7.0)
470   FORMAT(23X,5HSIGMA,15F7.0)
480   FORMAT(8X,I1,1H&,I1,F7.0,F5.0,2F7.3,2F8.3,4X,F7.0,1X,F7.0,3X,3(F5.
     13,1X),1X,2F8.3,3X,2F8.0)
490   FORMAT(1H ,'NSAMP LIMITED TO LEAST OF VALUE INPUT OR NUMBER NEEDED
     = TO GIVE SPECIFIED QUALITY TO PROBABILITY OF CUT.')
      END
*---------------------------------------------------------------------
      SUBROUTINE NCOMP
*
*-----THIS ROUTINE IS ENTERED TO CALCULATE THE MINIMUM SAMPLE SIZE
*     OF MONTE CARLO ITERATIONS TO GIVE A SPECIFIC CONFIDENCE LEVEL
*     AND INTERVAL FOR THE PROBABILITY OF CUTTING A TAKEOFF
*     SURFACE. IT CANNOT BE ENTERED UNLESS NFLAG3 IS SET IN MAIN
*     PROGRAM AND NSAMP SPECIFIED AS GREATER THAN 200.
*
      COMMON
     1 ADM(112)      ,GPHT(15)      ,MXPTCH         ,SIGADM(112),
     2 AMIN(3)       ,GPHTAC(15)                    ,SIGARP(3),
     3 APRA(3)       ,GPHTS(15)                     ,SIGASP(3),
     4 APRMIN(3)     ,LNHITS(112)   ,NSAMP1         ,SIGCRT(3),
     5 AREP(3)       ,ICRAT(4)      ,PASS(0:32,6)   ,SIGCTS(27),
     6 ASTP(3)       ,ICUT(4,3)     ,PATT(13,34)    ,SIGFIL(27),
     7 COUNTR(112)   ,INIT(3)       ,RAPF(112)      ,SIGHTS(112),
     8 CRIT(112,2)   ,IPASS(32,2)   ,RCUT(112)      ,SIGNAF(112),
     9 CRTAB(11,6,2) ,IPAT(12,4)    ,RHIT(112)      ,SMINA(4),
     & DECAR(112)    ,IPCUT(3)      ,SAPR(4)        ,SNAPFL(3),
     1 DSTR(3)                      ,SAPRA(4)       ,TGT(112,5),
     2 ENAPFL(3)     ,IPL(40)       ,SAVE(800,3)    ,XC(3),
     3 GPADAC(15)    ,ISAV(800)     ,SGAPR(4)       ,YC(3),
     4 GPADM(15)     ,ITGT(112,3)   ,SGAPRA(4),
     5 GPADMS(15)    ,I2CUT(4)      ,SGCRAT(4),
     6 GPAREA(15)    ,KH(3)         ,SGMINA(4)
*
      COMMON/END/NSAMP2,NELT,NTGPS,NCP,CRMIN,APPRCH,NAREA
*
      COMMON/JOHN/NFLAG1,NFLAG2,NMAX,NSAMPR,ZALPH,ERROR,NSAMP,NFLAG3
*
      DIMENSION PR(3)
*
*-----CALCULATE AND STORE IN A MATRIX THE PROBABILITY OF CUT FOR
*     EACH TARGET ELEMENT, USING THIS PATTERN.
*
      IF (NCP.GE.1) THEN
        IF (ZALPH.LT.1.645) ZALPH=1.645
        IF ((ERROR.GT.0.05).OR.(ERROR.LT.0.0001)) ERROR=0.05
        DO 10 J=1,NCP
         PR(J)=RCUT(J)/FLOAT(NSAMP)
10      CONTINUE
*
*-------INITIALIZE A LOOP TO FIND THAT PROBABILITY OF CUT CLOSEST
```

```
*      TO 0.5. THIS MAXIMIZES REQUIRED SAMPLE SIZE FOR WORST CASE
*      TARGET ELEMENT AND ATTACK.
*
       SMALL=ABS(PR(1)-0.5)
       IX=1
       JX=1
*
*-------LOOP TO FIND PROBABILITY OF CUT CLOSEST TO 0.5
*      AND RECORD IT AS PKNUM.
*
       DO 20 J=1,NCP
        SMALL1=ABS(PR(J)-0.5)
        IF (SMALL1.LT.SMALL) THEN
          IX=I
          JX=J
          SMALL=SMALL1
        ENDIF
20     CONTINUE
       PKNUM=PR(JX)
       NUM=0
*
*-------IF PKNUM IS VERY CLOSE TO ZERO OR ONE, THE STATISTICS
*      COLLAPSE MONTE CARLO ITERATIONS TO A VERY SMALL NUMBER.
*      THEN CALCULATION OF ADDITIONAL ITERATIONS TO RUN OR
*      RETURN TO THE MONTE CARLO LOOP SHOULD NOT BE COMPLETED.
*      THIS ACCOMPLISHED BY SETTING NFLAG1.
*
       IF ((PKNUM.GT.0.0009).AND.(PKNUM.LT.0.9995)) THEN
*
*---------CALCULATE TOTAL SAMPLE SIZE TO ASSURE CONFIDENCE LEVEL
*        AND ERROR INTERVAL.
*
         SSIZE=PKNUM*(1.-PKNUM)*((ZALPH/ERROR)**2.)
         NUM=SSIZE+1.
*
*---------TEST IF MORE ITERATIONS REQUIRED, SETTING APPROPRIATE FLAGS
*        WHETHER TO RETURN TO THE MONTE CARLO LOOP. IF SO, SET LOWER
*        AND UPPER MONTE CARLO LOOP LIMITS.
*
         IF (NUM.LE.NSAMP) THEN
           NFLAG1=1
           RETURN
         ELSE
           NSAMPR=NSAMP+1
           NFLAG2=1
           IF (NUM.LT.NMAX) THEN
             NSAMP=NUM
           ELSE
             NSAMP=NMAX
           ENDIF
           RETURN
```

```
          ENDIF
        ENDIF
      ENDIF
95    NFLAG1=1
      RETURN
      END
```

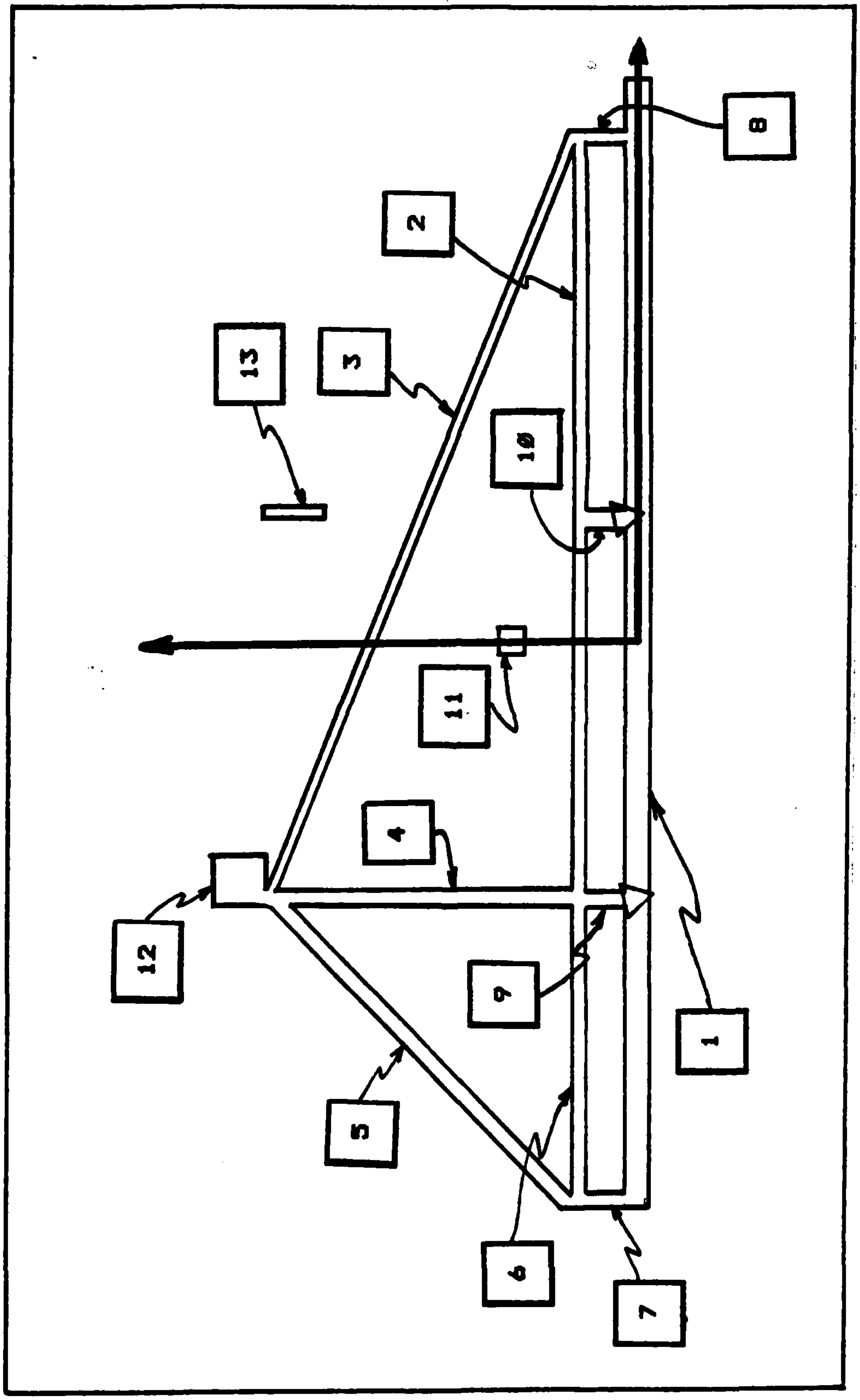


Figure F.1 Sketch of Sample Airfield. (Aimpoints enclosed in △.)

DATA INPUT ECHO

```
987654321
      250        100
        0      .0500      1.6450
       13          4        30.0          0
   0.0000                 0.0000      0.0000     9000.0000     200.0000     1     1     1
4000.0000                50.0000
1000.0000               450.0000      0.0000     6000.0000     100.0000     1     1     1
4000.0000                50.0000
1000.0000              1250.0000      2.7571     6550.0000     100.0000     1     2     2
   0.0000                30.0000
-2050.0000             1750.0000      1.5705     2500.0000     100.0000     1     2     2
   0.0000                30.0000
-3250.0000             1750.0000       .7853     3500.0000     100.0000     1     2     2
   0.0000                30.0000
-3250.0000              450.0000      0.0000     2500.0000     100.0000     1     2     2
   0.0000                30.0000
-4450.0000              250.0000      1.5705      300.0000     100.0000     1     2     3
   0.0000                30.0000
 4050.0000              250.0000      1.5705      300.0000     100.0000     1     2     3
   0.0000                30.0000
-2050.0000              250.0000      1.5705      300.0000     100.0000     1     2     3
   0.0000                30.0000
 1000.0000              250.0000      1.5705      300.0000     100.0000     1     2     3
   0.0000                30.0000
   0.0000              1000.0000      0.0000      200.0000     200.0000     0     3     4
-1900.0000             3200.0000      0.0000      400.0000     400.0000     0     3     4
 1050.0000             2750.0000      1.5705      500.0000     100.0000     0     3     4
```

```
    2       8
    1
12    1    1    0  370.000  185.000   80.000   30.000    0.000    0.000    0.000    0.000    1.000    0.000
  -220.0000      14.0000
  -180.0000     -14.0000
  -140.0000      14.0000
  -100.0000     -14.0000
   -60.0000      14.0000
   -20.0000     -14.0000
    20.0000      14.0000
    60.0000     -14.0000
   100.0000      14.0000
   140.0000     -14.0000
   180.0000      14.0000
   220.0000     -14.0000
    3       1
    15.9
    21.3
    31.9
    10.6
    15.9
    21.3
    2       1       6
 -2000.0000       0.0000        .6981       1.0000       1.0000       1       0
 -2000.0000       0.0000        .6981       1.0000       1.0000       1       0
 -2000.0000       0.0000        .6981       1.0000       1.0000       1       0
  1000.0000     100.0000        .6981       1.0000       1.0000       1       0
  1000.0000     100.0000        .6981       1.0000       1.0000       1       0
  1000.0000     100.0000        .6981       1.0000       1.0000       1       0
```

1 INPUT FILE: LLH1 OUTPUT FILE: LLHA

NSAMP = 100 CONF INTERVAL FOR 99% LEVEL = 1.875 FOR TGT ELT = 1
CONF INTERVAL FOR 90% LEVEL = 1.197

0 TGT ELEMENT	1	2	3	4	5	6	7	8	9	10	11	12	13
EXP NO. HITS	20.630	4.890	.130	.170	0.000	.440	0.000	0.000	1.130	1.690	0.000	0.000	0.000
SIGMA	7.270	3.502	.562	.726	0.000	1.183	0.000	0.000	1.709	2.131	0.000	0.000	0.000
EXP AREA DAM	16780.	3515.	151.	176.	0.	475.	0.	0.	1297.	1794.	0.	0.	0.
SIGMA	6077.	2617.	608.	812.	0.	1290.	0.	0.	2045.	2449.	0.	0.	0.
TGT GP. NO.	1	1	2	2	2	2	3	3	3	3	4	4	4

0TARGET GROUPS

0 TGT GP. NO.	1	2	3	4
EXP NO. HITS	25.520	.740	2.820	0.000
SIGMA	6.046	1.824	2.709	0.000
EXP AREA DAM	20295.	802.	3090.	0.
SIGMA	5141.	2042.	3134.	0.
EXP PER. DAM	.008	.001	.026	0.000

0 FOR RUNWAYS AND MAJOR TAXIWAYS

TGT ELT	MCL	MCW	PROB CUT	SIGMA	EXP NO CRATERS	SIGMA	EXP AREA FILL	SIGMA	EXP NO FILLED	SIGMA	EXP APPR NO CRAT	SIGMA	EXP APPR FILL	SIGMA
1	4000.	50.	.440	.050	.620	.789	418.	575.	*******	*******	1.920	1.587	863.	713.
2	4000.	50.	.520	.050	1.260	1.685	772.	1103.	*******	*******	.370	.691	166.	311.

0 COMBINED PROBABILITIES OF CUT

TGT ELTS	MCL	MCW	PROB CUT	SIGMA	EXP NO CRATERS	SIGMA	EXP AREA FILL	SIGMA	DISTRIBUTION MINIMUM CRATERS ELT 1	ELT 2	ELT 3	EXP APPR NO CRAT	SIGMA	EXP APPR FILL	SIGMA
1&2	4000.	50.	.190	.039	.200	.426	136.	326.	.170	.020	*****	.230	.601	103.	270.

0 TARGET ELEMENT	3	4	5	6	7	8	9	10
TARGET WIDTH	100.	100.	100.	100.	100.	100.	100.	100.
MINIMUM CLEAR WIDTH	30.	30.	30.	30.	30.	30.	30.	30.
EXPECTED NUMBER OF CUTS	0.000	0.000	0.000	.030	0.000	0.000	.200	.180
SIGMA	0.000	0.000	0.000	.223	0.000	0.000	.603	.539
EXPECTED CRATERS TO FILL	0.000	0.000	0.000	.020	0.000	0.000	.130	.140
SIGMA	0.000	0.000	0.000	.141	0.000	0.000	.367	.403
EXPECTED AREA TO FILL	0.	0.	0.	20.	0.	0.	131.	142.
SIGMA	0.	0.	0.	142.	0.	0.	371.	407.

```
NSAMP =  200     CONF INTERVAL FOR 99% LEVEL =  1.433  FOR TGT ELT =    1
                 CONF INTERVAL FOR 90% LEVEL =   .915
0 TGT ELEMENT       1      2      3      4      5      6      7      8      9     10     11     12     13
 EXP NO. HITS  21.210  4.965   .085   .175  0.000   .665  0.000  0.000  1.180  1.575  0.000  0.000  0.000
        SIGMA   7.067  3.698   .434   .740  0.000  1.595  0.000  0.000  1.787  2.060  0.000  0.000  0.000
 EXP AREA DAM  17160.  3464.   102.   184.     0.   722.     0.     0.  1308.  1670.     0.     0.     0.
        SIGMA   6424.  2629.   549.   825.     0.  1822.     0.     0.  1999.  2272.     0.     0.     0.
  TGT GP. NO.       1      1      2      2      2      2      3      3      3      3      4      4      4
0TARGET GROUPS
0 TGT GP. NO.       1      2      3      4
 EXP NO. HITS  26.175   .925  2.755  0.003
        SIGMA   6.589  2.096  2.772  0.000
 EXP AREA DAM  20624.  1009.  2977.     0.
        SIGMA   5496.  2300.  3029.     0.
 EXP PER. DAM    .009   .001   .025  0.000
0    FOR RUNWAYS AND MAJOR TAXIWAYS
        TGT    MCL  MCW   PROB  SIGMA  EXP NO   SIGMA   EXP AREA   SIGMA   EXP   NO   SIGMA   EXP APPR   SIGMA   EXP APPR   SIGMA
        ELT                CUT         CRATERS              FILL             FILLED           NO CRAT                 FILL
          1  4000.  50.   .395   .035    .590    .840       382.     583.    ******* *******     2.075    1.616       933.     726.
          2  4000.  50.   .525   .035   1.265   1.640       815.    1087.    ******* *******      .435     .860       196.     386.
0    COMBINED PROBABILITIES OF CUT
                                                                            DISTRIBUTION
                                                                          MINIMUM  CRATERS
        TGT    MCL  MCW   PROB  SIGMA  EXP NO   SIGMA   EXP AREA   SIGMA   ELT   ELT   ELT  EXP APPR   SIGMA   EXP APPR   SIGMA
       ELTS                CUT         CRATERS              FILL            1     2     3    NO CRAT                 FILL
        1&2  4000.  50.   .160   .026    .180    .434       113.     296.  .135  .025 *****     .230     .692       103.     311.
0    FOR MINOR TAXIWAYS
0             TARGET ELEMENT       3      4      5      6      7      8      9     10
                TARGET WIDTH    100.   100.   100.   100.   100.   100.   100.   100.
         MINIMUM CLEAR WIDTH     30.    30.    30.    30.    30.    30.    30.    30.
    EXPECTED NUMBER OF CUTS    0.000  0.000  0.000   .045  0.000  0.000   .160   .125
                       SIGMA   0.000  0.000  0.000   .289  0.000  0.000   .535   .436
   EXPECTED CRATERS TO FILL    0.000  0.000  0.000   .030  0.000  0.000   .110   .120
                       SIGMA   0.000  0.000  0.000   .171  0.000  0.000   .344   .420
      EXPECTED AREA TO FILL       0.     0.     0.    30.     0.     0.   111.   121.
                       SIGMA      0.     0.     0.   173.     0.     0.   348.   425.
```

NSAMP = 250 CONF INTERVAL FOR 99% LEVEL = 1.250 FOR TGT ELT = 1
CONF INTERVAL FOR 90% LEVEL = .798

# TGT ELEMENT	1	2	3	4	5	6	7	8	9	10	11	12	13
EXP NO. HITS	21.128	4.984	.076	.152	0.000	.624	0.000	0.000	1.212	1.544	0.000	0.000	0.000
SIGMA	7.674	3.737	.399	.689	0.000	1.571	0.000	0.000	1.788	2.036	0.000	0.000	0.000
EXP AREA DAM	17121.	3458.	94.	156.	0.	680.	0.	0.	1333.	1638.	0.	0.	0.
SIGMA	6319.	2652.	509.	751.	0.	1797.	0.	0.	1992.	2264.	0.	0.	0.
TGT GP. NO.	1	1	2	2	2	2	3	3	3	3	4	4	4

0TARGET GROUPS

# TGT GP. NO.	1	2	3	4
EXP NO. HITS	26.112	.852	2.756	0.003
SIGMA	6.451	2.008	2.796	0.000
EXP AREA DAM	20579.	929.	2971.	0.
SIGMA	5432.	2283.	3063.	0.
EXP PER. DAM	.009	.001	.025	0.003

0 FOR RUNWAYS AND MAJOR TAXIWAYS

TGT ELT	MCL	MCW	PROB CUT	SIGMA	EXP NO CRATERS	SIGMA	EXP AREA FILL	SIGMA	EXP NO FILLED	SIGMA	EXP APPR NO CRAT	SIGMA	EXP APPR FILL	SIGMA
1	4000.	50.	.408	.031	.584	.808	384.	575.	*******	*******	2.048	1.580	920.	710.
2	4000.	50.	.520	.032	1.244	1.621	817.	1105.	*******	*******	.440	.859	198.	386.

0 COMBINED PROBABILITIES OF CUT

TGT ELTS	MCL	MCW	PROB CUT	SIGMA	EXP NO CRATERS	SIGMA	EXP AREA FILL	SIGMA	DISTRIBUTION MINIMUM CRATERS ELT 1	ELT 2	ELT 3	EXP APPR NO CRAT	SIGMA	EXP APPR FILL	SIGMA
1&2	4000.	50.	.176	.024	.192	.433	122.	300.	.144	.032	*****	.248	.730	111.	328.

0 FOR MINOR TAXIWAYS

0 TARGET ELEMENT	3	4	5	6	7	8	9	10
TARGET WIDTH	100.	100.	100.	100.	100.	100.	100.	100.
MINIMUM CLEAR WIDTH	30.	30.	30.	30.	30.	30.	30.	30.
EXPECTED NUMBER OF CUTS	0.000	0.000	0.000	.048	0.000	0.000	.144	.136
SIGMA	0.000	0.000	0.000	.293	0.000	0.000	.502	.436
EXPECTED CRATERS TO FILL	0.000	0.000	0.000	.036	0.000	0.000	.100	.128
SIGMA	0.000	0.000	0.000	.207	0.000	0.000	.326	.410
EXPECTED AREA TO FILL	0.	0.	0.	36.	0.	0.	101.	129.
SIGMA	0.	0.	0.	209.	0.	0.	330.	415.

END

FILMED

6-84

DTIC