2

LA-SUB--94-141

QUANTICS incorporated

801 Springdale Drive, Exton, PA 19341 • (610) 993-9993

UNCLASSIFIED

Prototype Integration of the Joint Munitions Assessment and Planning Model with the OSD Threat Methodology

June 1994

DISCLAIMER

Prepared for:

Los Alamos National Laboratory
Military Systems Analysis
and Simulation Group (A-5)
Los Alamos, New Mexico 87545

QM 29:94
Prepared by:

Roger Y. S. Lynn
J. J. Bolmarcich

Under Contract No.
9-X53-1599F-1

UNCLASSIFIED

"quantitative guidance for decision making"

QUANTICS incorporated
801 Springdale Drive, Exton, PA 19341 • (610) 993-9993

UNCLASSIFIED

30 June 1994

MEMORANDUM (QM 29:94)

Contract No. 9-X53-1599F-1

To: Los Alamos National Laboratory
Military Systems Analysis and Simulation Group (A-5)
Los Alamos, New Mexico 87545
Attn: Mr. Douglas P. Anson

From: Roger Y. S. Lynn and J. J. Bolmarcich

Subject: Prototype Integration of the Joint Munitions Assessment and Planning Model with the OSD Threat Methodology

The purpose of this Memorandum is to propose a prototype procedure which the Office of Munitions might employ to exercise, in a supportive joint fashion, two of its High Level Conventional Munitions Models, namely, the OSD Threat Methodology and the Joint Munitions Assessment and Planning (JMAP) model.

Reference [a] describes the OSD Threat Methodology as comprising two parts: First, the OSD Threat Split Model, running within the R:BASE database manager, has the job of estimating how all types of enemy targets in the scenario will likely end up being killed by all types of "friendly" forces put there to combat them. And second, the OSD Threat Model (described in detail in reference [b]), running as a stand-alone application, estimates the munitions stockpiles needed to let the war turn out to match the OSD Threat Split Model estimates. Actually, the Threat Model prudently overestimates war reserve stockpiles by determining munitions both for expenditures and for safety stocks. These safety stocks take the form of the non-expended portions of munitions loadouts which are needed to assure that rounds are at the right place at the right time. Both Models consider time only in the aggregate; they can be considered independent of time in comparison to dynamic combat models.

JMAP, as described in references [c], [d], and [e], provides a means to optimize the allocation of munitions and platforms (shooters) to targets consistent with some user designated goal priorities. JMAP is a means of trading-off some munitions for others while applying hard constraints to munitions available and imposing specific goals on munition usage, target kill, shooter attrition, and costs. JMAP is basically designed to show how to kill the most scenario targets while consuming the fewest dollars from the existing investments in stocks of munitions and shooters.

The joint application of JMAP and the OSD Threat Methodology provides a tool to optimize munitions stockpiles. From JMAP's point of view, the Threat Methodology can help it work more realistically with stockpile limits because in warfare not all of a stockpile can be expended at the

enemy and the optimal shooter/muntion combination may not acquire each target. The Threat Model brings to JMAP the ability to consider the cost of all munitions needed to support the scenario, the expenditures plus safety stocks; JMAP can thus be made to operate on whole stockpiles instead of just on expenditures. From the Threat Methodology point of view JMAP gives it a way to trade-off one munition for another. JMAP supplies the ability to improve the allocation of shooters and munitions to targets in such a way as to reduce cost while meeting goals priorities concerning kills and attrition. JMAP can also force minimum expenditures of a munition type as well as limit expenditures of munitions which can no longer be obtained.

To make the subject prototype integration manageable, we modify JMAP's Version 1.0 Production formulation of reference [e]; we use the data from the Maritime Prototype of reference [d], filling in data gaps using the LANL MRC-West database; and we employ the multi-objective linear goal programming solver of Schniederjans utilized by the Mobile Hard Prototype of reference [c] as well as by the Maritime Prototype.

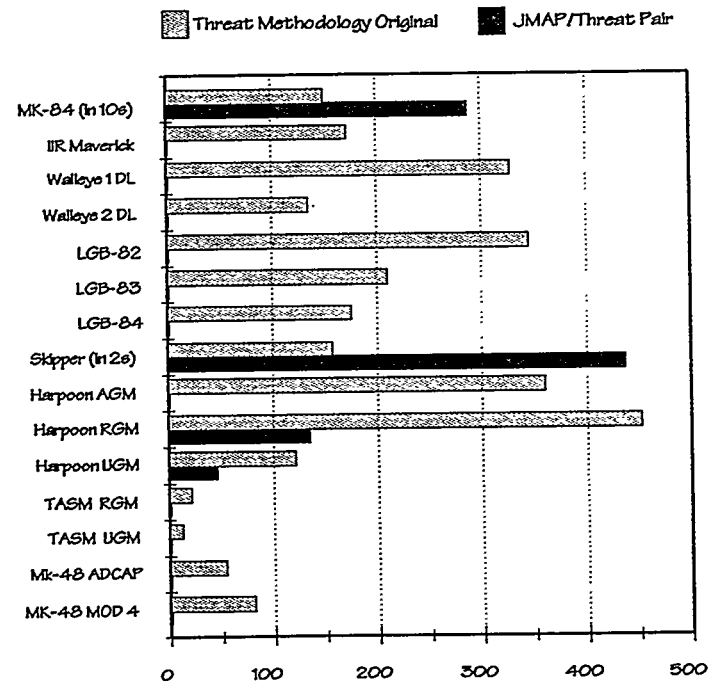
The joint application idea discussed in this memorandum involves the cyclic application of JMAP and the OSD Threat Methodology until a solution emerges. The procedure is: (1) allow the OSD Threat Methodology: (a) to initialize the allocation of shooter/munition combinations to targets, (b) to then generate a set of munitions expenditures, and (c) to then calculate the necessary safety stocks needed to support those expenditures; (2) invoke JMAP to apply constraints after factoring JMAP's munition costs to include the cost of the safety stocks — JMAP will then produce a more cost-efficient allocation; (3) feed this new allocation back to the Threat Model and let it recalculate the concomitant safety stocks, and (4) repeat steps (2) and (3) until a stable solution obtains.

To make this process work smoothly, we need to modify the formulation of JMAP Version 1.0 We remove the remaining time considerations from Version 1.0 by converting minimum and maximum rates of expenditure to aggregate expenditure constraints. We also add to JMAP a set of upper and lower constraints on the number of targets of each type a particular shooter type can kill; this is a critical output of the Threat Methodology. We privilege the Threat Methodology this way because it has passed both LANL's technical (references [f] and [g]) and the OM's utility (reference [h]) evaluations. We also suspend the quickest-kill munition selection within the Threat Methodlogy so that it can initialize JMAP properly.

The JMAP/Threat Pair works fine when JMAP is operated as originally designed, that is, when JMAP is free to make any allocation of shooter/munition combination to targets; in this case all targets turn out to be taken by aircraft with JMAP optimizing the allocation of aircraft/munition combinations to target type. The Pair also works fine when the minimal Threat Methodology shooter/target allocation constraint is applied; Exhibit 0 displays the stockpile preferred by the JMAP/Threat Pair in this constrained case as compared to the now robust stockpile produced by the Threat Methodology alone. Whereas the Threat Methodology spreads out the combat activity over all shooters and munitions brought into the scenario, the

EXHIBIT 0

Munitions Stockpile Solution Comparison



JMAP/Threat Pair concentrates combat activity onto specific shooter/munition combinations.

You can see that submarines are forced to kill some minimum number of targets, which are then most efficiently taken with Harpoon and not with TASM or torpedoes. Surface ships are also forced to kill some minimum number of targets, which are then most efficiently taken with Harpoon and not TASM. Airplanes handle the remaining targets, which are then most efficiently taken with Mk-84 (shown in units of 10) and, so it seems, Skipper (shown in units of 2). But the allocation to Skipper is by forcing 450 expenditures (as in reference [d]) which translates into Exhibit 0's recommended stockpile of 871. The JMAP/Threat Pair also works fine when JMAP is constrained to make exactly the Threat Methodology's allocation of shooters to targets but with the choice of munition free.

However, when the Pair is partially constrained in this allocation, we have sometimes observed erratic convergence behavior. One particular erratic behavior pattern is the bi-stable solution; this is one which oscillates alternatively from one type of allocation to another. We have also observed multi-stable solutions and other erratic behavior, including non-convergence. We surmise that convergence to a single solution fails because: (1) the cost factor changes introduced into each JMAP iteration could be a likely source of the observed multi-stable solutions from alternative linear goal programming runs, and (2) the cumulative effect of numerical rounding on many thousands of iterative calculations carried out in both the Threat Model and in the 181×330 dimensional JMAP linear goal program could be the other likely source.

The operation of the JMAP/Threat Pair is not as clean and automatic as we had hoped. Some modifications to our approach might solve these difficulties but we were unable to conduct extensive tests at the present time. Nevertheless, we believe there is value to the Pair concept for it provides a better conventional munitions planning tool than either model separately. For example, the Pair could be used to optimize the incremental cost of adding munitions to existing stockpiles. It would do this by costing the munitions in existing stockpiles at zero (encouraging their use) and costing the additional munitions required in excess of current stockpiles at the projected buy price (which could be set to infinity for munitions which were out of production). The Pair could thus provide a conventional munitions fiscal programming tool where neither of the original models could.

The remainder of this Memorandum comprises five parts. The first is a description of the structure and use of the OSD Threat Methodology. The second is a description of JMAP and its use. The third discusses the concept of the joint application of JMAP and the OSD Threat Methodology. The fourth displays sample output of the joint application. The fifth is a summary and epilogue. Finally, three appendices contain details of the formulation, data, and computer code.

References

[a] "Final Report for the Task 1 Implementation of the OSD Threat Methodology on the MRC-West Scenario", by J. J. Bolmarcich, Linda C. Thiel, and Elizabeth F. Arnold, QUANTICS incorporated (QM 54:93a), Malvern PA 19355, 21 July 1993 UNCLASSIFIED

[b] "Technical Documentation of the NATO Threat Methodology", by J. J. Bolmarcich, M. A. Carchidi, and G. B. Orr, QUANTICS incorporated (QR 9:91), Malvern PA 19355, December 1990 UNCLASSIFIED

[c] "Joint Munitions Assessment and Planning Model (JMAP): A Methodology Developed to Assess Conventional Munitions Capabilities — Final Report", The Los Alamos National Laboratory LA-UR-89-3896, 15 November 1989 UNCLASSIFIED {The Mobile Hard Prototype}

[d] "Description of the Maritime Module of the Joint Munitions Assessment and Planning Model (JMAP)", by W.L.May, R.L.Bivins, M.L.Stein, and R.C.Gordon III, LA-UR-92-1868, Military Systems Analysis Group, Los Alamos National Laboratory, May 1992 UNCLASSIFIED {The Maritime Prototype}

[e] "JMAP Version 1.0 User Guide", by Alanna Burke and Kym Kittel, Military Systems Analysis & Simulation Group, Los Alamos National Laboratory, October 1993 UNCLASSIFIED {Version 1.0 Production Model}

[f] "High Level Conventional Munitions Models: A Technical Evaluation", by Douglas P. Anson, A-5:93-183, Military Systems Analysis & Simulations, Los Alamos National Laboratory, September 1993 UNCLASSIFIED

[g] "Statistical Analysis of QUANTICS' Threat Model: A Technical Evaluation", by Paul Kvam and Rick Picard, TSA-1, Los Alamos National Laboratory, May 1994 UNCLASSIFIED

[h] "High Level Munitions Models Evaluation: Utility and User Friendliness", by Jay Mandelbaum, ODASD PR, November 1993 UNCLASSIFIED

[i] "Estimating Target Overlap for Tank Warfare Sustainability", by J. J. Bolmarcich, G. B. Orr, and D. M. Pasceri, QUANTICS incorporated (QR 1:91), Malvern PA 19355, January 1991 UNCLASSIFIED

1. The Structure and Use of the OSD Threat Methodology

The purpose of this section is to discuss the structure and use of the OSD Threat Methodology. We briefly discuss the basic approach, how the approach is implemented, which software products are used, how the methodology is best used, and what information of potential importance to the Office of Munitions it can not provide on its own.

The purpose of the OSD Threat Methodology is to estimate how many munitions of various types each of the Services need to win a future war. Because today's war-reserve munitions' stockpiles were primarily designed to fight a global war that has not come off, it is only prudent now to redesign them so as to "right-size" munitions' stockpiles for more likely future wars. Once this is done, munitions inventory analysts can create an explicit, feasible plan to manage the transition from the current stockpiles to the new desired stockpiles.

The basic approach of the OSD Threat Methodology to the sizing of munitions' stockpiles is to achieve robustness through controlled overstocking: If you were fighting a war today, you might reasonably arrive at the stockpile you need tomorrow by simply replacing what you expended yesterday. But if you are not fighting today, then how do you proceed? The Threat Methodology proceeds by first assuming: (1) that you view your job as one of providing a robust stockpile so future command can improvise in response to the unexpected, and (2) that you trust future wartime command decisions to take the stockpile you provide, set combat goals, resolve combat uncertainties as they arise, and win. The Methodology thus purposely avoids forecasts of unpredictable combat action details; it uses skeleton scenarios, general employment concepts, and simple models in order to get the munitions stockpiles roughly right.

The major implementation assumptions and concepts are: (1) count up all the targets in the scenario, (2) assume shooters kill targets, on average, in proportion to shooters' TASCFORM™ figures of merit, and their length of time in scenario, (3) augment average targets to be killed by munition type on the basis of past wars' kill variability — for robustness, (4) assume shooters select munitions which provide the quickest kill of each target type, and (5) compute the munitions required to kill X% of the augmented targets on average, or to kill at least X% of the targets with Y% assurance. For the prototype application discussed here, we select the former measure of effectiveness since it is simpler to implement.

In order to accomplish its job, the OSD Threat Methodology uses two kinds of software. First, R:BASE is used: (1) to hold the OSD threat data base, (2) to allot targets to shooter types, (3) to select the quickest-kill munition type for a shooter type to use against a target type, (4) to allocate overlap and false targets to each munition type's average target allotment, and (5) to construct input files for the next piece of software — the OSD Threat Model — one file for each munition type of each Service. (QUANTICS has also constructed specialized WINDOWS-based database editors which can build and maintain databases similar to that of the OSD threat database.)

The second piece of software is the OSD Threat Model itself. Its job is to estimate each munition type's expenditures needed to kill X% of all assigned targets. It also estimates the munitions shooters need "on-board" (up to maximum loads) in order to have munitions where they are needed when they are needed (it thus compensates for munitions' losses through attrition). It balances initial loads and combat reserves for replenishment so that X% of targets can be killed on average based on past wars' analyses of the engagement variability from shooter to shooter. This leads to a specific war-reserve stockpile recommendation for each munition type by Service. The stockpile recommendation less the estimated number of rounds expended in killing targets represents the non-expenditures left over after the war which the Model says you need as safety stocks to assure that munitions are in the right place at the right time.

The OSD Threat Model is designed to provide a basic combat load to each shooter and then to plan to replenish the highest expenders who might exceed their basic load. The Model cannot recommend war reserve stockpiles greater than total expenditures plus doctrinal basic loads. The Model is best used when "loadouts + expenditures" are too expensive or where post-scenario residuals are not a concern (for example, you expect to rebuild stockpiles later on in peacetime). You can mimic this latter situation in a more general context by packaging up enough scenarios to span one procurement cycle, say two years; the OSD Threat

Model will produce sufficient munitions' residuals to bridge the gap between scenarios' expenditures and new buys. You might also package up sufficient war and training scenarios to span the life-cycle of a munition; the Model will then estimate the total number of munitions of the type needed until the IOC of its follow-on.

There are several important things that the OSD Threat Model is not designed to do. First, because it is "time independent", it cannot assess whether or not its recommended munitions can be successfully delivered over time, neither the shooters ability to shoot it nor the logistics ability to transport it; it presumes the ability of your force structure to kill the enemy given enough time. Second, it cannot assess the relative desirability of buying or selling munition "A" versus munition "B" when each has recommended stockpiles which differ from the currently projected in-bin, neither for inventory management nor for inventory procurement strategy; the Model calculates stockpile requirements without reference to existing inventories. Third, it cannot force the consumption of an in-bin munition nor assess any "second best" tradeoffs, neither for munitions on the same shooter nor for different munitions on different shooters; however, after you review the Model's output, you are free to "factor" existing munitions against the Model's recommended stockpiles.

In summary, the OSD Threat Methodology exists to provide a forward-looking munitions' war reserves stockpile independent of existing inventory. R:BASE is used to invoke simple methods which embody aggregate time and space considerations and which produce a controlled overestimate of the number of targets each munition type should be prepared to kill. The Threat Model is designed to estimate both munitions' expenditures and non-expended safety stocks to produce controlled overestimate of munitions needed during combat so that X% of the targets are killed on average.

2. JMAP

The Joint Munitions Assessment Planning Model (JMAP) is a model designed by LANL to provide a methodology to enable the OSD Office of Munitions to do rapid turn-around, first order, munition trade-off analyses. It applies a linear goal programming technique to optimally allocate shooter effort and munitions to targets within the conflict duration. Mobile Hard Prototype of reference [c] used two overall goal constraints: a cost goal and a target-kill goal. The cost goal is there to minimize total munition cost and the kill goal to maximize the total number of targets killed.

The Maritime Prototype of reference [d] added fractional kill goals for selected target types and an attrition goal in order to minimize the total attrition to shooters. JMAP Version 1.0 (the production version) of reference [e] maintains these goals. The Version 1.0 also maintains the chance constraints on the number of rounds to kill a target which the Maritime version introduced. The remainder of our description refers to JMAP Version 1.0.

JMAP Version 1.0 has three sets of goal constraints and five sets of physical constraints. The three goal constraints are a cost goal, a targets-killed goal, and a shooters-killed goal. The cost goal is there to minimize the cost of total munitions expenditures plus the cost of lost shooters plus the cost of munitions lost on those lost shooters. For each target type in each time period there is a goal constraint on desired fraction of targets killed. For each shooter type, there is a goal constraint on the maximum fraction of shooter attrited.

JMAP Version 1.0 has three sets of physical constraints on munitions. They are (1) upper bounds on munitions expenditures due to stockpile availability as incremented by the maximum amount of munitions which could be produced within the time interval, (2) upper bounds on munitions expenditures limited by the operational rates of fire from shooters, and (3) lower bounds on munitions expenditures which must be fired due to doctrinal considerations in each time period, for example, artillery fired for suppressive effects. All these munitions constraints are 'hard' physical constraints and must be met.

There are also two sets of physical constraints on targets: (1) upper bounds on the number of targets available to be killed, and (2) upper bounds on certain types of targets to be engaged by certain munition types due to limitations on operational availabilities in each time period; this can account for lost opportunities because of battlefield geometry and maneuver. All these target constraints are 'hard' physical constraints and must be met.

Under the restriction that 'hard' physical constraints must be satisfied, JMAP tries to select

munitions for each shooter type in such a way that the total targets killed is as close to the *high* target goal as possible, the total shooter attrition is as close to the *low* attrition goal as possible, and the total cost is as close to the *low* cost goal as possible. In this sense, one may say that JMAP attains the maximum effectiveness in shooter/munition/target selections, and, hence, gives the best "bang for the buck".

JMAP Version 1.0 requires a number of inputs which can be categorized into the four sets found in Exhibit 1. Inputs representing goals are italicized. Not all these inputs are necessary for the joint application of JMAP with the OSD Threat Methodology. The next section indicates how we chose to modify JMAP Version 1.0 so that it can best be used with the OSD Threat Methodology.

3. Operating JMAP and the OSD Threat Methodology as an Integrated Pair

JMAP optimizes the allocation of munitions and shooters to targets, trading-off some for others, while under constraints on munitions stocks, expenditures, target kills, shooter attrition, and costs. JMAP tells you how you should run a war to consume the fewest dollars you have invested in munitions and shooters. The OSD Threat Methodology tells you the munitions stockpiles (expenditures + safety stocks) you need to run a war flexibly given the various ways you might end up operating your warfighting capabilities. Our idea for a working JMAP/Threat Integrated Pair combines these two different points of view in the following way:

JMAP does not trade-off munitions stockpiles; rather it trades-off munitions expenditures. JMAP could tradeoff stockpiles if the Threat Model would assess the safety stocks corresponding to JMAP's expenditures. Munitions expenditures from JMAP's goal optimization turn out identical to those of the Threat Model when the Threat Model operates with the same allocation of shooters/munitions to targets as does JMAP, and when JMAP dispenses with its chance-constrained rounds-to-kill. The allocation of shooters/munitions to targets that the Threat Model ordinarily gets from the Threat Split Model can be substituted for by JMAP's optimization instead. So, the Threat Model could 'tax' JMAP's expenditures for the cost of safety stocks and JMAP could be run with 'inflated' munitions' costs when generating its allocations.

EXHIBIT 1
JMAP Input Summary

Stocks:

(1) the number of shooters of each type available and *the desired fraction of each you will tolerate being attrited*;

(2) the number munitions of each type available to be expended;

(3) the number of targets of each type available to be killed and *the desired fraction of each you wish to kill;*

Allocations/Assignments:

(4) for each pairing of one shooter type with one target type, the number of munitions for each type employable by the shooter per engagement, the number of engagements needed to kill a target, and shooter attrition per engagement;

(5) the maximum fraction of each target type available to be killed by each munition type;

Costs:

(7) the cost of one munition of each type;

(8) the cost of one shooter;

Time:

(9) the length of each time period,

(10) maximum (operational) rates of fire by each shooter of each munition type,

(11) the minimum doctrinal rate of consumption of each munition type.

To make this idea work, we streamline JMAP so that it does what it does best, namely, provide an optimal allocation. Our first action is to dispense with the chance-constrained rounds-to-kill. Next, we remove the time-based constraints so as to make JMAP as threat-oriented as possible and thus have it more closely match the ethos of the OSD Threat Methodology. JMAP Version 1.0 has already disposed of multiple time periods and, since JMAP treats time as a period aggregate (not as a daily curb), any daily rates within JMAP now serve to impose expenditures limits during the overall conflict period. As a result, the time factors (9), (10), and (11) in Exhibit 1 disappear after we multiply the operational rates in (10) and the doctrinal rates in (11) by the total conflict time. This has the net effect of translating those time-based factors directly into expenditure constraints.

Furthermore, using JMAP with the Threat Model, whose job it is to estimate the right size of stockpiles, means that as a first cut even stockpile constraints should be lifted from JMAP. This would provide a baseline for the 'optimal' stockpiles. As a second cut, for example for out-of-production munitions such as Walleye, the current stock levels could be fed to JMAP as a stockpile constraint so that JMAP can reallocate effort if the baseline case required more than is available. Similarly, minimum expenditures of a munition, for example for doctrinal purposes or for to using up older munitions, can be introduced into JMAP as an expenditure constraint if the baseline case required fewer than we wish to consume.

In addition, to operate the JMAP/Threat Pair, we modify JMAP's cost goal. Ever since the Maritime Prototype JMAP has included in its cost goal the cost of lost shooters and the cost of munitions lost on those shooters. These formulations of JMAP mimic the Services munitions selection models which optimize "bang for the buck" using these cost factors as well as others. The idea is that when a shooter incurs different risks in delivering different munitions then the cost of delivery is somehow germane to the munition selection. But the Service models are not linear goal programs with attrition goal constraints while JMAP is; JMAP has a separate attrition goal to alter its shooter/muntion to target allocation. So why double count the impact of attrition. For our prototype JMAP/Threat Pair, we drop the explicit costing of shooter losses within the cost goal. We maintain the shooter attrition goal to represent attrition's operational reality.

Similarly, we drop the explicit costing of munitions lost on attrited shooters. The reason is that the Threat Model accounts for the safety stocks needed to support combat, which includes munitions which might be lost due to attrition. Since JMAP, working in the JMAP/Threat Pair, already includes the cost of the safety stocks through a 'safety-stock tax' resulting in an inflated cost-per-expenditure, it would be double counting to include the cost of these losses again.

The JMAP/Threat Pair operates through the cyclic application of JMAP and the OSD Threat Model until a stable solution emerges. The procedure is: (1) the OSD Threat Split Model initially allocates shooter/munition combinations to targets as a baseline feed to the Threat Model; (2) the Threat Model generates a set of munitions stockpiles comprising expenditures plus safety stocks — this is the 'TM-initialization'; (3) JMAP applies goal constraints after JMAP's munition costs are modified to include the cost of the safety stocks — this is accomplished by feeding JMAP the Threat Model's stockpile-to-expenditures ratios (the OSD Threat Methodology's quickest-kill munition selection is suspended so that it can initialize this ratio for each munition type); (4) JMAP's new allocation is fed back to the Threat Model which recalculates the stockpile-to-expenditures ratios; and (5) we repeat steps (3) and (4) until the iteration results stabilize. We end up with munitions requirements which are optimal in the JMAP sense and which are war reserve stockpile recommendations in the Threat Methodology sense.

Finally, to make the JMAP/Threat Pair work in a truly integrated fashion, we choose to place allocation constraints on JMAP — because JMAP yearns to allocate all targets to the single most efficient shooter-type/munition-type combination available. In order to use the operational capability of all the shooters put into the scenario, we include the influence of the baseline allocation of shooters to targets provided by the OSD Threat Methodology. This Methodology allocates shooters to targets in proportion to each shooter's TASCFORM™ figure of merit multiplied by each shooter's length of time in scenario. This estimate of the amount of combat action each type of shooter in the scenario will likely see assumes shooters will be employed in rough proportion to their capability, not cost.

The OSD Threat Methodology already has within it a method of augmenting the number of targets which it will allocate to each shooter/munition type on the basis of past wars' kill variability (see reference [a]). It does this by estimating a standard deviation σ in target kills, based on past warfare experience, and adding some number of σs to the baseline allocation. The idea is that in a real war any particular combination of shooter-type/munition-type may end up killing more targets than originally expected and the Methodology is designed to estimate stockpiles so that no shooter runs out with high confidence. For our prototype integration, we apply various multipliers of the standard deviation to create sets of both upper and lower bounds on the number of targets of each type which could be allocated to each shooter type by JMAP. JMAP remains free to select the 'optimal' munition type. Each set of kill-bounds corresponds to some confidence level that shooters will kill numbers of

targets between these bounds — assuming the OSD Threat Methodology baseline allocation was a correct estimate of the results of the 'average' war.

The larger the confidence level associated with a set of bounds, the larger the resulting spread in kill-bounds, and the larger the spread in targets JMAP can allocate to any shooter. (Consult reference [i] for details.) For example, we take the 100% confidence set of kill-bounds to corresponds to a lower bound 3.3 σ below the baseline (bottomed by zero targets) and an upper bound 3.3 σ above the baseline (capped by the maximum number of targets). Similarly, the 85% confidence set of kill-bounds corresponds to a lower bound 1.04 σ below the baseline (bottomed by zero targets) and an upper bound 1.04 σ above the baseline (capped by maximum targets). As a numerical example, when the baseline allocates 100 targets of a total of 400 to a shooter type with 10 shooters, then σ is about 30 targets and an 85% confidence set would allow an upper bound of about 130 targets and a lower bound of about 70. (The '50% confidence' set adds and subtracts 0.0 σ to the baseline allocation; this makes both the upper and lower bounds equal to the baseline and JMAP must allocate shooters to targets as specified by the Threat Methodology.)

A detailed formulation of the JMAP/Threat Pair is given in Appendix A. Exhibit A-1 defines the terms used. Exhibit A-2 shows the inputs and outputs of a Threat Model run for a given munition type. Exhibit A-3 provides the JMAP formulation in detail. Exhibit A-4 describes the precise iterative procedure of the JMAP/Threat Pair. Exhibit A-5 compares the various previous formulations of JMAP from references [c], [d], and [e] with the Pair formulation here.

Our prototype application draws input data from the Maritime Prototype report of reference [d] — as far as it goes. We retain the four target types (small major combatants, minor combatants, landing craft, and false targets), the 12 shooter (platform) types, the 17 munition types, the forced expenditure of 450 Skipper, and the stockpile constraints of which only that on Submarine-launched Harpoon (UGM) had any effect. We supplement some of reference [d]'s exchange ratios with the attrition data from the LANL MRC-West database. For the shooter and target figures-of-merit, we utilize the relative platform weights and relative target weights from JMAP Version 1.0.

The initial allowance, refill size, and reorder point for each munition type on each shooter type is taken from the LANL MRC-West database. The maximum rate of fire by each shooter of each munition type is based on the corresponding initial allowance. Appendix B contains our sample data with a sample input file given in Exhibit B-1. We emphasize our 'scenario' could in no way be interpreted as a MRC-West scenario. In particular, the counts of enemy targets and US shooters were unlike those in the MRC-West scenario. In addition, the baseline allocation of shooter/munition combinations to targets (see Exhibit B-2) was obtained from the US Navy Threat Split Model rather than on the existing allocation from the application of the OSD Threat Methodology on the MRC-West scenario reported in reference [a].

4. Results From the JMAP/Threat Pair

The purpose of this section is to report on the some of the JMAP/Threat Pair sample runs which were made to prove the prototype concept. This section is divided into two parts. The first presents some general observations concerning our sample runs. The second examines the results of some specific sample runs for the purpose of illustrating the observed convergence and non-convergence behaviors of the JMAP/Threat Pair.

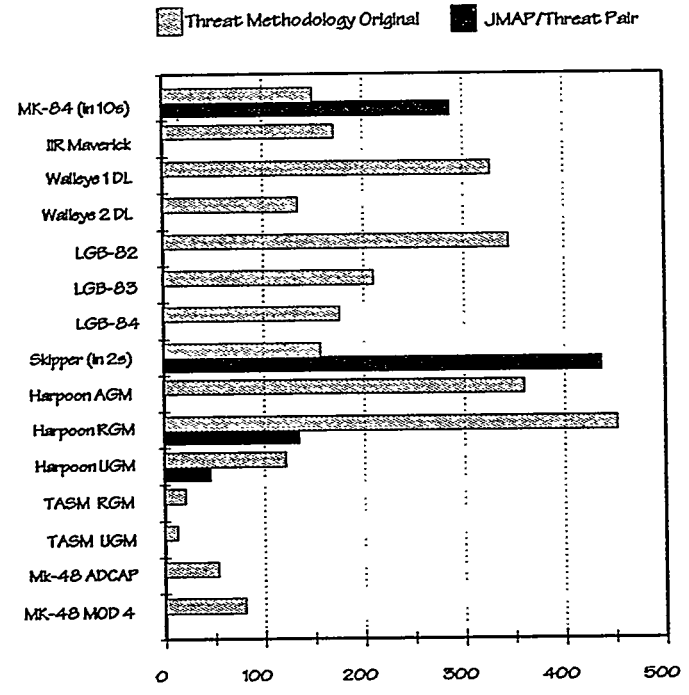
4.1 General Observations

For each of our sample runs, we needed to choose the relative priority of JMAP's three goals — the kill goal, the attrition goal, and the munitions cost goal. We chose the target kill goal to be of the highest priority, that is, JMAP was asked to make sure that a minimum fraction of targets of each type were killed before any other tradeoffs were permitted. We chose the shooter attrition goal as the second priority, that is, JMAP was asked to kill a minimum fraction of targets of each type and, in the process, to also keep the attrition to shooters down to the maximum levels desired, yet consistent with killing the desired minimum fraction of targets. Thus, if attrition to shooters was sufficiently high across the board for each engagement, the attrition goal would, of necessity, be violated, but to the minimum extent possible, while the target goal was being satisfied. The total munition cost was the third priority goal. That is, within the bounds for the killing of minimum fractions of targets of each type and within the bounds for maximum tolerable attrition, munitions were selected to reduce total munitions costs.

We also needed to select the stopping rule for pair-iterations. We chose to stop at one of two criteria: either (1) the difference of Threat Model average expenditures in two consecutive pair-iterations was less than 1% — which is a tighter criterion than the settling down of the JMAP allocation of munition to target, or (2) the number of pair-iterations had reached 16 — which indicates that the process would not likely converge to a single solution. We found that the number of pair-iterations were mostly more than 16 when applying kill-bounds associated with 77% through 99.5% confidence levels. Other cases normally converged in just a few iterations.

EXHIBIT 2

Least Kill-Bound Constraint Solution Comparison



The stockpile solution for the most constrained ('50% confidence') case had total munition costs of $0.968B compared to $0.271B for the least constrained (100% confidence) case. Intermediate munitions costs corresponded to intermediate confidence levels — as the confidence levels increased, the cost solutions decreased monotonically. This was because JMAP had no freedom to allocate targets to shooters in the '50% confidence' case while it had much greater freedom to allocate shooters to targets at the 100% confidence case where it then made the more efficient selection of munition against targets.

4.2 Some Specific Run Results

Let us first consider the Pair's least constrained case of 100% confidence in kill-bounds. Exhibit 2, repeating Exhibit 0, shows the original Threat Methodology solution (the 'TM-initialization') and the final (the third) of the JMAP/Threat Pair iterative solutions. The imposed expenditure of 450 Skipper appears in all iteration after the 'TM-initialization'. Exhibit 2 includes safety stocks of 421 from the final iteration (Skipper's total stockpile of 871 = 450 + 421 is shown in twos). On the first pair-iteration (not shown) JMAP selects some LGB-82 in addition to the Mk-84 and Harpoon shown in Exhibit 2. On the second pair-iteration, JMAP drops LGB-82 in favor of more Mk-84. This allocation is retained in the third pair-iteration and each munition's expenditures stabilized. (A forced fourth iteration turned out identical to the third.)

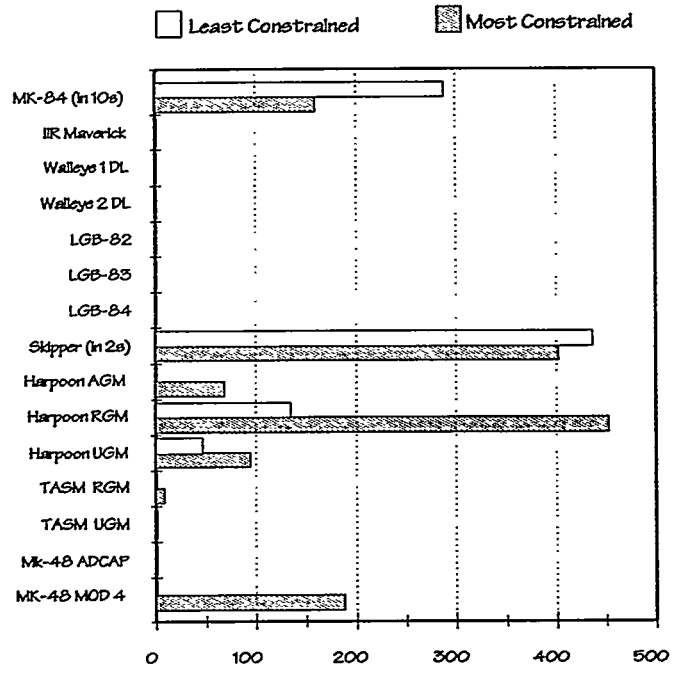
At the 100% confidence kill-bounds, it turns out that JMAP is constrained to allocate some minimal number of targets to surface ships and to submarines. AEGIS CGs and DDGs end up with the minimal number of minor combatants and other DDGs and SSNs end up with minimal numbers of landing craft. JMAP ends up choosing Harpoon as the most efficient munition in each case. Without these constraints JMAP would allocate all targets to Mk-84 after using up the forced 450 Skipper expenditures, which, as it turns out, are most efficiently applied against minor combatants.

The Mk-84 turns out to come from all the attack and fighter aircraft types (A-6 SWIP, F/A-18, F-14A/B, F-14D) each used most efficiently up to its attrition goal limits. The F/A-18 is used 'last' and takes up the slack just short of its attrition goal. Unfortunately, all false targets end up allocated to Mk-84, defeating the stockpile purpose of the false targets which is to allow each selected munition type a few extra expenditures to compensate for mis-targeting. False targets might be better handled by the JMAP/Threat Pair by allowing the Threat Model to pass them on to JMAP by

expanding the safety-stock tax idea into a combined a false-target/safety-stocks tax on expenditures. The JMAP/Threat Pair formulation here, however, retains the JMAP Maritime Prototype approach.

Next, let us consider the other extreme case, that is, kill-bounds at the '50% confidence' level. Here JMAP has no freedom to allocate targets to shooters but must duplicate the Threat Methodology's shooter/target allocation — the most constrained case — although JMAP does get to select the munition the shooter uses. The JMAP/Threat Pair stabilizes in three iterations. In addition to the 450 forced Skipper expenditures, JMAP selects Mk-84, Harpoon AGM, Harpoon RGM, Harpoon UGM, TASM RGM, and the Mk-48 MOD4. Exhibit 3 compares this most constrained case with the least constrained case of Exhibit 2. Several things need explanation.

EXHIBIT 3
Comparison of JMAP/Threat Pair Solutions Under Kill-Bound Constraint Extremes



First, in the most constrained case, certain amounts of targets are allocated to each type of shooter. Compared to the least constrained case, this shifts the target allocation towards surface ships and submarines and breaks the aircraft monopoly on targets; this then lowers the Mk-84 usage. Second, Skipper requirements are different because the 450 expenditures in the most constrained case are more concentrated on the F/A-18 and the net safety stocks produced by the Threat Model fall from 421 to 352; this means the recommended stockpile falls from 871 to 802.

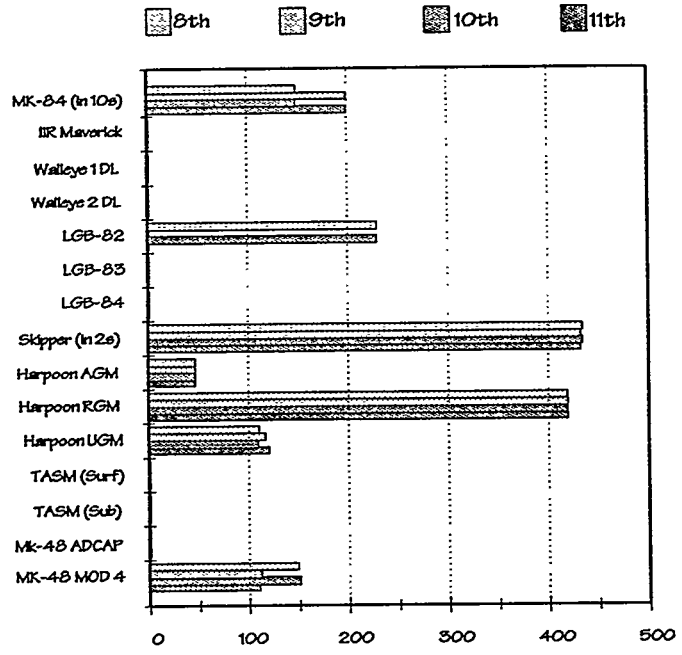
Third, in the most constrained case the patrol and ASW aircraft must kill their baseline allocated targets using Harpoon AGM since their lower kill bound is no longer 0 targets as it was in the least constrained case. Fourth, in the most constrained case all surface ships must kill their baseline allocated targets using much more Harpoon RGM. Also the non-AEGIS DDGs are forced well over their attrition goal in order to kill their allocated targets, so much so that they end up using some expensive TASM RGM in an attempt to balance munitions cost with attrition. Fifth, in the most constrained case both submarine classes must kill their allocated targets; the most efficient munition is Harpoon UGM but it bumps into its stockpile constraint and Mk-48 MOD4 is used to complete the kills.

For the 80% confidence level on kill-bounds, the iterative application of the JMAP/Threat Pair fails to converge to a single solution after 16 pair-iterations. However, it exhibits a bi-stable behavior in that the JMAP/Threat Pair solutions for the 4-th, 6-th, 8-th, 10-th, 12-th, 14-th, and 16-th pair-iterations (the even ones) strongly resemble one another while the solutions for the 5-th, 7-th, 9-th, 11-th, 13-th, and 15-th pair-iterations (the odd ones) also strongly resemble one another but are different from the even ones. Compared to the even ones, the odd one show a lot more Mk-84s, no LGB-82s, and a little less Mk-48 MOD4s. Comparing the pair-iterations 15 and 16, the 15-th has a higher total munition cost ($0.914B vs $0.861B) but a lower total shooter attrition cost ($0.394B vs $0.653B). In gross dollars for just these two cost items you might prefer iteration 15, but what about the other costs incident to the selection, such as storage, maintenance, etc. Exhibit 4 provides a graphical description of four consecutive pair-iterations: the 8-th, 9-th, 10-th, and 11-th.

Looking behind the pair-iterations displayed in Exhibit 4, you find that the same number of Minor Combatants (347) and FALSE targets (40) are killed in both the odd ones and the even ones. These numbers represent about the 0.8 goal of the total targets of each type available to be killed. The pair-iterations displayed some jitter on the kills of the 3 Small Major Combatants — 2.40 odd vs. 2.44 even — but the major distinction is in the kills of the 82 LCPAs — 81.56 odd vs. 66.68 even. LGB-82s are used by F-14A/B to kill Minor Combatants in the odd ones, while LGB-82 is not used at all in the even ones. On the other hand, more Mk-84s are used by A-6 SWIP, F/A-18, and F-14A/B to kill Minor Combatants in the even ones and that is why the even ones sustain a higher attrition. Basically, all the additional 14.88 LCPAs killed in the odd ones are killed by SSN 688 using Mk-48 MOD4s.

EXHIBIT 4

Four JMAP/Threat Pair-Iterations Under Kill-Bound Constraints for the 80% Confidence Level

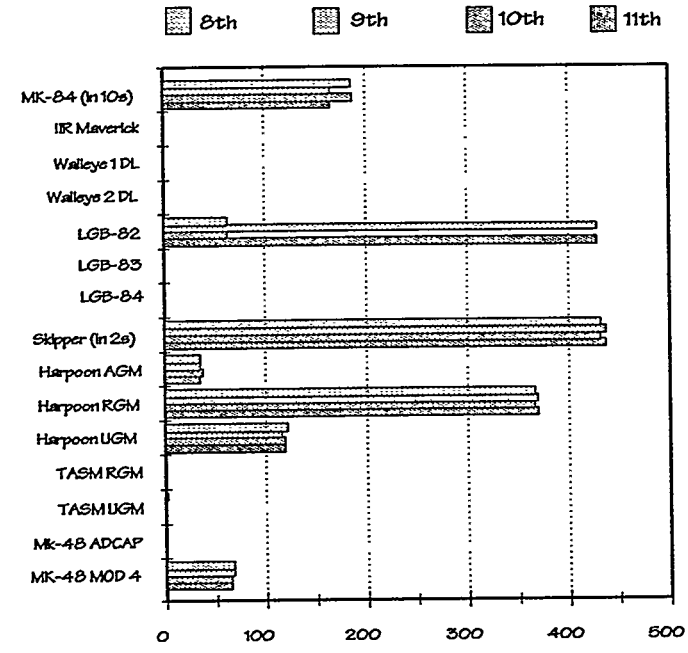


This bi-stable pattern seems to be an artifact of the changes in stockpile-to-expenditure ratio from one run to another. But in addition, there is jitter in the Mk-48 MOD4/Harpoon UGM struggle for targets in both the odd cases taken among themselves and in the even cases taken among themselves. Such an effect within the same families would seem to point to the impact of numerical rounding from one run to another.

Let us now consider the case of 90% confidence kill-bounds. The JMAP/Threat Pair process again goes on for 16 iterations with the process flip-flopping for pair-iterations 5 through 12, similar to that in the 80% confidence case. Exhibit 5 provides a graphical description of the Threat Model stockpiles for four consecutive Pair-iterations, the 8-th, 9-th, 10-th, and 11-th. More Mk-84s and less LGB-82s are expended in the even ones than in the odd ones. However, unlike in the 80% confidence case, the flip-flop behavior does not persist. From the 13-th pair-iteration to the 14-th, the Mk-84 requirements only increase slightly while the LGB-82 requirements

EXHIBIT 5

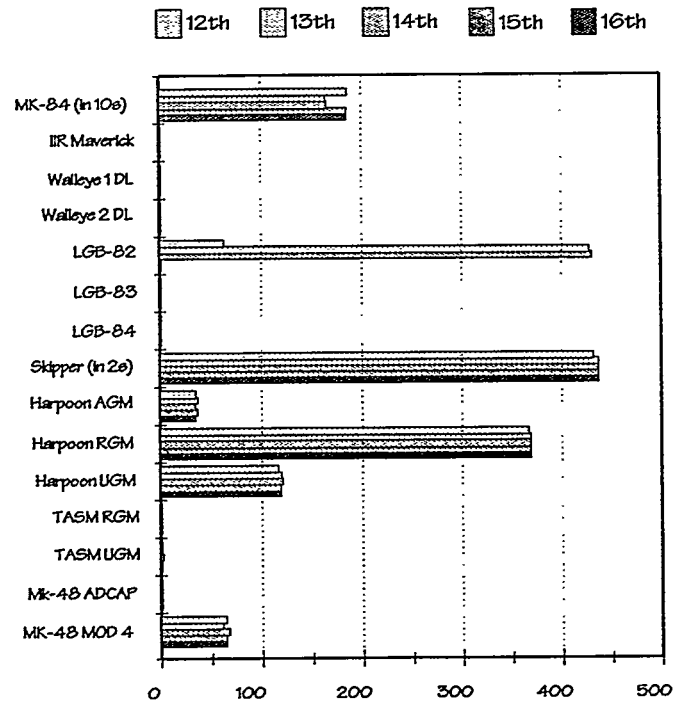
Four JMAP/Threat Pair-Iterations Under Kill-Bound Constraints for the 90% Confidence Level



increase instead of decrease. On the 15-th iteration, the Mk-84 requirements increase to about that of the previous even ones while LGB-82s are dropped completely. This stabilizes on the 16-th iteration in that the difference in Threat Model's expenditures with that of the next iteration would have been less than 1% for every munition type if the process had continued. Exhibit 6 illustrates the tail-end behavior for 90% confidence — showing the Threat Model stockpiles for pair-iterations 12 through 16.

EXHIBIT 6

Five Final JMAP/Threat Pair-Iterations Under Kill-Bound Constraints for the 90% Confidence Level



Results for the 95% confidence kill-bounds (not shown) exhibit no sign of bi-stable convergence. Consecutive iterations appear to look fairly close to one another but fail to meet our convergence criteria of 1% differences in consecutive iterations' expenditures per munition.

5. Summary and Epilogue

The joint application of JMAP and the OSD Threat Methodology discussed in this memorandum was conceived to produce a tool both to analyze and to optimize munitions stockpiles. From JMAP's point of view, the Threat Methodology helped it do its job by letting it work more realistically with stockpiles because in warfare not all of a stockpile can be expended at the enemy and the optimal shooter/munition combination may not acquire each target. From the Threat Methodology point of view JMAP gave it a way to trade-off one munition for another given various constraints.

The concept of joint application relied on the cyclic use of JMAP and the Threat Methodology hoping a stable solution would emerge. The procedure was: (1) allow the OSD Threat Methodology to initially allocate shooter/munition combinations to targets, then to generate a set of munitions expenditures, and then to calculate the necessary safety stocks needed to support those expenditures; (2) invoke JMAP to apply constraints after factoring JMAP's munition costs to include the cost of the safety stocks — JMAP then produces a more cost-efficient allocation; (3) feed this new allocation back to the Threat Model and let it recalculate the corresponding safety stocks, and (4) repeat steps (2) and (3) until a stable solution obtains.

The results of our sample runs showed that the JMAP/Threat Pair iteration process 'converges' rapidly when JMAP either has a relatively small amount of freedom or virtually complete freedom to allocate shooters over targets, corresponding to the target overlap confidence level from 50% to 76% and also from 99.6% to 100%. For most of the confidence levels in between, Pair iterations did not often converge to less than 1% differences between successive iterations. Some oscillated between two states but eventually converged, some exhibited bi-stable behavior without convergence, and some (such as the case of 93% confidence) oscillated among many states without convergence. Increasing the convergence criterion to 2% differences turned out not to capture many more solutions.

At this time, we do not have a complete explanation for the erratic behavior exhibited by our sample runs. But it appears that JMAP's sensitivity to small changes, the Threat Model's iterative cost factor recomputations, and the cumulative effect of numerical rounding of the thousands upon thousands of calculations in both models sometimes combine to have a destabilizing effect on the solutions from the JMAP/Threat Model Pair.

APPENDIX A

JMAP/Threat Pair

Formulation

EXHIBIT A-1
Glossary of Terms Used in the JMAP/Threat Pair

User Parameters :

t	KC^0_m	initial kill criterion for type m munition (average fraction of targets to be killed by type m munition - input to Threat Model)
j	$FTBK_t$	minimum fraction of type t targets to be killed (kill goal)
j	$FATTR_p$	maximum fraction of type p platforms allowed to be attrited (attrition goal)
j	Cost_Goal	overall cost goal
j	P_c	priority level for overall cost goal
j	P_k	priority level for kill goals
j	P_a	priority level for attrition goals
j	T_O_Confidence	confidence level for target overlap
j	Tolerance	stopping tolerance

Scenario Parameters :

j	CP_p	cost of one type p platform
j	CM_m	cost of one round of type m munition
tj	PQ_p	number of type p platforms
j	MQ_m	number of type m munitions
tj	TQ_t	number of type t targets
tj	T^0_{pmt}	number of type t targets to be killed by type p platforms using type m munition (initial target split - input to Threat Model)
j	T^0_{pt}	number of type t targets to be killed by type p platforms $T^0_{pt} = \sum_m T^0_{pmt}$ $TQ_t = \sum_p T^0_{pt} = \sum_{pm} T^0_{pmt}$
j	RPE_{pmt}	rounds-per-engagement of type m munitions, fired by type p platforms, against type t targets (i.e., salvo size)
j	STK_{pmt}	average number of salvos of type m munitions needed to kill a type t target, fired by a type p platform
	RTK_{pmt}	average number of rounds of type m munitions needed to kill a type t target, fired by a type p platform $RTK_{pmt} = RPE_{pmt} * STK_{pmt}$
j	ATT_{pmt}	fraction of a type p platform attrited per engagment while attacking type t targets using type m munitions

j	$DOCT_m$	doctrinal minimum expenditures of type m munitions per day
j	ROF_{pm}	maximum rate-of-fire of type m munitions fired by a type p platform each day
j	TIME	number of days in the scenario
j	FOM_P_p	figure-of-merit (relative value) for type p platforms
j	FOM_T_t	figure-of-merit (relative value) for type t targets
t	IA_{pm}	initial allowance of type m munitions on a type p platform
t	RS_{pm}	Refill Size of type m munitions on a type p platform
t	RP_{pm}	Reorder Point of type m munitions on type p platforms

Variables :

t	R_m	requirements for type m munitions (Threat Model output)
t	E_{pmt}	average expenditures of type m munitions by type p platforms against type t targets $E_{pmt} = KC^0_m * RPE_{pmt} * STK_{pmt} * T_{pmt}$
t	E_m	average expenditures of type m munitions $E_m = \sum_{pt} E_{pmt} = KC^0_m * [\sum_{pt} (RPE_{pmt} * STK_{pmt} * T_{pmt})]$
j	R_E_m	requirements-to-expenditures ratio for type m munitions $R\_E_m = R_m / E_m$
j	T_{pmt}	number of type t targets killed by type p platforms using type m munitions (JMAP outputs)
j	δ_{pt}	maximum allowable type t target overlap for type p platform
	X_{pmt}	average number of type m munitions, fired by type p platforms, against type t targets $X_{pmt} = RTK_{pmt} * T_{pmt} = RPE_{pmt} * STK_{pmt} * T_{pmt}$
j	CG^- , CG^+	deviation variables from the cost goal
j	KG^-_t , KG^+_t	deviation variables from the kill goal for type t targets
j	AG^-_p , AG^+_p	deviation variables from the attrition goal for type p platforms

Note: A 't' in the left most column indicates the term is used in the Threat Methodology and a 'j' indicates the term is used in JMAP.

EXHIBIT A-2
Inputs and Outputs of the Threat Model

For given munition type m :

Input : T^0_{pmt}
KC^0_m
RTK_{pmt}
PQ_p
IA_{pm}
RS_{pm}
RP_{pm}

Output : R_m

EXHIBIT A-3
JMAP Formulation

Inputs :

$FTBK_t$	CP_p	RPE_{pmt}	$DOCT_m$
$FATTR_p$	PQ_p	STK_{pmt}	ROF_{pm}
Cost_Goal	CM_m	$ATTRIT_{pmt}$	$COMP_{tt'}$
	MQ_m	FOM_P_p	$AVAIL_{pt}$
R_E_m	TQ_t	FOM_T_t	TIME

Outputs : T_{pmt} , CG^- , CG^+ , KG_t^- , KG_t^+ , AG_p^- , AG_p^+

Objective :

Minimize $Z = P_c [CG^+] + P_k [\sum_t FOM\_T_t * KG_t^-] + P_a [\sum_p FOM\_P_p * AG_p^+]$

Goal Constraints :

- Overall Cost Goal :
  $\sum_m [CM_m * R\_E_m * \sum_{pt} (RPE_{pmt} * STK_{pmt} * T_{pmt})] + CG^- - CG^+ = Cost\_Goal$

- Kill Goal per Target Type :
  $\sum_{pm} T_{pmt} + KG_t^- - KG_t^+ = FTBK_t * TQ_t$, $\forall$ t

- Attrition Goal per Platform Type :
  $\sum_{mt} (ATT_{pmt} * STK_{pmt} * T_{pmt}) + AG_p^- - AG_p^+ = FATTR_p * PQ_p$, $\forall$ p

Physical Constraints :

- Upper Bounds of Targets Killed by Platform Type :
  $\sum_m T_{pmt} \leq T_{pt}^0 + \delta_{pt}$, $\forall$ p, t

- Lower Bounds of Targets Killed by Platform Type :
  $\sum_m T_{pmt} \geq T_{pt}^0 - \delta_{pt}$, $\forall$ p, t

- Target Count per Target Type :
  $\sum_{pm} T_{pmt} \leq TQ_t$, $\forall$ t

- Platform Attritions per Platform Type :
  $\sum_{mt} (ATT_{pmt} * STK_{pmt} * T_{pmt}) \leq PQ_p$, $\forall$ p

- Munitions Available per Munition Type :
  $\sum_{pt} (R\_E_m * RPE_{pmt} * STK_{pmt} * T_{pmt}) \leq MQ_m$, $\forall$ m

- Doctrinal Expenditures per Munition Type :
  $\sum_{pt} (RPE_{pmt} * STK_{pmt} * T_{pmt}) \geq DOCT_m * TIME$, $\forall$ m

- Maximum Rate-of-Fire of Type of Munitions by Platform Type :
  $\sum_t (RPE_{pmt} * STK_{pmt} * T_{pmt}) \leq PQ_p * ROF_{pm} * TIME$, $\forall$ p, m

EXHIBIT A-4
Iterative Procedure of the JMAP/Threat Pair

set (P_c, P_k, P_a) e.g. (3, 1, 2)

T^0_{pmt}, $\forall$ p, m, t e.g. output from the Threat Split Program

Tolerance e.g. 0.01

[Computed Input] **[Output]**

$T^*_{pmt} \leftarrow T^0_{pmt}$, $\forall$ p, m, t

calculate average expenditures for the upcoming Threat Model run

do

$E^*_m \leftarrow E_m$, $\forall$ m

run Threat Model : [T^*_{pmt}] [$\mathbb{R}_m$]

calculate new requirement/expenditure ratios for the next JMAP run

run JMAP : [R_E_m] [$\mathbb{T}_{pmt}$]

calculate new target splits for the next Threat Model run

calculate average expenditures for the upcoming Threat Model run

while ($| 1 - E_m / E^*_m | >$ Tolerance, for any m)

Calculate average expenditures for the upcoming Threat Model run :

$$E_m = KC^0_m * [\sum_{pt} (RPE_{pmt} * STK_{pmt} * T^*_{pmt})] , \quad \forall m$$

Calculate new requirement/expenditure ratios :

$$R\_E_m = \mathbb{R}_m / E_m , \quad \forall m$$

Calculate new target splits :

$$T^*_{pmt} = TQ_t * \mathbb{T}_{pmt} / (\sum_{pm} \mathbb{T}_{pmt}) , \quad \forall p, m, t$$

EXHIBIT A-5
Comparison of Different Versions of JMAPs

	Nov_89 (Mobil Hard)	May_92 (Maritime)	Oct_93 (Ver. 1.0)	Jun_94 (Paired)
Constraints :				
Overall platform attrition ($\Sigma_p PQ_p$)	–	G	–	–
Overall cost (munitions expended)	G	–	–	G
" " (munitions expended + munitions lost + platforms lost)	–	G	G	–
Overall targets killed ($\Sigma_t TQ_t$)	G	G	–	–
Fraction of targets killed by target type and interval ($FTBK_{tl}$)	–	–	G	G
" " " " for high priority targets ($FTBK_{tl}$)	ph	G^L	–	–
Fraction of platform attrition by platform type (FA_p)	–	ph	G	G
Platform attrition by platform type (FA_p)	–	–	–	ph
Munition – availability by munition type ($STOCK/MQ_m$)	ph	ph	ph	ph
– minimum production by munition type ($PRODMIN_m$)	ph	–	–	–
– rate-of-fire (ROF_{pml})	ph	ph^L	ph	ph
– doctrinal ($DOCT_m$)	ph	ph^L	ph	ph
Target – count by target type (TQ_t)	ph	ph	ph	ph
– availability by platform ($AVAIL_{pl}$)	ph	–	ph	–
– upper bounds of targets killed by platform type ($T^0_{pt} + \delta_{pt}$)	–	–	–	ph
– lower bounds of targets killed by platform type ($T^0_{pt} - \delta_{pt}$)	–	–	–	ph

Note : G indicates goal constraint, ph indicates physical constraint, and superscript L indicates time-period dependency.

RTK – random variable (Chance Constrained LGP/LP)	–	x	x	–
Compound targets	–	–	x	–
Figure of merit (relative value) for platforms and targets	–	–	x	x
Number of intervals (L)	1	3	1	1
Number of days per interval	3	15, 15, 15	120	45
Number of platform types	10	12	81	12
Number of munition types	9	17	97	17
Number of target types	10	4	124	4
Goals :				
Priority Level :				
Attrition	–	2	1	2
Cost	1	3	1	3
Kill	1	1	1	1
Priority Weight :				
Attrition	–	1	40	1
Cost	1	1	400	1
Kill	1	1	1	1
	LGP Schniederjans	LGP Schniederjans	LP Lindo	LGP Schniederjans

APPENDIX B

JMAP/Threat Pair

Input File : file.in

Cost, Kill and Attrition Goal Prioritites
3 1 2

Cost Goal (Billions $)
0.0

Target Overlap Confidence Level (0.5 - 1.0)
0.7

Stopping Tolerance and Maximum Number of Pair-Iterations
0.01 16

Length of Senario (days)
45

Number of Types of Platforms, Munitions, and Targets
12 17 4

(p)	Platforms	Cost(Millions) (CP)	Quantity (PQ)	Figure of Merit (FOM_P)	Max Fraction of Attrition (FATTR)
1	A-6 SWIP	34.0	84	22.9	0.05
2	F/A-18	41.0	109	19.8	0.05
3	F-14A/B	37.0	45	36.5	0.05
4	F-14D	37.0	46	36.5	0.05
5	EA-6B	80.0	25	50.0	0.05
6	S-3	25.0	30	8.7	0.05
7	P-3	20.0	16	13.3	0.05
8	SSN 637	186.0	1	130.0	0.01
9	SSN 688	800.0	15	229.0	0.01
10	CGN 9/36/38	1225.0	3	116.6	0.01
11	AEGIS CG/DDG	886.0	16	394.6	0.01
12	DD/DDG	512.0	9	175.9	0.01

(m)	Munitions	Cost($) (CM)	Stocked (MQ)	Doctrine (DOCT)	Kill_Criterion by Munition Type (KC_O)
1	Mk 84	8000	20750	0	0.80
2	ABF(L)	6240	0	0	0.80
3	IIR Maverick	10000	987	0	0.80
4	Walleye 1DL	170000	2878	0	0.80
5	Walleye 2DL	170000	2878	0	0.80
6	LGB-82	12300	1947	0	0.80
7	LGB-83	16400	507	0	0.80
8	LGB-84	21000	504	0	0.80
9	Skipper	30700	2964	10	0.80
10	AIWS P-3I	67000	0	0	0.80
11	Harpoon (AGM)	1240000	548	0	0.80
12	Harpoon (RGM)	1240000	692	0	0.80
13	Harpoon (UGM)	1240000	118	0	0.80
14	TASM Conv(Surf)	3100000	133	0	0.80
15	TASM Conv (Sub)	3100000	149	0	0.80
16	MK 48 ADCAP	2044000	597	0	0.80
17	Mk 48 MOD4	800000	1616	0	0.80

(t)	Target	Type	Quantity (TQ)	Figure of Merit (FOM_T)	Min Fraction to be Killed (FTBK)
1	SMC		3	7.00	0.80
2	MC		431	1.07	0.80
3	LCPA		82	0.50	0.80
4	FALSE	FALSE	50	5.00	0.80

Platform (p)	Munition (m)	Target (t)	Attritition per Engagement (ATT)	Rounds per Engagement (RPE)	Salvos to Kill (STK)
1	1	1	0.050	4.	3.330
1	1	2	0.020	4.	2.380
1	1	3	0.000	4.	3.850
1	1	4	0.000	4.	1.000
1	2	1	0.040	5.	5.260
1	2	2	0.020	5.	2.500
1	2	3	0.000	5.	5.880
1	2	4	0.000	5.	1.000
1	3	1	0.001000	4.	2.380
1	3	2	0.001000	4.	1.110
1	3	3	0.001000	4.	2.700
1	3	4	0.000	4.	1.000
1	4	1	0.001000	2.	3.450
1	4	2	0.001000	2.	1.610
1	4	4	0.000	2.	1.000
1	5	1	0.001000	2.	3.570
1	5	3	0.001000	2.	3.030
1	5	4	0.000	2.	1.000
1	6	2	0.010	4.	1.960
1	6	4	0.000	4.	1.000
1	7	1	0.040	4.	2.560
1	7	2	0.010	4.	1.330
1	7	3	0.000	4.	3.700
1	7	4	0.000	4.	1.000
1	8	1	0.040	4.	2.040
1	8	2	0.010	4.	1.150
1	8	3	0.000	4.	2.000
1	8	4	0.000	4.	1.000
1	9	1	0.000	4.	2.500
1	9	2	0.000	4.	1.190
1	9	3	0.000	4.	3.330
1	9	4	0.000	4.	1.000
1	11	1	0.001000	4.	1.390
1	11	2	0.001000	2.	1.250
1	11	3	0.001000	2.	1.250
1	11	4	0.000	2.	1.000
2	1	1	0.050	2.	5.560
2	1	2	0.020	2.	4.000
2	1	3	0.000	2.	5.880
2	1	4	0.000	2.	1.000
2	2	1	0.040	5.	5.260
2	2	2	0.020	5.	2.500
2	2	3	0.000	5.	5.880
2	2	4	0.000	5.	1.000
2	3	1	0.000750	2.	4.760
2	3	2	0.000750	2.	1.850
2	3	3	0.000750	2.	5.560
2	3	4	0.000	2.	1.000
2	4	1	0.000900	2.	3.450
2	4	2	0.000900	2.	1.610
2	4	4	0.000	2.	1.000
2	5	1	0.000900	1.	7.140
2	5	3	0.000900	1.	5.880
2	5	4	0.000	1.	1.000
2	6	2	0.000	2.	3.850
2	6	4	0.000	2.	1.000
2	7	1	0.040	2.	5.000
2	7	2	0.010	2.	2.630
2	7	3	0.000	2.	7.140
2	7	4	0.000	2.	1.000
2	8	1	0.040	2.	4.170
2	8	2	0.010	2.	1.560
2	8	3	0.000	2.	4.000
2	8	4	0.000	2.	1.000
2	9	1	0.000	2.	5.000
2	9	2	0.000	2.	1.670

2	9	3	0.000	2.	6.670
2	9	4	0.000	2.	1.000
2	10	1	0.000	2.	4.350
2	10	2	0.000	2.	2.220
2	10	3	0.000	2.	4.760
2	10	4	0.000	2.	1.000
2	11	1	0.000750	4.	1.390
2	11	2	0.000750	2.	1.250
2	11	3	0.000750	2.	1.250
2	11	4	0.000	2.	1.000
3	1	1	0.050	2.	5.560
3	1	2	0.020	2.	4.000
3	1	3	0.000	2.	5.880
3	1	4	0.000	2.	1.000
3	6	2	0.000	2.	3.850
3	6	4	0.000	2.	1.000
3	7	1	0.040	2.	5.000
3	7	2	0.010	2.	2.630
3	7	3	0.000	2.	7.140
3	7	4	0.000	2.	1.000
3	8	1	0.040	2.	4.170
3	8	2	0.010	2.	1.560
3	8	3	0.000	2.	4.000
3	8	4	0.000	2.	1.000
4	1	1	0.050	2.	5.560
4	1	2	0.020	2.	4.000
4	1	3	0.000	2.	5.880
4	1	4	0.000	2.	1.000
4	6	2	0.000	2.	3.850
4	6	4	0.000	2.	1.000
4	7	1	0.040	2.	5.000
4	7	2	0.010	2.	2.630
4	7	3	0.000	2.	7.140
4	7	4	0.000	2.	1.000
6	11	1	0.000	2.	1.390
6	11	2	0.000	2.	1.250
6	11	3	0.000	2.	1.250
6	11	4	0.000	2.	1.000
7	11	1	0.000900	4.	1.390
7	11	2	0.000900	2.	1.250
7	11	3	0.000900	2.	1.250
7	11	4	0.000	2.	1.000
8	13	1	0.025000	3.	1.610
8	13	2	0.002500	2.	1.250
8	13	3	0.000050	1.	1.390
8	13	4	0.000	1.	1.000
8	15	1	0.000	3.	2.630
8	16	1	0.050000	2.	1.720
8	16	2	0.005000	2.	1.720
8	16	3	0.000250	1.	1.470
8	16	4	0.000	1.	1.000
8	17	1	0.050000	2.	2.630
8	17	2	0.005000	2.	2.630
8	17	3	0.000250	1.	2.440
8	17	4	0.000	1.	1.000
9	13	1	0.010000	3.	1.610
9	13	2	0.001500	2.	1.210
9	13	3	0.000050	1.	1.390
9	13	4	0.000	1.	1.000
9	15	1	0.000	7.	1.230
9	16	1	0.025000	2.	1.720
9	16	2	0.002500	2.	1.720
9	16	3	0.000100	1.	1.470
9	16	4	0.000	1.	1.000
9	17	1	0.025000	2.	2.630
9	17	2	0.002500	2.	2.630
9	17	3	0.000100	1.	2.440
9	17	4	0.000	1.	1.000
10	12	1	0.000	4.	1.390

10	12	2	0.000	2.	1.250
10	12	3	0.000	1.	1.390
10	12	4	0.000	1.	1.000
10	14	1	0.000	6.	1.330
11	12	1	0.000	4.	1.390
11	12	2	0.000	2.	1.250
11	12	3	0.000	1.	1.390
11	12	4	0.000	1.	1.000
11	14	1	0.000	7.	1.230
12	12	1	0.012500	4.	1.390
12	12	2	0.007500	2.	1.250
12	12	3	0.007500	1.	1.390
12	12	4	0.000	1.	1.000
12	14	1	0.000500	6.	1.330
0	0	0	0.000	0.	0.000

Platform (p)	Munition (m)	Rate-of-Fire/Day (ROF)	Initial Allowance (IA)	Refill Size (RS)	Reorder Point (RP)	
1	1	8	4	4	0	
1	2	16	8	8	0	Not used
1	3	4	2	2	0	
1	4	4	2	2	0	
1	5	4	2	2	0	
1	6	8	4	4	0	
1	7	8	4	4	0	
1	8	4	2	2	0	
1	9	8	4	4	0	
1	11	8	4	4	0	
2	1	8	4	4	0	
2	2	16	8	8	0	Not used
2	3	4	2	2	0	
2	4	4	2	2	0	
2	5	4	2	2	0	
2	6	8	4	4	0	
2	7	8	4	4	0	
2	8	4	2	2	0	
2	9	8	4	4	0	
2	10	6	3	3	0	Not used
2	11	8	4	4	0	Added
3	1	8	4	4	0	
3	6	8	4	4	0	
3	7	8	4	4	0	
3	8	4	2	2	0	
3	9	8	4	4	0	Not used
4	1	8	4	4	0	
4	6	8	4	4	0	
4	7	8	4	4	0	
4	9	8	4	4	0	Not used
6	2	8	4	4	0	Not used
6	6	8	4	4	0	Not used
6	11	8	4	4	0	
7	11	8	4	4	0	
8	13	8	4	2	2	
8	15	8	4	2	2	15 <- 14
8	16	20	10	5	5	16 <- 15
8	17	30	15	7.5	7.5	
9	13	16	8	4	4	
9	15	8	4	2	2	
9	16	40	20	10	10	
9	17	60	30	15	15	
10	12	16	8	4	4	
10	14	16	8	4	4	
11	12	16	8	4	4	
11	14	16	8	4	4	Added
12	12	16	8	4	4	
12	14	122	61	30.5	30.5	
0	0	0	0	0	0	

JMAP/Threat Pair

Input File : t_splits.in

Initial Targets Splits by Platform and Munition

	A-6 SW	F/A-18	F-14A/	F-14D	EA-6B	S-3	P-3	SSN 63	SSN 68	CGN 9/	AEGIS	DD/DDG	Total
Mk 84													
SMC	.20	.25	.11	.11	.00	.00	.00	.00	.00	.00	.00	.00	.67
MC	35.76	46.40	19.16	19.58	.00	.00	.00	.00	.00	.00	.00	.00	120.90
LCPA	2.27	2.95	1.22	1.24	.00	.00	.00	.00	.00	.00	.00	.00	7.68
FALSE	4.59	5.95	2.46	2.51	.00	.00	.00	.00	.00	.00	.00	.00	15.51
ABF(L)													
SMC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
IIR Maverick													
SMC	.01	.02	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.03
MC	3.58	4.65	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	8.23
LCPA	.13	.16	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.29
FALSE	.20	.26	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.46
Walleye 1DL													
SMC	.05	.06	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.11
MC	13.02	16.89	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	29.91
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
FALSE	.82	1.07	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	1.89
Walleye 2DL													
SMC	.04	.06	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.10
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.68	.88	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	1.56
FALSE	1.14	1.49	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	2.63
LGB-82													
SMC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
MC	3.67	4.76	1.97	2.01	.00	.00	.00	.00	.00	.00	.00	.00	12.41
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
FALSE	.43	.56	.23	.24	.00	.00	.00	.00	.00	.00	.00	.00	1.46
LGB-83													
SMC	.01	.01	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.02
MC	1.40	1.81	.75	.77	.00	.00	.00	.00	.00	.00	.00	.00	4.73
LCPA	.05	.06	.03	.03	.00	.00	.00	.00	.00	.00	.00	.00	.17
FALSE	.11	.15	.06	.06	.00	.00	.00	.00	.00	.00	.00	.00	.38
LGB-84													
SMC	.01	.01	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.02
MC	2.01	2.61	1.08	.00	.00	.00	.00	.00	.00	.00	.00	.00	5.70
LCPA	.09	.11	.05	.00	.00	.00	.00	.00	.00	.00	.00	.00	.25
FALSE	.11	.14	.06	.00	.00	.00	.00	.00	.00	.00	.00	.00	.31
Skipper													
SMC	.03	.04	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.07
MC	10.89	14.13	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	25.02
LCPA	.31	.40	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.71
FALSE	.59	.77	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	1.36
AIWS P-3I													
SMC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00

Harpoon (AGM)													
SMC	.09	.12	.00	.00	.00	.03	.02	.00	.00	.00	.00	.00	.26
MC	12.12	15.72	.00	.00	.00	4.33	2.31	.00	.00	.00	.00	.00	34.48
LCPA	4.03	5.24	.00	.00	.00	1.44	.77	.00	.00	.00	.00	.00	11.48
FALSE	1.41	1.82	.00	.00	.00	.50	.27	.00	.00	.00	.00	.00	4.00
Harpoon (RGM)													
SMC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.03	.14	.08	.25
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	18.01	96.03	54.03	168.07
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	1.58	8.43	4.74	14.75
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	.00	2.09	11.14	6.27	19.50
Harpoon (UGM)													
SMC	.00	.00	.00	.00	.00	.00	.00	.01	.17	.00	.00	.00	.18
MC	.00	.00	.00	.00	.00	.00	.00	1.35	20.20	.00	.00	.00	21.55
LCPA	.00	.00	.00	.00	.00	.00	.00	.62	9.23	.00	.00	.00	9.85
FALSE	.00	.00	.00	.00	.00	.00	.00	.16	2.34	.00	.00	.00	2.50
TASM Conv(Surf)													
SMC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.04	.19	.11	.34
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	:00	.00	.00	.00	.00
TASM Conv (Sub)													
SMC	.00	.00	.00	.00	.00	.00	.00	.02	.25	.00	.00	.00	.27
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
MK 48 ADCAP													
SMC	.00	.00	.00	.00	.00	.00	.00	.02	.32	.00	.00	.00	.34
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.00	.00	.00	.00	.00	.00	.00	1.10	16.53	.00	.00	.00	17.63
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
Mk 48 MOD4													
SMC	.00	.00	.00	.00	.00	.00	.00	.02	.32	.00	.00	.00	.34
MC	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00
LCPA	.00	.00	.00	.00	.00	.00	.00	1.10	16.53	.00	.00	.00	17.63
FALSE	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00	.00

****
* The following are obtained by summing the above over munition types. *
* They are not read in as inputs. They are for information purpose only. *
****

Initial Targets Splits by Platform

	A-6 SW	F/A-18	F-14A/	F-14D	EA-6B	S-3	P-3	SSN 63	SSN 68	CGN 9/	AEGIS	DD/DDG	Total
SMC	.44	.57	.11	.11	.00	.03	.02	.07	1.06	.07	.33	.19	3.00
MC	82.45	106.97	22.96	22.36	.00	4.33	2.31	1.35	20.20	18.01	96.03	54.03	431.00
LCPA	7.56	9.80	1.30	1.27	.00	1.44	.77	2.82	42.29	1.58	8.43	4.74	82.00
FALSE	9.40	12.21	2.81	2.81	.00	.50	.27	.16	2.34	2.09	11.14	6.27	50.00

APPENDIX C

JMAP/Threat Pair

Code File : pair.for

```
*************************************************************************
*                                                                       *
*  File Name    :  pair.for                                             *
*  Program Name :  JMAP_Threat_Pair                                     *
*                                                                       *
*  This program experiments with the integration of the OSD Threat Model *
*  and  JMAP ( a modification of LANL JMAP October 1993 Version 1.0 ).  *
*                                                                       *
*  Reference :  "Prototype Integration of the Joint Munitions Asseement and *
*               Planning model with the OSD Threat Methodology", by Roger *
*               Y. S. Lynn and J. J. Bolmarcich, QUANTICS incorporated  *
*               (QM 29:94), 30 June 1994.                               *
*                                                                       *
*  Programmed By :   Roger Lynn,   Quantics Inc.,  6/23/94              *
*                                                                       *
*************************************************************************


c     FORTRAN Source Files :
c
c         PAIR  -  top levels
c             pair.for
c
c         JMAP  -  This portion of the source code is a modification of the
c                  LANL JMAP (May 1992 Maritime Module) which utilizes the
c                  Schniederjans Linear Goal Programming Solver.
c             jmap.for
c
c         Threat Methodology  -  OSD Threat Model
c             tm_input.for
c             tm_monte.for
c             tm_inter.for
c             tm_rand.for
c
c         Include Files
c             pair_1.inc
c             pair_2.inc


c     FORTRAN Executable File :
c             pair.exe


c     Input Files :
c             file.in             inputs to JMAP_Threat_Pair run
c             t_splits.in         initial target splits (from Target Split Program)


c     Output Files :
c             overlap.bnd         upper and lower bounds of target splits
c             tm_t_in.#           Threat Model target splits input for pair-iteration #
c             tm_echo.#           Threat Model input echo for pair-iteration #
c             tm_req.#            Threat Model average requirements output for pair-iteration #
c             j_t_out.#           JMAP output of target allocatition over platforms and munitions for pair-iteration #
c             j_sum.#             JMAP solution summary for pair-iteration #
c             pair_sum.#          high level summary of Threat Model and JMAP outputs for pair-iteration #
c             overview            JMAP cost solution and fractions of targets killed for ALL pair-iterations
c             j_lgp.in            input for LGP solver
c             j_con               constraint type
c             j_obj               objective terms
c             j_coeff             coefficients for the last pair-iteration
c             j_rhs               right-hand-sides
c             j_lgp.out           LGP report for the last pair-iteration
```

```
      program   JMAP_Threat_Pair

c     ********************************************************************************
c     *  This program experiments with the integration of the OSD Threat Model       *
c     *  and  JMAP ( a modification of LANL JMAP October 1993 Version 1.0 ).         *
c     ********************************************************************************

      include 'pair_1.inc'

      integer       pass
      real          max_aberration



      call initialization

      pass = 1
      max_aberration = 1.0
      do while  ( (pass .le. max_pass) .and.
     .            (max_aberration .gt. stopping_tolerance) )
        call store_average_expenditures
        call threat_model(pass)
        call cal_new_req_expenditure_ratios(pass)
        call jmap(pass)
        call cal_new_target_splits
        call cal_ave_exp_for_coming_tm_run
        call cal_max_aberration(pass, max_aberration)
        pass = pass + 1
      enddo



      end
c     (*   JMAP_Threat_Pair   *)
```

```
c     Skeleton  of  JMAP_Threat_Pair
c                                                                    Code File

c     initialization                                                 pair.for
c          read_inputs                                               pair.for
c          read_init_t_splits_cal_bounds                             pair.for
c               factor                                               pair.for
c          read_first_target_splits                                  pair.for
c          set_percents_for_confidence                               pair.for
c          initialize_req_exp_ratios                                 pair.for
c          cal_ave_exp_for_coming_tm_run                             pair.for

c     do for each pair-iteration

c          store_average_expenditures                                pair.for

c          threat_model                                              pair.for
c               get_target_splits                                    pair.for
c               do  for each munition type  j
c                    get_inputs                                      tm_input.for
c                         echo_threat_model_inputs                   tm_input.for
c                    assign                                          tm_input.for
c                         amean                                      tm_input.for
c                    main_up                                         tm_monte.for
c                         chk_scaling                                tm_monte.for
c                         inventory                                  tm_monte.for
c                         get_iter                                   tm_monte.for
c                         initialize                                 tm_monte.for
c                              zero                                  tm_monte.for
c                         monte                                      tm_monte.for
c                              box                                   tm_monte.for
c                                   randm(1)                         tm_rand.for
c                              ref_setup                             tm_monte.for
c                              refills                               tm_monte.for
c                                   randm(1)                         tm_rand.for
c                              analyze                               tm_monte.for
c                              zerod                                 tm_monte.for
c                              dist                                  tm_monte.for
c                              calctarget                            tm_monte.for
c                              cumsum                                tm_monte.for
c                         randm(2)                                   tm_rand.for
c                         summary                                    tm_monte.for
c                              print_dist                            tm_monte.for
c                                   cdf                              tm_monte.for
c                                   getbd                            tm_monte.for
c                                   calc_mean                        tm_monte.for
c                         moe                                        tm_monte.for
c                              fval                                  tm_monte.for
c                              killperc                              tm_monte.for
c                              interpsetup                           tm_inter.for
c                              interp                                tm_inter.for
c                                   chechrng                         tm_inter.for
c                    display                                         tm_input.for
c                    call cal_tm_req_for_munition_type(j)            pair.for
c               enddo

c          cal_new_req_expenditure_ratios                            pair.for

c          jmap                                                      jmap.for
c               open_files                                           jmap.for
c               lgpinp                                               jmap.for
c                    g_con_overall_cost                              jmap.for
c                    g_con_targets_kill                              jmap.for
c                    g_con_platform_attrition                        jmap.for
```

```
c                   ph_con_upper_bound_targets_kill                    jmap.for
c                   ph_con_lower_bound_targets_kill                    jmap.for
c                   ph_con_target_counts                               jmap.for
c                   ph_con_max_platform_attrition                      jmap.for
c                   ph_con_munition_available                          jmap.for
c                   ph_con_doctrinal                                   jmap.for
c                   ph_con_max_rate_of_fire                            jmap.for
c               lgpsolve                                               jmap.for
c                   start                                              jmap.for
c                   simplx                                             jmap.for
c                   lgprint                                            jmap.for
c                   prtsolu                                            jmap.for
c                       xet                                            jmap.for
c               close_files                                            jmap.for


c           cal_new_target_splits                                      pair.for

c           cal_ave_exp_for_coming_tm_run                              pair.for

c           cal_max_aberration                                         pair.for

c       enddo
```

```
      subroutine   initialization

c     ***************************************************************************************
c     *   This subroutine is the main initialization process for the JMAP_Threat_PAIR.        *
c     *   Referenced by:                                                        main program  *
c     ***************************************************************************************



      call read_inputs
      call read_init_t_splits_cal_bounds
c      call read_first_target_splits
      call set_percents_for_confidence
      call initialize_req_exp_ratios
      call cal_ave_exp_for_coming_tm_run



      return
      end
c     (*   initialization   * )
```

```
      subroutine   read_inputs

C     ************************************************************************
c     *   This subroutine reads all inputs to the JMAP_Threat_PAIR,          *
c     *             excepts the initial target splits which is read in       *
c     *             by read_initial_target_splits.                           *
c     *   Referenced by:                                initialization       *
C     ************************************************************************

C     ********************************************************************************
C     *  DEFINITIONS:                                                                *
C     *    CM(J)           = THE UNIT COST(BILLION $) OF TYPE J MUNITIONS            *
C     *    COSTGOAL        = THE MUNITION COST GOAL                                  *
C     *    DOCTRIN(J)      = REQUIRED NUMBER OF TYPE J MUNITIONS TO BE EXPENDED      *
C     *    ATT(I,J,K)      = THE FRACTION OF TYPE I PLATFORMS LOST WHEN ENGAGING A   *
C     *                      TYPE K TARGET WITH TYPE J MUNITIONS.                    *
C     *    FTBK(K,L)       = THE FRACTION OF TYPE K TARGETS TO BE KILLED IN          *
C     *                      TIME INTERVAL L.                                        *
C     *    FATTR(I)        = THE MAXIMUM FRACTION OF TYPE I PLATFORMS WHICH MAY BE LOST *
C     *    NBNZX           = THE NUMBER OF NON-ZERO DECISION VARIABLES X(I,J,K)      *
C     *    PQ(I)           = NB OF TYPE  I  PLATFORMS                                *
C     *    RPE(I,J,K)      = THE NB OF TYPE J MUNITIONS FIRED BY A TYPE I PLATFORM   *
C     *                      AGAINST A TYPE K TARGET IN ONE ENGAGEMENT.              *
C     *   STK(I,J,K,L)     = THE NUMBER OF SALVOS OF TYPE J MUNITIONS FIRED FROM     *
C     *                      TYPE I PLATFORMS REQUIRED TO KILL TYPE K TARGETS        *
C     *                      WITH STANDARD DEVIATION, STDEV(I,J,K,L) IN INTERVAL L   *
C     *    STOCK(J)        = THE INVENTORY OF TYPE J MUNITIONS AT TIME ZERO          *
C     *    STDEV(I,J,K,L)  = THE STANDARD DEVIATION OF SALVOS REQUIRED TO KILL       *
C     *                      TYPE K TARGET WITH TYPE J MUNITIONS FIRED FROM          *
C     *                      TYPE I PLATFORMS DURING TIME INTERVAL L.                *
C     *    TQ(K)           = NB OF TYPE  K  TARGETS                                  *
C     ********************************************************************************

      INCLUDE 'pair_2.inc'


      DIMENSION STOKIJK(MAXL),STDEVIJK(MAXL)
      REAL LAMBDAIK(MAXL), LAMBDA(MAXI,MAXK,MAXL)

      CHARACTER*132 HEADING,ALLTIMES
      character*14  file_in


      print*,' Enter input file name'
      read(5,1000)    file_in
      open(1,file=file_in)
      REWIND 1

      IF (PTEST .EQ. 0) THEN
       PTEST=1
      ENDIF


      TEST=0
      NLINE=0


C     *************************
C     *  READ THE STUDY TITLE  *
C     *************************

      READ(1,1000) TITLE
 1000 FORMAT(A)
      NLINE=NLINE + 1

C     ***************************************************
C     *  READ THE COST,KILL AND ATTRIT GOAL PRIORITIES  *
C     ***************************************************
```

```
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,*) PREPRI(1),PREPRI(2),PREPRI(3)
      NLINE=NLINE + 4

      IF (PREPRI(2).EQ.PREPRI(1) .OR.
     1    PREPRI(3).EQ.PREPRI(1)) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' ************************************'
        PRINT*,' '
        PRINT*,' CAUTION  CAUTION  CAUTION  CAUTION'
        PRINT*,' '
        PRINT*,' KILL AND/OR ATTRIT GOAL PRIORITY IS THE SAME AS',
     1         ' THE COST GOAL PRIORITY'
        PRINT*,' '
        PRINT*,' ************************************'
        PRINT*,' '
        PRINT*,' '
        TEST=1
      ENDIF

      IF (PREPRI(1).GT.4 .OR. PREPRI(2).GT.4 .OR.
     1    PREPRI(3).GT.4) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' THE COST,KILL, OR ATTRIT  priority  IS GREATER THAN 4'
        PRINT*,' '
        PRINT*,' '
      ENDIF

c     NBPRIORS=4                                                    !  Hardwired
      NBPRIORS=3                                                    !!



C     ****************************************
C     *  READ THE COST GOAL VALUE (BILLION $),  *
C     ****************************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
c     READ(1,*)    COSTGOAL, costgoal_reduction_factor              !!  commented out  -  see  !#1
      READ(1,*)    COSTGOAL
      NLINE=NLINE + 3

C     ************************************************************
C     *  READ ALPHA - THE TARGET overlap CONFIDENCE MEASURE       *
C     ************************************************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
      read (1,*)    target_overlap_confidence
      NLINE=NLINE + 3
      if ((target_overlap_confidence .lt. 0.5) .or.
     .          (target_overlap_confidence .gt. 1.0)) then
        write(6,100)
        stop
      endif
c     ZK=ALFTOZK(ALPHA)                                             !  not used in JMAP



      READ(1,1000) HEADING
      READ(1,1000) HEADING
      read (1,*)    stopping_tolerance, max_pass
      NLINE=NLINE + 3
```

```
C     ***********************************
C     *  READ THE MODEL TIME INTERVALS  *
C     ***********************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,*)    TIME(1)
      NLINE=NLINE + 3


C     ************************************************************
C     *  READ THE NB OF TYPES OF PLATFORMS,MUNITIONS, AND TARGETS  *
C     ************************************************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
c     READ(1,*) NBI,NBJ,NBK,NBL
      READ(1,*) NBI,NBJ,NBK                                          !!
      NBL = 1                                                        !!
      NLINE=NLINE + 3

      IF (NBI.GT.MAXI .OR. NBJ.GT.MAXJ .OR. NBK.GT.MAXK
     1    .OR. NBL.GT.MAXL) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' CHECK FILE 1'
        PRINT*,' THE NUMBER OF PLATFORM,MUNITION OR TARGET TYPES',
     1         ' IS TOO LARGE - INCREASE PARAMETER VALUES IN THE CODE'
        PRINT*,' '
        PRINT*,' '
        STOP
      ENDIF


C     ***************************************************************************
C     *  READ THE PLATFORM NAMES, QUANTITIES, figure-of-merit, and MAX ATTRITIONS *
C     ***************************************************************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      DO 1 I=1,NBI

c       READ(1,1005) N,PLAT(I),PLATCOST(I),PQ(I),FATTR(I)
c1005   FORMAT(I3,2X,A,F9.0,F12.0,4F15.0)

        READ(1,1005) N,PLAT(I),PLATCOST(I),PQ(I),FOM_P(I),FATTR(I)   !!
 1005   FORMAT(I3,2X,A,F9.0,F12.0,4F15.0)                            !!
C       ****************************************
C       *  CHANGE COSTS TO BILLIONS OF DOLLARS  *
C       ****************************************
        PLATCOST(I)=PLATCOST(I) * 1.E-3
    1 CONTINUE


C     ***************************************************************************
C     *  READ THE  MUNITION NAMES, COSTS, STOCKPILED, and Doctrinal Expenditures  *
C     ***************************************************************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING

      STKCOST=0
      DO 2 J=1,NBJ
        READ(1,1005) N,MUN(J),CM(J),STOCK(J),DOCTRIN(J),KC_0(J)
C       ****** DOCTRIN(J) ADDED ON 7 FEB 92 ***********
```

```
C       ****************************************
C       *  CHANGE COSTS TO BILLIONS OF DOLLARS  *
C       ****************************************
        CM(J)=CM(J)*1.E-9
        STKCOST=STKCOST + CM(J)*STOCK(J)
    2 CONTINUE


C     ********************************************************************************
C     *  READ THE TARGET NAMES, QUANTITES, figure-of-merit, and FRACTIONS-TO-BE-KILLED  *
C     ********************************************************************************

      TOTARGS=0

      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      DO 4 K=1,NBK

        READ(1,1007) N,TARG(K),TARGTYPE(K),TQ(K),FOM_T(K),
     .               (FTBK(K,L),L=1,NBL)                              !!
 1007   FORMAT(I3,2X,2A,F17.0,5(F15.0))                               !!

        DO 3 L=1,NBL
          IF (FTBK(K,L) .GT. 1.0) THEN
C           ****************************************
C           *  FRACTION-TO-KILL IS GREATER THAN 1  *
C           ****************************************
            PRINT*,' '
            PRINT*,' CHECK FILE 1'
            PRINT*,' SOME FRACTIONS-TO-KILL ARE GREATER THAN 1'
            PRINT*,' '
            TEST=1
          ENDIF
    3   CONTINUE

        TOTARGS=TOTARGS + TQ(K)
    4 CONTINUE

C     ************************************************************************
C     *  PRE-SET THE SALVOS-TO-KILL ARRAY STK(I,J,K,L) TO LARGE VALUES       *
C     *  AND THE ARRAY ATT(I,J,K) TO ZERO.                                   *
C     ************************************************************************

      DO 20 I=1,NBI
        DO 15 J=1,NBJ
          DO 10 K=1,NBK
            DO  5 L=1,NBL
              STK(I,J,K,L)=1.E25
    5       CONTINUE
            ATT(I,J,K)=0.0
   10     CONTINUE
   15   CONTINUE
   20 CONTINUE

C     ********************************************************************************************************************
C     *  READ THE attrition-per-engagement, rounds-per-engagemengt, salvos-to-kill, and standard deviations  *
C     ********************************************************************************************************************

      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      NLINE=NLINE + 4
      NIJK=0
c  30 READ(1,*) I,J,K,EXIJK,SALVOIJK,(STOKIJK(L),STDEVIJK(L),L=1,NBL)
   30 READ(1,*) I,J,K,EXIJK,SALVOIJK,(STOKIJK(L),L=1,NBL)
```

```
      NLINE=NLINE + 1
      IF (I.GT.NBI .OR. J.GT.NBJ .OR. K.GT.NBK) THEN
        PRINT*,' '
        PRINT*,' CHECK FILE 1,  LINE ',NLINE,' - I,J,OR K TOO LARGE'
        TEST=1
      ENDIF
      IF (I.GT.0) THEN
        IF (SALVOIJK .EQ. 0) GO TO 30
        ATT(I,J,K) = EXIJK
        RPE(I,J,K) = SALVOIJK
        DO 31 L=1,NBL
        STK(I,J,K,L) = STOKIJK(L)
        STDEV(I,J,K,L) = STDEVIJK(L)
   31   CONTINUE
        NIJK=NIJK + 1
        MCOL(I,J,K)=NIJK
        GO TO 30
      ENDIF



C     ********************************************************************************
C     *  READ  maximum Rate-of-Fire OF TYPE J MUNITIONS by a TYPE I PLATFORM each day  *
C     ********************************************************************************
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      READ(1,1000) HEADING
      NLINE=NLINE + 4
   50 READ(1,*) I, J, LOAD, IA1, RS1, RP1
      NLINE=NLINE + 1
      IF (I.GT.NBI .OR. J.GT.NBJ) THEN
        PRINT*,' '
        PRINT*,' CHECK FILE 1,  LINE ',NLINE,' - I OR J TOO LARGE'
        TEST=1
      ENDIF
      IF (I.GT.0) THEN
        ROF(I,J) = LOAD
        IA(I,J)  = IA1
        RS(I,J)  = RS1
        RP(I,J)  = RP1
        GO TO 50
      ENDIF

      IF (TEST .GT. 0) THEN

C       ************************************************
C       *  THERE HAVE BEEN ERRORS IN INPUT FILE 1      *
C       ************************************************

        STOP
      ENDIF


      DO 70 L=4,MAXROWS
      PREPRI(L) =4.
      IF (PREPRI(L) .GT. NBPRIORS) THEN
C       SET THE NB OF DEVIATION PRIORITY LEVELS
        NBPRIORS=PREPRI(L)
      ENDIF
   70 CONTINUE


      open(8,file='overview')
      write(8,200)   target_overlap_confidence

      write(8,701)  prepri(1)
      write(8,702)  prepri(2)
      write(8,704)  prepri(3), (targ(k), k=1,NBK)
```

```
      return
  100 format('Target overlap confidence is NOT between 0.5 and 1.0 !')
  200 format(/,1x,'Target Overlap Confidence Level :  ',f5.3)
  701 format(/,38x,'Priority',
     .       /,1x,'Overall Cost             ',15x,i1)
  702 format(1x,'Fractions of Targets Killed',13x,i1)
  704 format(1x,'Fractions of Platforms Attrited',9x,i1,
     .       ///,4x,'Pair',8x,'Cost',18x,'Fraction of Targets Killed'
     .        ,/,1x,'Iteration    Solution',15x,4a8)
      END
c              ! read_inputs
```

```
      subroutine   read_init_t_splits_cal_bounds

c     ***************************************************************************
c     *   This subroutine reads initial targets splits and                      *
c     *   calculates the Overlapping Bounds for target splits.                  *
c     *   It writes to file 'targets_t_in' these target splits input for this   *
c     *   Threat Model run and target overlap bounds for the next JMAP run.     *
c     *   Referenced by:                                    initialization      *
c     ***************************************************************************
      include 'pair_1.inc'

      common /BOUNDS/ upper_bound(MAXI,MAXK), lower_bound(MAXI,MAXK)
        real          upper_bound, lower_bound
      common /TJ_0/ TJ_0(MAXJ)
        real          TJ_0

      character*1     blank
      character*14    filename
      character*15    name
      character*132   heading
      integer         i, j, k
      real            TIK(MAXI,MAXK)
      real            tot_targets(MAXK)
      real            temp1, temp2, beta, f, g, factor
      data            blank/' '/


c     ---   read initial target splits

      print*,' Enter file name for the initial target splits'
      read (5,100)   filename
      open(50,file=filename)

      read(50,1000) heading
      read(50,1000) heading
      read(50,1000) heading
      read(50,1000) heading
      read(50,1000) heading
      do  J=1,NBJ
        read(50,1000) heading
        read(50,807)  name
        do  K=1,NBK
          read(50,808)  name,  (TIJK(I,J,K),I=1,NBI)
        enddo
      enddo
      close(50)

c     ---   Calculate Target Splits by Platform
      do  K=1,NBK
        temp1 = 0.0
        do  I=1,NBI
          temp2 = 0.0
          do  J=1,NBJ
            temp2 = temp2 + TIJK(I,J,K)
          enddo
          TIK(I,K) = temp2
          temp1 = temp1 + temp2
        enddo
        tot_targets(K) = temp1
      enddo

c       ---   Calculate the overlap bounds for targets to be killed
        beta = 1.0                                                  ! Polya Parameter
        do  K=1,NBK
          do  I=1,NBI
            f = TIK(I,K) / tot_targets(K)
            g = sqrt(f*(1.0-f) * (f/(pq(I)*beta) + 1.0/tot_targets(K)))
            delta(I,K) = factor(target_overlap_confidence) * g
```

```
                    * tot_targets(K)
          TIK_0(I,K) = TIK(I,K)

          upper_bound(I,K) = TIK_0(I,K) + delta(I,K)
          lower_bound(I,K) = max( (TIK_0(I,K) - delta(I,K)), 0.0 )

        enddo
      enddo

c     ---   Calculate initial target splits by munition type
      do  J=1,NBJ
        temp1 = 0.0
        do  I=1,NBI
          do  K=1,NBK
            temp1 = temp1 + TIJK(I,J,K)
          enddo
        enddo
        TJ_0(J) = temp1
      enddo


      filename = 'overlap.bnd'
      open(23,file=filename)

c     ---   Write Target Splits by Platform
      write(23,820)   filename
      write(23,809)
      write(23,810)  (plat(I), I=1,NBI)
      do  K=1,NBK
        write(23,808)  targ(K), (TIK(I,K),I=1,NBI), tot_targets(K)
      enddo

c       ---   Write the overlap bounds for targets to be killed
        write(23,815)   target_overlap_confidence
        write(23,811)
        do  K=1,NBK
          write(23,808)  targ(K), (delta(I,K),I=1,NBI)
        enddo

c       ---   Write the lower bounds for targets to be killed
        write(23,812)
        do  K=1,NBK
          write(23,808)  targ(K), (lower_bound(I,K),I=1,NBI)
        enddo

c       ---   Write the upper bounds for targets to be killed
        write(23,813)
        do  K=1,NBK
          write(23,808)  targ(K), (upper_bound(I,K),I=1,NBI)
        enddo

c     ---   Write Initial Target Splits by Platform and Munition
      write(23,805)
      write(23,1000)  blank
      write(23,806)  (plat(I), I=1,NBI)
      do  J=1,NBJ
        write(23,1000)  blank
        write(23,807)  mun(J)
        do  K=1,NBK
          temp1 = 0.0
          do  I=1,NBI
            temp1 = temp1 + TIJK(I,J,K)
          enddo
          write(23,808)  targ(K), (TIJK(I,J,K),I=1,NBI), temp1
        enddo
      enddo

      close(23)
```

```
      return
  100 format(a14)
  805 format(///,32x,'Initial Targets Splits by Platform and Munition')
  806 format(10x,12(1x,a6),6x,'Total')
  807 format(a15)
  808 format(3x,a5,12(1x,f6.2),2x,2(5x,f6.2),5x,f6.4)
  809 format(///,40x,'Initial Targets Splits by Platform')
  810 format(/,10x,12(1x,a6),6x,'Total',/)
  815 format(///,1x,'Target Overlapping Confidence  =  ',f5.3)
  811 format(/,30x,'Overlapping Bounds for Targets to be killed')
  812 format(///,30x,'Lower Bounds for Targets to be killed')
  813 format(///,30x,'Upper Bounds for Targets to be killed')
  820 format(//,'File Name :   ',a14)
 1000 format(a)
      end
c     (*   read_init_t_splits_cal_bounds   *)









      real function   factor(xx)

      integer       i
      real          xx, yy, x(11), y(11)

      data  x/.5, .6, .7, .8, .85, .90, .95, .98, .99, .999, 1.0/       !!!
      data  y/0.0, .2533, .5244, .8416, 1.0364, 1.2816, 1.6449,
     .       2.0537, 2.3263, 3.09, 3.3/                                 !!!

      if ( (xx .ge. 0.5) .and. (xx .le. 1.0) ) then
        i = 1
        do while (xx .gt. x(i))
          i = i + 1
        enddo
        i = i - 1
        yy =  y(i) + (xx-x(i)) * (y(i+1)-y(i)) / (x(i+1)-x(i))
      else
        print *, 'Confidence level is not between 0.5 and 1.'
        stop
      endif
      factor = yy

      return
      end
c    (*   factor   *)
```

```
      subroutine   read_first_target_splits

c     ********************************************************************************
c     *   Currently, this subourtine call from 'initialization' is de-activated.     *
c     *   If the call of this subroutine was activated in 'initialization', then     *
c     *   the user would have to supply a file name for the 'first target splits'.   *
c     *   Usually, it is the same as the 'initial target split', unless the user     *
c     *   wants to begin a new run which picks up from where it was left off from a  *
c     *   previous run.  In this case, the user should use for 'first target splits' *
c     *   a 'tm_t_in.*' file from a previous run (with the first 16 lines deleted).  *
c     *   However, one should be aware of the loss of accuracy since the target     *
c     *   allocations stored in 'tm_t_in.*' are given only up to two places after    *
c     *   the decimal.                                                               *
c     *   Referenced by:                                        initialization       *
c     ********************************************************************************

      include 'pair_1.inc'

      character*14    filename
      character*15    name
      character*132   heading
      integer         i, j, k

c     ---   read initial target splits

      print*,' Enter file name for the first target splits'
      read (5,100)   filename
      open(51,file=filename)

      read(51,1000) heading
      read(51,1000) heading
      read(51,1000) heading
      read(51,1000) heading
      read(51,1000) heading
      do  J=1,NBJ
        read(51,1000) heading
        read(51,807)  name
        do  K=1,NBK
          read(51,808)  name,  (TIJK(I,J,K),I=1,NBI)
        enddo
      enddo
      close(51)

      return
  100 format(a14)
  807 format(a15)
  808 format(3x,a5,12(1x,f6.2),2x,2(5x,f6.2),5x,f6.4)
 1000 format(a)
      end
c     (*   read_first_target_splits   *)
```

```
      subroutine   set_percents_for_confidence

c     ************************************************************************
c     *   Choose confidence levels.                                          *
c     *   In evaluating the fraction of targets killed at different levels   *
c     *   of confidence, perckill are the chosen confidences (actually in    *
c     *   fractional form, not %).                                           *
c     *   Referenced by:                                initialization       *
c     ************************************************************************
      common/percents/perckill(6),nperc
        integer nperc
        real perckill

      integer       i

      nperc = 6                                                        !!!   v
      do  i=1,nperc
        perckill(i) = 0.75 + (i-1) * 0.05
      enddo
      perckill(6) = 0.99                                               !!!   ^

      return
      end
c     (*   set_percents_for_confidence   *)
```

```
      subroutine   initialize_req_exp_ratios

c     ************************************************************************
c     *   This subroutine initializes the requirment-to-expenditure ratios.  *
c     *   Referenced by:                                initialization       *
c     ************************************************************************

      include 'pair_1.inc'

      do  j=1,NBJ
        R_E(j) = 1.0
      enddo

      return
      end
c     (*   initialize_req_exp_ratios   *)
```

```
      subroutine  cal_ave_exp_for_coming_tm_run

c     ********************************************************************************
c     *   This subruotine calculates the average expenditures of each munition      *
c     *   type for the next Threat Model run.   Note that this is possible since    *
c     *   the average expenditures only depend on the target splits and kill        *
c     *   criteria.                                                                 *
c     *   Referenced by:                  initialization and the main program       *
c     ********************************************************************************

      include 'pair_1.inc'

      integer       i, j, k, L
      real          epsilon, E

      data          epsilon/0.00001/

      L = 1                                                          !  Single time interval

      do  j=1,NBJ
        E = 0.0
        do  i=1,NBI
          do  k=1,NBK
            E = E +  RPE(i,j,k) * STK(i,j,k,L) * KC_0(j) * TIJK(i,j,k)
          enddo
        enddo
        if (E .lt. epsilon) then
          EJ(j)  = 0.0
        else
          EJ(j)  = E
        endif
      enddo

      return
      end
c     (*   cal_ave_exp_for_coming_tm_run   *)




      subroutine   store_average_expenditures

c     ************************************************************************************************
c     *   This subroutine stores the munition expenditures by munition type of the current pass      *
c     *   so they may be compared with that of the next pass.                                        *
c     *   Referenced by:                                                          main program       *
c     ************************************************************************************************

      include 'pair_1.inc'

      do  j=1,NBJ
        EJ_old(J) = EJ(j)
      enddo

      return
      end
c     (*   store_average_expenditures   *)
```

```
      subroutine   threat_model(pass)

c     ************************************************************************
c     *   This subroutine is the main loop for processing the Threat Model run.  *
c     *   Referenced by:                                          main program   *
c     ************************************************************************

      include 'pair_1.inc'

      integer       pass
      character*1   ch_pass1
      character*2   ch_pass2
      character*14  filename1, filename2


      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
      common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200)
        integer  ntt, rt
        real     rnt, r
        real*8   ext
        character*20 targnames
      common/THRT_REQ/ thrt_model_req(MAXJ)
        real        thrt_model_req


      integer       iflag
      integer       file_unit, k


      call get_target_splits(pass)


      if (pass .lt. 10) then
        write(ch_pass1,321)   pass
        filename1 = 'tm_echo.'//ch_pass1
        filename2 = 'tm_req.'//ch_pass1
      else
        write(ch_pass2,322)   pass
        filename1 = 'tm_echo.'//ch_pass2
        filename2 = 'tm_req.'//ch_pass2
      endif

      file_unit = 10
      open(file_unit,file=filename1)
      write(file_unit,1000)   filename1

      lun=16
      open(lun,file=filename2)
      write(lun,100)   filename2

      write(6,10)   pass
      do  j=1,NBJ
        thrt_model_req(j) = 0.0
        call get_inputs(iflag,file_unit,j)                    !!  find it in    tm_inputs.f
        write(lun,200)   mun(j)
        write(6,200)     mun(j)
        do  k = 1, ncases
          ncase=k
          call assign                                         !!  find it in    tm_inputs.f
*         call echo
          if (rt.ge.1)   call main_up(rt,iflag)               !!  find it in    tm_montesc.f
          if (lun.eq.6)  call display                         !!  find it in    tm_inputs.f
        enddo
        if (rt.ge.1) then
          call cal_tm_req_for_munition_type(j)
```

```
        endif
      enddo
      close(lun)
      close(file_unit)

      return
   10 format(//,1x,'Pair_Iteration   ',i2,' :',//,1x,'Threat Model :')
  100 format(//,'File Name :  ',a14,
     .       30x,'Threat Model Output',//,15x,'Average Munition',
     .         ' Requirements for  X%  Targets Kill by Munition Type')
  200 format(2x,a)
  321 format(i1)
  322 format(i2)
 1000 format(//,'File Name :  ',a14,/)
      end
c     (*   threat_model   *)
```

```
      subroutine   get_target_splits(pass)

c     ************************************************************************
c     *   This subroutine gets target splits input from common block /TIJK/ and   *
c     *   writes to file 'targets_t_in' these target splits input for this Threat *
c     *   Model run.                                                             *
c     *   Referenced by:                                    threat_model         *
c     ************************************************************************

      include 'pair_1.inc'

      integer       pass
      character*1   ch_pass1
      character*2   ch_pass2
      character*14  filename

      character*1   blank
      integer       i, j, k
      real          TIK(MAXI,MAXK)
      real          tot_targets(MAXK)
      real          temp1, temp2
      data          blank/' '/


      if (pass .lt. 10) then
        write(ch_pass1,321)   pass
        filename = 'tm_t_in.'//ch_pass1
      else
        write(ch_pass2,322)   pass
        filename = 'tm_t_in.'//ch_pass2
      endif

      open(24,file=filename)

c     ---   Calculate Target Splits by Platform
      do  K=1,NBK
        temp1 = 0.0
        do  I=1,NBI
          temp2 = 0.0
          do  J=1,NBJ
            temp2 = temp2 + TIJK(I,J,K)
          enddo
          TIK(I,K) = temp2
          temp1 = temp1 + temp2
        enddo
        tot_targets(K) = temp1
      enddo

c     ---   Write Target Splits by Platform
      write(24,820)   filename
      write(24,809)
      write(24,810)  (plat(I), I=1,NBI)
      do  K=1,NBK
        write(24,808)  targ(K), (TIK(I,K),I=1,NBI), tot_targets(K)
      enddo

c     ---   Write Threat Model Input : Target Splits by Platform and Munition
      write(24,805)
      write(24,1000)  blank
      write(24,806)  (plat(I), I=1,NBI)
      do  J=1,NBJ
        write(24,1000)  blank
        write(24,807)  mun(J)
        do  K=1,NBK
          temp1 = 0.0
          do  I=1,NBI
            temp1 = temp1 + TIJK(I,J,K)
          enddo
```

```
          write(24,808)  targ(K), (TIJK(I,J,K),I=1,NBI), temp1
        enddo
      enddo

      close(24)

      return
  321 format(i1)
  322 format(i2)
  805 format(///,32x,'Targets Splits by Platform and Munition')
  806 format(10x,12(1x,a6),6x,'Total')
  807 format(a15)
  808 format(3x,a5,12(1x,f6.2),2x,2(5x,f6.2),5x,f6.4)
  809 format(///,40x,'Targets Splits by Platform')
  810 format(/,10x,12(1x,a6),6x,'Total',/)
  820 format(//,'File Name :   ',a14)
 1000 format(a)
      end
c     (*   get_target_splits   *)
```

```
      subroutine   cal_tm_req_for_munition_type(j)

c     ***********************************************************************
c     *   This subroutine calculates the average requirements of type  j    *
c     *   munition  -  output of Threat Model                                *
c     *   Referenced by:                                 threat_model        *
c     ***********************************************************************

      include 'pair_1.inc'

      integer       j

      common/EXPECTED_REQ/ expected_req_array(6)                    ! shared with  moe  in  tm_montesc.f
        real        expected_req_array
      common/THRT_REQ/ thrt_model_req(MAXJ)
        real        thrt_model_req


      integer       i
      real          xx, yy, x(8), y(8)

      data  x/0.00, 0.75, 0.80, 0.85, 0.90, 0.95, 0.99, 1.00/


      y(1) = 0.0                                                    ! Extrapolating
      do  i=2,7
        y(i) = expected_req_array(i-1)
      enddo
      y(8) = 2.0 * expected_req_array(6) - expected_req_array(5)    ! Extrapolating

      xx = KC_0(j)                                                  ! Threat Model Kill Criterion by munition type

      if ( (xx .gt. 0.0) .and. (xx .le. 1.0) ) then
        i = 1
        do while (xx .gt. x(i))
          i = i + 1
        enddo
        i = i - 1
        yy =  y(i) + (xx-x(i)) * (y(i+1)-y(i)) / (x(i+1)-x(i))
      else
        print *, 'Fraction-To-Be-Killed is not between 0 and 1 !'
        stop
      endif
      thrt_model_req(j) = yy

      return
      end
c     (*   cal_tm_req_for_munition_type   *)
```

```
      subroutine   cal_new_req_expenditure_ratios

c     ********************************************************************************
c     *   This subroutine calculates the new requirment-to-expenditure ratio for each   *
c     *   munition type.                                                                 *
c     *   Referenced by:                                              main program   *
c     ********************************************************************************

      include 'pair_1.inc'

      common/THRT_REQ/ thrt_model_req(MAXJ)
        real        thrt_model_req
      common/TM_COST/cost_m, cost_plat_att, cost
        real        cost_m, cost_plat_att, cost

      integer       i, j, k, L
      real          temp

      L = 1                                                  !  Single time interval

      cost_m = 0.0
      do  j=1,NBJ
        if (EJ(j) .gt. 0) then                               !  do not change R_E if Threat Model expenditure is zero
          temp   =  thrt_model_req(j) / EJ(j)
          R_E(j) =  anint(temp * 10000.0) / 10000.0          !  this is to take four places precision after decimal
        endif
        cost_m = cost_m + CM(j) * thrt_model_req(j)
      enddo

      cost_plat_att = 0.0
      do  i=1,NBI
        temp = 0.0
        do  j=1,NBJ
          do  k=1,NBK
            temp = temp +  ATT(i,j,k) * STK(i,j,k,L) * TIJK(i,j,k)
          enddo
        enddo
        cost_plat_att = cost_plat_att + PLATCOST(i) * temp
      enddo

c     cost = cost_m + cost_plat_att                          !  including platform attrition in cost goal
      cost = cost_m                                          !  not including platform attrition in cost goal

c     costgoal = costgoal_reduction_factor * cost            !#1 - commented out, so the cost goal for the
c                                                                  next pair-iteration is not a reduciotn of
c                                                                  the actual cost from the current pair-
c                                                                  iteration; instead, uses user supplied
c                                                                  cost goal for every pair-iteration
      return
      end
c     (*   cal_new_req_expenditure_ratios   *)
```

```
      subroutine   jmap(pass)

c     ************************************************************
c     *   This subroutine is the main process for the JMAP run.   *
c     *   Referenced by:                          main program    *
c     ************************************************************

      integer       pass

      call open_files(pass)                          !!  find it in   jmap.f
      call lgpinp                                    !!  find it in   jmap.f  -  form input file for LGP solver
      call lgpsolve(pass)                            !!  find it in   jmap.f  -  solve LGP problem
      call close_files                               !!  find it in   jmap.f

      return
      end
c     (*   jmap   *)
```

```
      subroutine   cal_new_target_splits

c     ********************************************************************************
c     *   This subroutine calculates the new target splits for the next Threat Model run.   *
c     *   Referenced by:                                                    main program   *
c     ********************************************************************************

      include 'pair_1.inc'

      common/THRT_REQ/ thrt_model_req(MAXJ)
        real         thrt_model_req

      common /TJ_0/ TJ_0(MAXJ)
        real          TJ_0
      common/TM_COST/cost_m, cost_plat_att, cost
        real         cost_m, cost_plat_att, cost


      integer       j
      real          temp, epsilon

      data          epsilon/0.00001/


c     ---   Calculate  jmap Targets Killed,  jmap Kill Criterions,
c                      and  DIFF  by munition type
      do  J=1,NBJ
        temp = 0.0
        do  I=1,NBI
          do  K=1,NBK
            temp = temp + TIJK(I,J,K)
          enddo
        enddo
c         print*,'TJ(',J,') = ',temp,'    TJ_0(',J,') = ',TJ_0(J)       !!!
        if ((temp .lt. epsilon) .and. (TJ_0(J) .lt. epsilon)) then
          KC(J) = KC_0(J)
        else
          KC(J) = temp/TJ_0(J)
        endif
c         print*,'KC(',J,') = ',KC(J)                                     !!!
        DIFF(J) = abs( 1.0 - KC(J)/KC_0(J) )
      enddo



c     ---   Calculate new Target Splits for Threat Model to use in the next pass

      do  K=1,NBK
        temp = 0.0
        do  I=1,NBI
          do  J=1,NBJ
            temp = temp + TIJK(I,J,K)
          enddo
        enddo
        do  I=1,NBI
          do  J=1,NBJ
            TIJK(I,J,K) = tq(K) * TIJK(I,J,K) / temp
          enddo
        enddo
      enddo

      return
      end
c     (*   cal_new_target_splits   *)
```

```
      subroutine    cal_max_aberration(pass, max_aberration)

c     ********************************************************************************
c     *   This subroutine calculates the maximum difference of the ratio of expenditures    *
c     *   of two consecutive passes from  1  (of all munition types).                       *
c     *   Referenced by:                                                    main program   *
c     ********************************************************************************

      include 'pair_2.inc'

      integer       pass
      real          max_aberration

      common/THRT_REQ/ thrt_model_req(MAXJ)
        real        thrt_model_req
      common/TM_COST/cost_m, cost_plat_att, cost
        real        cost_m, cost_plat_att, cost

      integer       j
      character*1   ch_pass1
      character*2   ch_pass2
      character*14  filename
      real          ratio(MAXJ), aberration, epsilon

      data          epsilon/0.00001/


      if (pass .lt. 10) then
        write(ch_pass1,321)   pass
        filename = 'pair_sum.'//ch_pass1
      else
        write(ch_pass2,322)   pass
        filename = 'pair_sum.'//ch_pass2
      endif


      max_aberration = 0.0
      do  j=1,NBJ
        if (EJ_old(j) .lt. epsilon)  then
          if (EJ(j) .lt. epsilon)  then
            ratio(j)  =  1.0
          else
            ratio(j)  =  99999.0
          endif
        else
          ratio(j)  =  EJ(j)/EJ_old(j)
        endif
        aberration  =  abs ( 1.0 - ratio(j) )

        if (aberration .gt. max_aberration) then
          max_aberration = aberration
        endif
      enddo


c     ---   Write  Given Kill Criteria,
c                  Threat Model : Requirements, Expenditures, and Req-to-Expenditure Ratios
c                  JMAP         : Expenditures, Requirements, and Kill Criteria

      open(17,file=filename)

      write(17,100)   target_overlap_confidence, pass
      write(17,105)
      do  j=1,NBJ
        write(17,200) mun(j), thrt_model_req(j), EJ_old(j), R_E(j),
     .                       rndsum(j), reqsum(j), ratio(j)
      enddo
```

```
      write(17,210)
      write(17,220)  costgoal, tot_mun_req_cost, prepri(1)

      write(17,230)  prepri(2)
      L = 1
      do  k=1,NBK
        write(17,240)  targ(k), ftbk(k,L), ftbk(K,L), totkill(k)/tq(k)
      enddo

      write(17,250)  prepri(3)
      do  i=1,NBI
        write(17,260)  plat(i), fattr(i), totatt(i)/pq(i)
      enddo

      close(17)

  100 format('Overlap Confidence = ',f5.3,22x,'Pair_Iteration ',i2,
     .       //,18x,'<----------   Threat Model  ---------->'
     .   ,'  <-------   JMAP   ------->   Threat Model',/
     .   ,92x,'ExpRatio')
  105 format('MUNITION',10x,'Requirements  Expenditures ',
     .   ' Req/ExpRatio  Expenditures  Requirements       NEW/OLD',/)
  200 format(3x,a15,3x,f9.3,5x,f9.3,f14.4,5(5x,f9.3))
  210 format(/,68x,'Goal      Solution      Priority')
  220 format(/,'MUNITION  -  Cost',49x,f6.4,8x,f6.4,13x,i1)
  230 format(/,'TARGET    -  Fraction Killed',71x,i1)
  240 format(3x,a15,6x,f6.4,36x,f6.4,8x,f6.4)
  250 format(/,'PLATFORM  -  Fraction Attrited',69x,i1)
  260 format(3x,a15,48x,f6.4,8x,f6.4)
  321 format(i1)
  322 format(i2)
      return
      end
c     (*   cal_max_aberration   *)
```

JMAP/Threat Pair

Code File : jmap.for

```
c     File Name :    jmap.for


c       Based on  JMAP  5/92  Martime  Module
c       Modified for adoption on HP-9000 HP-UX FORTRAN and MS PowerStation FORTRAN.
c               4/5/94     Roger Lynn


c       Arrangement  of .Program  Units :


c               open_files

c               lgpinp

c               g_con_overall_cost
c               g_con_targets_kill
c               g_con_platform_attrition
c               ph_con_max_platform_attrition
c               ph_con_munition_available
c               ph_con_doctrinal
c               ph_con_max_rate_of_fire
c               ph_con_target_counts
c               ph_con_upper_bound_targets_kill
c               ph_con_lower_bound_targets_kill

c               LGPSOLVE
c               START
c               SIMPLX
c               LGPRINT
c               PRTSOLU
c               TEX
c               XET

c               close_files


c     ---   based on   ---

C     ************************************************************************
C     *  INPUT FILES:                                                        *
C     *     1     MUNITION MODEL DESCRIPTORS - DETAILED INFO                 *
C     *                                                                      *
C     *  OUTPUT FILES:                                                       *
C     *     3     SOLUTION FILE - SUMMARY OF MUNITIONS USED AND COSTS        *
C     *    15     LGP REPORT FILE                                            *

c           9     input file for LGP SOLVER
c          10     constraint type
c          20     objective terms
c          30     coefficients
c          40     Right-Hand Sides

C     *  PROGRAMMED BY: M.L.STEIN  -  LOS ALAMOS NATIONAL LABORATORY         *
C     *  DATE:          NOVEMBER 1991                                        *
C     ************************************************************************
```

```
      subroutine   open_files(pass)

c     ***********************************************************************
c     *   This subroutine opens all the files used by the jmap module.   *
c     *   Referenced by:                                          jmap   *
c     ***********************************************************************

      include 'pair_1.inc'


      integer       pass
      character*1   ch_pass1
      character*2   ch_pass2
      character*14  filename

      if (pass .lt. 10) then
        write(ch_pass1,321)   pass
        filename = 'j_sum.'//ch_pass1
      else
        write(ch_pass2,322)   pass
        filename = 'j_sum.'//ch_pass2
      endif

      open(9,file='j_lgp.in')
      open(10,file='j_con')
      open(20,file='j_obj')
      open(30,file='j_coeff')
      open(40,file='j_rhs')

      open(15,file='j_lgp.out')
      open(3,file=filename)

      write(3,100)  target_overlap_confidence, filename

      return
  100 format(//,'Overlap Confidence = ',f5.3,22x,'File Name : ',a14)
  321 format(i1)
  322 format(i2)
 1000 format(a)
      end   ! open_files
```

```
      SUBROUTINE   LGPINP

C     ************************************************************
C     *  THIS ROUTINE FORMS THE INPUT FILE FOR THE LGP SOLVER  *
C     ************************************************************

      INCLUDE 'pair_2.inc'


      REWIND 9
      REWIND 10
      REWIND 20
      REWIND 30
      REWIND 40


C     ***********************************************************************
C     *  FIND THE NON-ZERO X(I,J,K) AND RECORD THEIR INDICIES           *
C     ***********************************************************************

      NBNZX=0

      DO 200 I=1,NBI

        DO 190 J=1,NBJ

          DO 185 K=1,NBK

            DO 180 L=1,NBL

C             ********************************************************************
C             *  CHECK IF TYPE J MUNITIONS FIRED FROM TYPE I PLATFORMS CAN  *
C             *  HARM TYPE K TARGETS                                        *
C             ********************************************************************

              IF (STK(I,J,K,L) .EQ. 1.E25) GO TO 180


              NBNZX=NBNZX + 1

              IF (NBNZX .GT. MAXX) THEN
                PRINT*,' '
                PRINT*,' '
                PRINT*,' THE NB OF NON-ZERO X's HAS EXCEEDED THE LIMIT',
     1                 ' MAXX = ',MAXX
                PRINT*,' '
                PRINT*,' INCREASE THE VALUE OF THE PARAMETER MAXX IN THE CODE.'
                PRINT*,' '
                PRINT*,' '
                STOP
              ENDIF

              TARGCHK(K)=TARGCHK(K) + 1

  180       CONTINUE
  185     CONTINUE
  190   CONTINUE
  200 CONTINUE


C     ***************************************************************************
C     *  CHECK THAT ALL TARGETS ARE ACCOUNTED FOR IN THE SET OF NON-ZERO E'S  *
C     ***************************************************************************

CCCC      PRINT*,' '
CCCC      PRINT*,' '
CCCC      PRINT*,' NB OF NON-ZERO X(I,J,K) = ',NBNZX
CCCC      PRINT*,' '
```

```
CCCC      PRINT*,' '
      TEST=0

      DO 230 K=1,NBK
      IF (TARGCHK(K) .EQ. 0) THEN
        TEST=1
        PRINT 1000,    ' TARGET NOT SHOT AT = ',TARG(K)
        WRITE(3,1000) ' TARGET NOT SHOT AT = ',TARG(K)
      ENDIF
  230 CONTINUE


      IF (TEST .EQ. 1) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' CHECK FILE 1 - SOME TARGETS CANNOT BE SHOT AT'
        PRINT*,' '
        PRINT*,' '
      ENDIF


      call   g_con_overall_cost
      call   g_con_targets_kill
      call   g_con_platform_attrition
      call   ph_con_max_platform_attrition
      call   ph_con_munition_available
      call   ph_con_doctrinal
      call   ph_con_max_rate_of_fire
      call   ph_con_target_counts
      call   ph_con_upper_bound_targets_kill
      call   ph_con_lower_bound_targets_kill

      NBROWS=ROW
      NBCOLS=NBNZX

      REWIND 10
      REWIND 20
      REWIND 30
      REWIND 40

      WRITE(9,1000) TITLE
 1000 FORMAT(A)
      WRITE(9,*) ' '
      WRITE(9,1010)
 1010 FORMAT('NB OF ROWS, COLUMNS, AND PRIORITIES')

      WRITE(9,*) NBROWS,NBCOLS,NBPRIORS

      WRITE(9,*) ' '
      WRITE(9,1020)
 1020 FORMAT('THE TYPE OF EACH CONSTRAINT  -  B,L,G,OR E')

      DO 10 LL=1,NBROWS
      READ(10,1000) TYPEC
      WRITE(9,1000) TYPEC
   10 CONTINUE

      WRITE(9,*) ' '
      WRITE(9,1030)
 1030 FORMAT('OBJECTIVE TERMS - DEVIATION TYPES,ROW,PRIORITY,WEIGHT')

      DO 20 LL=1,NBF20
      READ(20,1000) CLINE
      WRITE(9,1000) CLINE
   20 CONTINUE

      WRITE(9,1060) 0,0,0.0
 1060 FORMAT('END',2I7,F10.2)
```

```
      WRITE(9,*) ' '
      WRITE(9,1070)
 1070 FORMAT('CONSTRAINT COEFFS. - ROW,COLUMN, VALUE')

      DO 30 L=1,NBF30
      READ(30,1000) CLINE
      WRITE(9,1000) CLINE
 1080 FORMAT(2I7,E15.6)
   30 CONTINUE
      WRITE(9,1080) 0,0,0.0

      WRITE(9,*) ' '
      WRITE(9,1090)
 1090 FORMAT('RIGHT HAND SIDE VALUES OF CONSTRAINTS')

      DO 40 IR=1,NBROWS
      READ(40,1000) CLINE
      WRITE(9,1000) CLINE
   40 CONTINUE

      REWIND 9

      RETURN
      END
c             ! LGPINP
```

```
      SUBROUTINE   g_con_overall_cost

C     ************************************************************
C     *  THIS ROUTINE FORMS  the goal constraint on overall cost  *
c     *  CON1                                                     *
C     ************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      ROW=0
      TEST=0
      NBF20=0
      NBF30=0

      DO 10 L=1,NBL
        DO 20 I=1,NBI
          DO 30 J=1,NBJ
            DO 40 K=1,NBK
              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                TEST=1
              ENDIF

C             *******************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             *******************************************
c             FL=FRACLOST

c             coef =   CM(J) * R_E(J) * RPE(I,J,K) * STK(I,J,K,L)      !
c    .               + PLATCOST(I) * ATT(I,J,K) * STK(I,J,K,L)         ! if platform attrition cost is included in
c                                                                        overall cost
              coef =   CM(J) * R_E(J) * RPE(I,J,K) * STK(I,J,K,L)      ! platform attrition cost not included in
c                                                                        overall cost
              WRITE(30,3000) ROW,M,coef
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1
   40       CONTINUE
   30     CONTINUE
   20   CONTINUE
   10 CONTINUE

      IF (TEST .NE. 0) THEN

C       ****************************************
C       *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C       ****************************************
        WRITE(10,1000) 'B'
 1000   FORMAT(A)

C       ****************************************
C       *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C       ****************************************
        WRITE(20,2010) ROW,PREPRI(1),1.0
 2010   FORMAT('POS',2I7,F10.2)
        NBF20=NBF20 + 1

C       ****************************************
C       *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C       ****************************************
        WRITE(40,4000) COSTGOAL
 4000   FORMAT(F15.5)
      ENDIF

      RETURN
      END
c             !  g_con_overall_cost
```

```
      SUBROUTINE   g_con_targets_kill

C     ***************************************************************************************
C     *  THIS ROUTINE FORMS  the goal constraints on fractions of targets killed by target type  *
c     *  CON7                                                                                   *
C     ***************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      NBC7ROWS=0

      DO 10 K=1,NBK
        DO 20 L=1,NBL

        TEST=0

          DO 30 I=1,NBI
            DO 40 J=1,NBJ

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 47
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                NBC7ROWS=NBC7ROWS + 1
                TEST=1
              ENDIF

C             ******************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             ******************************************
              WRITE(30,3000) ROW, M, 1.0                              !!
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1

   47         DO 45 LP=1,L-1
                M=TEX(I,J,K,LP)
                IF (M .EQ. 0) GO TO 45
                IF (TEST .EQ. 0) THEN
                  ROW=ROW + 1
                  TEST=1
                ENDIF

C               ***********************************************************
C               *  WRITE CONSTRAINT COEFFS FOR THE SOLVER                 *
c               *  (from the surviving targets of the right-hand-side)    *
C               ***********************************************************
                WRITE(30,3000) ROW, M, FTBK(K,L)                      !!
                NBF30=NBF30 + 1
   45         CONTINUE

   40       CONTINUE
   30     CONTINUE

        IF (TEST .NE. 0) THEN

C         ****************************************
C         *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C         ****************************************
          WRITE(10,1000) 'B'
 1000     FORMAT(A)

C         ****************************************
C         *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C         ****************************************
          WRITE(20,2000) ROW, PREPRI(2), FOM_T(k)                     !!
 2000     FORMAT('NEG',2I7,F10.2)
          NBF20=NBF20 + 1
```

```
C         ****************************************
C         *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C         ****************************************
          WRITE(40,4000) FTBK(K,L)*TQ(K)
 4000     FORMAT(F15.5)
        ENDIF

   20   CONTINUE
   10 CONTINUE

      RETURN
      END
c             !   g_con_targets_kill
```

```
      SUBROUTINE   g_con_platform_attrition

C     ***************************************************************************************************
C     *  THIS ROUTINE FORMS  the goal constraints on fractions of platforms attrited by platform type  *
c     *  CON8                                                                                           *
C     ***************************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      NBC8ROWS=0

      DO 10 I=1,NBI

        TEST=0

        DO 20 L=1,NBL
          DO 30 J=1,NBJ
            DO 40 K=1,NBK

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                NBC8ROWS=NBC8ROWS + 1
                TEST=1
              ENDIF

C             ********************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             ********************************************
              IF (ATT(I,J,K) .GT. 0) THEN
                coef = ATT(I,J,K) * STK(I,J,K,L)                    !!
                WRITE(30,3000) ROW, M, coef
 3000           FORMAT(2I7,E15.6)
                NBF30=NBF30 + 1
              ENDIF

   40       CONTINUE
   30     CONTINUE
   20   CONTINUE

        IF (TEST .NE. 0) THEN

C         ****************************************
C         *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C         ****************************************
          WRITE(10,1000) 'B'                                        !!
 1000     FORMAT(A)

C         ****************************************
C         *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C         ****************************************
          WRITE(20,2000) ROW, PREPRI(3), FOM_P(I)                   !!
 2000     FORMAT('POS',2I7,F10.2)
          NBF20=NBF20 + 1

C         ****************************************
C         *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C         ****************************************
          WRITE(40,4000) FATTR(I)*PQ(I)
 4000     FORMAT(F15.5)
        ENDIF

   10 CONTINUE

      RETURN
      END
c             !   g_con_platform_attrition
```

```
      SUBROUTINE   ph_con_max_platform_attrition

C     ***************************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on maximum platform attritions by platform type  *
c     *  CON3                                                                                           *
C     ***************************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      DO 10 I=1,NBI

        TEST=0

        DO 20 L=1,NBL
          DO 30 J=1,NBJ
            DO 40 K=1,NBK

              IF (TARGTYPE(K).EQ.'FALSE' .OR. TARGTYPE(K).EQ.'false')
                                                          GO TO 40

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                TEST=1
              ENDIF

C             *****************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             *****************************************
              IF (ATT(I,J,K) .GT. 0) THEN
                coef = ATT(I,J,K) * STK(I,J,K,L)                    !!
                WRITE(30,3000) ROW, M, coef
 3000           FORMAT(2I7,E15.6)
                NBF30=NBF30 + 1
              ENDIF

   40       CONTINUE
   30     CONTINUE
   20   CONTINUE

        IF (TEST .NE. 0) THEN
C         ****************************************
C         *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C         ****************************************
          WRITE(10,1000) 'L'
 1000     FORMAT(A)

C         ****************************************
C         *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C         ****************************************
C         (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C         ****************************************
C         *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C         ****************************************
          WRITE(40,4000)  pq(i)
 4000     FORMAT(F15.5)
        ENDIF

   10 CONTINUE

      RETURN
      END
c             !   ph_con_max_platform_attrition
```

```
      SUBROUTINE   ph_con_munition_available

C     ********************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on munition stockpile by munition type  *
c     *  CON5                                                                                  *
C     ********************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      NBC5ROWS=0

      DO 10 J=1,NBJ

        TEST=0

        DO 20 L=1,NBL
          DO 30 I=1,NBI
            DO 40 K=1,NBK

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                NBC5ROWS=NBC5ROWS + 1
                TEST=1
              ENDIF

C             ******************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             ******************************************
              FL=FRACLOST
              COEF = R_E(J) * RPE(I,J,K) * STK(I,J,K,L)
              WRITE(30,3000) ROW, M, COEF
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1

   40       CONTINUE
   30     CONTINUE
   20   CONTINUE

      IF (TEST .NE. 0) THEN
C       ****************************************
C       *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C       ****************************************
        WRITE(10,1000) 'L'
 1000   FORMAT(A)

C       ****************************************
C       *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C       ****************************************
C       (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C       ****************************************
C       *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C       ****************************************
        WRITE(40,4000) STOCK(J)
 4000   FORMAT(F15.5)
      ENDIF

   10 CONTINUE

      RETURN
      END
c             !   ph_con_munition_available
```

```
      SUBROUTINE   ph_con_doctrinal

C     ***************************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on doctrinal expenditures by munition type  *
c     *  CON9                                                                                     *
C     ***************************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      NBC9ROWS=0

      DO 10 L=1,NBL
        DO 20 J=1,NBJ

          IF (DOCTRIN(J) .EQ. 0) GO TO 20

          TEST=0

          DO 30 I=1,NBI
            DO 40 K=1,NBK

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                NBC9ROWS=NBC9ROWS + 1
                TEST=1
              ENDIF

C             ******************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             ******************************************
              coef = RPE(i,j,k) * STK(i,j,k,l)                          !!
              WRITE(30,3000)  ROW, M, coef
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1

   40       CONTINUE
   30     CONTINUE

          IF (TEST .NE. 0) THEN

C           ****************************************
C           *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C           ****************************************
            WRITE(10,1000) 'G'
 1000       FORMAT(A)

C           ****************************************
C           *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C           ****************************************
C           (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C           ****************************************
C           *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C           ****************************************
            WRITE(40,*)   DOCTRIN(J) * TIME(L)
 4000       FORMAT(F15.5)
          ENDIF

   20   CONTINUE
   10 CONTINUE

      RETURN
      END
c             !   ph_con_doctrinal
```

```
      SUBROUTINE   ph_con_max_rate_of_fire

C     ***************************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on Rate-of-Fire by platform type and munition type  *
c     *  CON4                                                                                             *
C     ***************************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      NBC4ROWS=0

      DO 10 L=1,NBL
        DO 20 I=1,NBI
          DO 30 J=1,NBJ

          TEST=0
          RTHS=0
          DO 40 K=1,NBK
            M=TEX(I,J,K,L)
            IF(M.EQ.0) GO TO 40
            IF(TEST.EQ.0) THEN
              ROW=ROW + 1
              NBC4ROWS=NBC4ROWS + 1
              TEST=1
            ENDIF
C           *****************************************
C           *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C           *****************************************
            coef = RPE(I,J,K) * STK(I,J,K,L)                          !!
            WRITE(30,3000) ROW, M, coef
 3000       FORMAT(2I7,E15.6)
            NBF30=NBF30 + 1

   40     CONTINUE

          IF (TEST .NE. 0) THEN
C           ******************************************
C           *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C           ******************************************
            WRITE(10,1000) 'L'
 1000       FORMAT(A)

C           ******************************************
C           *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C           ******************************************
C           (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C           ******************************************
C           *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C           ******************************************
            RTHS = PQ(I) * ROF(I,J) * TIME(L)                         !!
            WRITE(40,4000) RTHS
 4000       FORMAT(F15.5)
          ENDIF

   30     CONTINUE
   20   CONTINUE
   10 CONTINUE

      RETURN
      END
c             !   ph_con_max_rate_of_fire
```

```
      SUBROUTINE   ph_con_target_counts

C     ***************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on maximum target kills by target type  *
c     *  CON6                                                                                 *
C     ***************************************************************************************

      INCLUDE 'pair_2.inc'
      integer  tex

      NBC6ROWS=0

      DO 10 K=1,NBK

        TEST=0

        DO 20 L=1,NBL
          DO 30 I=1,NBI
            DO 40 J=1,NBJ

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                NBC6ROWS=NBC6ROWS + 1
                TEST=1
              ENDIF

C             *******************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             *******************************************
              WRITE(30,3000) ROW, M, 1.0
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1

   40       CONTINUE
   30     CONTINUE
   20   CONTINUE

      IF (TEST .NE. 0) THEN

C       ****************************************
C       *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C       ****************************************
        WRITE(10,1000) 'L'
 1000   FORMAT(A)

C       ****************************************
C       *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C       ****************************************
C       (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C       ****************************************
C       *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C       ****************************************
        WRITE(40,4000) TQ(K)
 4000   FORMAT(F15.5)
      ENDIF

   10 CONTINUE

      RETURN
      END
c             !   ph_con_target_counts
```

```
      SUBROUTINE   ph_con_upper_bound_targets_kill

C     ********************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on maximum target kills by target type  *
c     *  CON6                                                                                  *
C     ********************************************************************************************

      INCLUDE 'pair_2.inc'
      common /BOUNDS/ upper_bound(MAXI,MAXK), lower_bound(MAXI,MAXK)
        real          upper_bound, lower_bound
      integer         tex

      DO 10 I=1,NBI
        DO 20 K=1,NBK

          TEST=0

          DO 30 L=1,NBL
            DO 40 J=1,NBJ

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                TEST=1
              ENDIF

C             *******************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             *******************************************
              WRITE(30,3000) ROW, M, 1.0
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1

   40       CONTINUE
   30     CONTINUE

        IF (TEST .NE. 0) THEN

C         ****************************************
C         *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C         ****************************************
          WRITE(10,1000) 'L'
 1000     FORMAT(A)

C         ****************************************
C         *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C         ****************************************
C         (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C         ****************************************
C         *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C         ****************************************
          WRITE(40,4000)   upper_bound(I,K)
 4000     FORMAT(F15.5)

        ENDIF

   20   CONTINUE
   10 CONTINUE

      RETURN
      END
c             !   ph_con_upper_bound_targets_kill
```

```
      SUBROUTINE   ph_con_lower_bound_targets_kill

C     ********************************************************************************************
C     *  THIS ROUTINE FORMS  the physical constraints on maximum target kills by target type  *
c     *  CON6                                                                                  *
C     ********************************************************************************************

      INCLUDE 'pair_2.inc'
      common /BOUNDS/ upper_bound(MAXI,MAXK), lower_bound(MAXI,MAXK)
        real          upper_bound, lower_bound
      integer         tex

      DO 10 I=1,NBI
        DO 20 K=1,NBK

          TEST=0

          DO 30 L=1,NBL
            DO 40 J=1,NBJ

              M=TEX(I,J,K,L)
              IF(M.EQ.0) GO TO 40
              IF(TEST.EQ.0) THEN
                ROW=ROW + 1
                TEST=1
              ENDIF

C             *****************************************
C             *  WRITE CONSTRAINT COEFFS FOR THE SOLVER  *
C             *****************************************
              WRITE(30,3000) ROW, M, 1.0
 3000         FORMAT(2I7,E15.6)
              NBF30=NBF30 + 1

   40       CONTINUE
   30     CONTINUE

        IF (TEST .NE. 0) THEN

C         ****************************************
C         *  WRITE CONSTRAINT TYPE FOR THE SOLVER  *
C         ****************************************
          WRITE(10,1000) 'G'
 1000     FORMAT(A)

C         ****************************************
C         *  WRITE OBJECTIVE TERMS FOR THE SOLVER  *
C         ****************************************
C         (Physical CONSTRAINT DOES NOT ENTER THE OBJECTIVE FUNCTION)

C         ****************************************
C         *  WRITE RIGHT HAND SIDES FOR THE SOLVER  *
C         ****************************************
          WRITE(40,4000)   lower_bound(I,K)
 4000     FORMAT(F15.5)

        ENDIF

   20   CONTINUE
   10 CONTINUE

      RETURN
      END
c             !   ph_con_lower_bound_targets_kill
```

```
c*****************************************************************************
c*            Schniederjans  Linear  Goal  Programming  Solver              *
c*****************************************************************************


      SUBROUTINE   LGPSOLVE(pass)

C     ******************************************************
C     *  THE ROUTINE CALLS THE LPG SOLVER OF SCHNIEDERJANS  *
C     *  AND PRINTS THE RESULTS                             *
C     ******************************************************

      INCLUDE 'pair_2.inc'

      integer       pass

      DIMENSION AVTIME(2)

c     time0 = secnds (0.0)

      CALL START
      CALL SIMPLX
      CALL LGPRINT

C     ************************
C     *  PRINT THE SOLUTION  *
C     ************************

      CALL PRTSOLU(pass)


c     time1 = secnds (time0)


      PRINT*,' '
      PRINT*,'   NB ROWS =',NBROWS,'          NB COLUMNS =',NBCOLS
      PRINT*,' '
      PRINT*,' '

      WRITE(3,*) ' '
      WRITE(3,*) ' NB ROWS = ',NBROWS,'  NB COLUMNS = ',NBCOLS
      WRITE(3,*) ' '
c     write(3,*) ' SOLUTION TIME = ',TIME1,' SECS.'
      WRITE(3,*) ' '

      RETURN
      END
c             ! LGPSOLVE
```

```
      SUBROUTINE   START

C     ************************************************************
C     *  THIS ROUTINE READS INPUT AND INITIATES WORKING MATRICES  *
C     *           REF :  "LINEAR GOAL PROGRAMMING"                *
C     *                    BY M.J. SCHNIEDERJANS                  *
C     ************************************************************



      INCLUDE 'pair_2.inc'


      CHARACTER*3 KSIGN
      CHARACTER*100 HEADING

      READ(9,1000) HEADING
      READ(9,1000) HEADING
      READ(9,1000) HEADING
 1000 FORMAT(A)

      READ(9,*) NBROWS,NVAR,NBPRIORS

      IF (NBROWS.LE.0 .OR. NVAR.LE.0 .OR. NBPRIORS.LE.0) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' NB OF CONSTRS., VARIS., OR PRIORS. INCORRECT'
        PRINT*,' CHECK INPUT FILE 9'
        PRINT*,' '
        PRINT*,' '
        STOP
      ENDIF

      NBCOLS=NBROWS + NVAR

      IF (NBROWS .GT. MAXROWS) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' NB OF ROWS IS TOO LARGE'
        PRINT*,' PRESENT MAX = ',MAXROWS
        PRINT*,' INCREASE VALUE OF THE PARAMETER MAXROWS IN THE CODE'
        PRINT*,' '
        PRINT*,' '
        STOP
      ENDIF

      IF (NBCOLS .GT. MAXCOLS) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' NB OF COLUMNS IS TOO LARGE'
        PRINT*,' PRESENT MAX = ',MAXCOLS
        PRINT*,' INCREASE VALUE OF THE PARAMETER MAXCOLS IN THE CODE'
        PRINT*,' '
        PRINT*,' '
        STOP
      ENDIF

      DO 2 I=1,NBROWS
      DO 1 J=1,NBCOLS
      BASIS(I,J)=0
      INDEX=J - NVAR
      IF (INDEX .EQ. I) BASIS(I,J)=1.0
    1 CONTINUE
      IND=I + NBCOLS
      IBASIC(I)=IND
    2 CONTINUE

      DO 3 J=1,NBCOLS
```

```
      JCOL(J)=J
    3 CONTINUE

      KEND=NBPRIORS + 1

      DO 6 K=1,KEND

      DO 4 J=1,NBCOLS
      VALC(K,J)=0
    4 CONTINUE

      DO 5 I=1,NBROWS
      VALB(K,I)=0
    5 CONTINUE

    6 CONTINUE

      KTEST=0

      READ(9,1000) HEADING
      READ(9,1000) HEADING

      DO 7 I=1,NBROWS

      READ(9,1000) ISIGN(I)
    7 CONTINUE

      DO 10 I=1,NBROWS
      IF (ISIGN(I) .EQ. 'B') GO TO 10
      IF (ISIGN(I) .EQ. 'E') THEN
        KTEST=1
        INDEX=I + NVAR
        VALB(1,I)=1.0
        VALC(1,INDEX)=1.0
        JCOL(INDEX)=0
      ELSE IF (ISIGN(I) .EQ. 'G') THEN
        INDEX=I + NVAR
        KTEST=1
        VALC(1,INDEX)=1.0
        JCOL(INDEX)=0
      ELSE IF (ISIGN(I) .EQ. 'L') THEN
        KTEST=1
        VALB(1,I)=1.0
      ELSE
        PRINT*,' '
        PRINT*,' '
        PRINT*,' CONSTRAINT TYPE  .NE. E,G,L, OR B'
        PRINT*,' CHECK INPUT FILE 9'
        PRINT*,' '
        PRINT*,' '
        STOP
      ENDIF

   10 CONTINUE

      IF (KTEST .EQ. 1) NBPRIORS=NBPRIORS + 1

      READ(9,1000) HEADING
      READ(9,1000) HEADING

   11 READ(9,2000) KSIGN,I,K,WGT
 2000 FORMAT(A,2I7,F10.2)

      IF (KSIGN .NE. 'END') THEN
        IF (KTEST .EQ. 1) K=K+1
        IF (KSIGN .EQ. 'POS') THEN
          VALB(K,I)=WGT
        ELSE IF (KSIGN .EQ. 'NEG') THEN
          INDEX=I + NVAR
```

```
        VALC(K,INDEX)=WGT
      ELSE IF (KSIGN .EQ. 'END') THEN
        GO TO 15
      ELSE
        PRINT*,' '
        PRINT*,' '
        PRINT*,' DEVIATION TYPE  .NE. "POS" OR "NEG" '
        PRINT*,' CHECK INPUT FILE 9'
        PRINT*,' '
        PRINT*,' '
        STOP
      ENDIF
      GO TO 11
    ENDIF

 15 READ(9,1000) HEADING
    READ(9,1000) HEADING

 17 READ(9,*) I,J,AIJ

    IF (I .NE. 0) THEN
      BASIS(I,J)=AIJ
      GO TO 17
    ENDIF

    READ(9,1000) HEADING
    READ(9,1000) HEADING
    DO 23 I=1,NBROWS
    READ(9,*) PRHS(I)
    IF (PRHS(I) .LT. 0) THEN
      PRINT*,' '
      PRINT*,' '
      PRINT*,' ALL RHS MUST BE POSITIVE'
      PRINT*,' CHECK INPUT FILE 9 - MULTIPLY CONSTRAINT BY  -1'
      PRINT*,' '
      PRINT*,' '
      STOP
    ELSE IF (PRHS(I) .EQ. 0) THEN
      PRHS(I)=1.D-12
    ENDIF

    RHS(I)=-PRHS(I)

 23 CONTINUE

    DO 31 J=1,NBCOLS

    IF (JCOL(J) .EQ. 0) THEN
      DO 30 I=1,NBROWS
      BASIS(I,J)=0.0
 30   CONTINUE
    ENDIF

 31 CONTINUE

    RETURN
    END
c             ! START
```

```
      SUBROUTINE   SIMPLX

C     ************************************************
C     *                                              *
C     *  THIS ROUTINE PERFORMS THE SIMPLEX OPERATION  *
C     *                                              *
C     *       REF: "LINEAR GOAL PROGRAMMING"         *
C     *             BY M.J. SCHNIEDERJANS            *
C     *                                              *
C     ************************************************


      INCLUDE 'pair_2.inc'


      DIMENSION JFAIL(MAXROWS),JPICK(MAXCOLS)
      DOUBLE PRECISION ZVAL(MAXPRI,MAXCOLS),PIV,rRMIN,AIJ,ZVALUE,ZETA
      DOUBLE PRECISION THETA,DUMMY

      ITER500=0

      KEND=NBPRIORS + 1

      DO 16 J=1,NBCOLS
      JPICK(J)=KEND
   16 CONTINUE

      DO 18 J=1,NBCOLS
      DO 17 K=1,NBPRIORS
      IF (VALC(K,J) .GT. 1.D-10) THEN
        JPICK(J)=K
      ENDIF
   17 CONTINUE
   18 CONTINUE

      ITER=0

    1 KEYROW=0
      KEYCOL=0
      KUNACH=0

      DO 2 I=1,NBROWS
      JFAIL(I)=1
    2 CONTINUE

C     IDENTIFY THE HIGHEST UNACHIEVED PRIORITY

      DO 4 K=1,NBPRIORS
      DO 3 I=1,NBROWS
      IF (VALB(K,I) .GT. 1.D-10) THEN
        KUNACH=K
        GO TO 11
      ENDIF
    3 CONTINUE
    4 CONTINUE

C     IDENTIFY THE MOST NEGATIVE RHS

   11 rRMIN = -1.D-10

      DO 12 I=1,NBROWS
        IF (RHS(I) .GT. rRMIN) GO TO 12
        IF (JFAIL(I) .EQ. 0) GO TO 12
        KEYROW=I
        rRMIN=RHS(I)
   12 CONTINUE
```

```
C     IF KEYROW .EQ. 0, THEN ALL RHS ARE  .GE. 0

      IF (KEYROW .NE. 0) THEN
C       PATH FOR NEG RHS
        AIJ=1.D-8
        DO 25 M=1,KEND
        L=KEND - M + 1
        DO 24 J=1,NBCOLS
        IF (JCOL(J) .EQ. 0) GO TO 24
        IF (JPICK(J) .LT. L) GO TO 24
        IF (BASIS(KEYROW,J) .LE. AIJ) GO TO 24
        AIJ=BASIS(KEYROW,J)
        KEYCOL=J
   24   CONTINUE
        IF (KEYCOL .GT. 0) GO TO 40
   25   CONTINUE
        JFAIL(KEYROW)=0
        GO TO 11

      ENDIF

      IF (KUNACH .EQ. 0) THEN
        PRINT*,' '
        PRINT*,' ALL GOALS ACHIEVED'
        PRINT*,' '
                                                        goto 999 !!
      ENDIF

      KFIN=KUNACH

C     THE ZJ MATRIX IS DEVELOPED

      DO 33 K=KUNACH,NBPRIORS
      DO 32 J=1,NBCOLS
      ZVAL(K,J)=0.0
      IF (JCOL(J) .EQ. 0) GO TO 32
      IF (JPICK(J) .LT. KFIN) GO TO 32
      DO 31 I=1,NBROWS
      IF (VALB(K,I) .LE. 1.D-10) GO TO 31
      IF (DABS(BASIS(I,J)) .LE. 1.D-10) GO TO 31
CCCC  IF (ABS(BASIS(I,J)) .LE. 1.D-10) GO TO 31
      ZVAL(K,J)=ZVAL(K,J) + VALB(K,I) * BASIS(I,J)
   31 CONTINUE
      ZVAL(K,J)=ZVAL(K,J) + VALC(K,J)
   32 CONTINUE
   33 CONTINUE

      ZVALUE=-1.D-8

      DO 36 K=KUNACH,NBPRIORS
      DO 35 J=1,NBCOLS
      IF (JCOL(J) .EQ. 0) GO TO 35
      IF (JPICK(J) .LT. KFIN) GO TO 35
      IF (ZVAL(K,J) .GE. ZVALUE) GO TO 35
      IF (K .LE. KUNACH) GO TO 39
      M=K-1
      DO 34 L=1,M
      IF (ZVAL(L,J) .GE. 1.D-8) GO TO 35
   34 CONTINUE

   39 ZVALUE=ZVAL(K,J)
      KEYCOL=J
   35 CONTINUE
      IF (KEYCOL .GT. 0) GO TO 37
      KFIN=KFIN + 1
   36 CONTINUE

      IF (KEYCOL .EQ. 0)                                goto 999 !!
```

```
   37 THETA=1.D100

      DO 38 I=1,NBROWS
      IF (BASIS(I,KEYCOL) .GE. -1.D-10) GO TO 38
      IF (RHS(I) .LE. -1.D-10) GO TO 38
      IF (RHS(I) .LE. 1.D-10) RHS(I)=1.D-10
      ZETA=-RHS(I)/BASIS(I,KEYCOL)
      IF (ZETA .GE. THETA) GO TO 38
      THETA=ZETA
      KEYROW=I
   38 CONTINUE

      IF (KEYROW .LE. 0)                                    goto 999 !!

   40 PIV=BASIS(KEYROW,KEYCOL)

      DO 43 I=1,NBROWS
      IF (I .EQ. KEYROW) GO TO 43
      IF (DABS(BASIS(I,KEYCOL)) .LE. 1.D-10) GO TO 43
      IF (DABS(RHS(KEYROW)) .LE. 1.D-10) GO TO 41
CCCC  IF (ABS(BASIS(I,KEYCOL)) .LE. 1.D-10) GO TO 43
CCCC  IF (ABS(RHS(KEYROW)) .LE. 1.D-10) GO TO 41
      RHS(I)=RHS(I) - (RHS(KEYROW)/PIV) * BASIS(I,KEYCOL)

   41 DO 42 J=1,NBCOLS
      IF (J .EQ. KEYCOL) GO TO 42
      IF (DABS(BASIS(KEYROW,J)) .LE. 1.D-10) GO TO 42
CCCC  IF (ABS(BASIS(KEYROW,J)) .LE. 1.D-10) GO TO 42
      BASIS(I,J)=BASIS(I,J) - (BASIS(I,KEYCOL)/PIV)*BASIS(KEYROW,J)
   42 CONTINUE
      BASIS(I,KEYCOL)=BASIS(I,KEYCOL)/PIV
   43 CONTINUE

      IF (DABS(RHS(KEYROW)) .GT. 1.D-10) THEN
CCCC  IF (ABS(RHS(KEYROW)) .GT. 1.D-10) THEN
        RHS(KEYROW)=-RHS(KEYROW)/PIV
      ENDIF

   44 DO 45 J=1,NBCOLS
      IF (J .EQ. KEYCOL) GO TO 45
      IF (DABS(BASIS(KEYROW,J)) .LE. 1.D-10) GO TO 45
CCCC  IF (ABS(BASIS(KEYROW,J)) .LE. 1.D-10) GO TO 45
      BASIS(KEYROW,J)=-BASIS(KEYROW,J)/PIV
   45 CONTINUE

      BASIS(KEYROW,KEYCOL)=1./PIV
      INDEX=JCOL(KEYCOL)
      JCOL(KEYCOL)=IBASIC(KEYROW)
      IBASIC(KEYROW)=INDEX

      DO 46 K=1,NBPRIORS
      DUMMY=VALB(K,KEYROW)
      IF (DUMMY .GE. 1.D-8) JPICK(KEYCOL)=K
      VALB(K,KEYROW)=VALC(K,KEYCOL)
      VALC(K,KEYCOL)=DUMMY
   46 CONTINUE

      IF (KTEST .NE. 1) GO TO 51
      IF (VALC(1,KEYCOL) .EQ. 0.0) GO TO 51
      JCOL(KEYCOL)=0

      DO 50 I=1,NBROWS
      BASIS(I,KEYCOL)=0.0
   50 CONTINUE

   51 ITER=ITER + 1
c     PRINT*,' SOLUTION ITERATION ',ITER                     !!
      IF (MOD(ITER,2500) .EQ. 0) THEN
        PRINT*,' '
```

```
      PRINT*,' '
      PRINT*,' '
      PRINT*,' '
      PRINT*,' ******************************************'
      PRINT*,' '
      PRINT*,' MESSAGE FROM SUBROUTINE   SIMPLEX :'
      PRINT*,' '
      PRINT*,'    THERE HAVE BEEN 2500 ITERATIONS'
      PRINT*,'    THE LGP SOLUTION IS QUESTIONABLE'
      PRINT*,' '
      PRINT*,' ******************************************'
      PRINT*,' '
      PRINT*,' DO YOU WANT TO TRY 2500 MORE ?  Y/N'
      PRINT*,' '
      READ*,YESNO
      IF (YESNO.EQ.'Y' .OR. YESNO.EQ.'y') GO TO 1
      ITER500=1
                                                    goto 999 !!
    ENDIF

    GO TO 1

999 print*, ' '
    print*, 'Number of  JMAP  iterations  =', iter
    return
    END
c           ! SIMPLX
```

```
      SUBROUTINE   LGPRINT

C     ********************************************
C     *  THIS ROUTINE REPORTS THE FINAL SOLUTION  *
C     ********************************************

      INCLUDE 'pair_2.inc'


      DIMENSION X(MAXCOLS),POSD(MAXCOLS),RNEGD(MAXCOLS)


C     THIS SECTION REPORTS VALUES OF ALL MODEL VARIABLES

      REWIND 15

      DO 1 J=1,NVAR
      X(J)=0.0
    1 CONTINUE

      DO 2 I=1,NBROWS
      POSD(I)=0.0
      RNEGD(I)=0.0
    2 CONTINUE

      DO 12 I=1,NBROWS
      IVAR=IBASIC(I)
      IF (IVAR .GT. NBCOLS) THEN
        IND=IVAR - NBCOLS
        POSD(IND)=RHS(I)
      ELSE IF (IVAR .GT. NVAR) THEN
        IND=IVAR - NVAR
        RNEGD(IND)=RHS(I)
      ELSE
        X(IVAR)=RHS(I)
      ENDIF
   12 CONTINUE

      WRITE(15,1000) ITER
c     WRITE(3,1000) ITER                                          !!
 1000 FORMAT(I6,'    ITERATIONS')
      WRITE(15,1001)
 1001 FORMAT(' DECISION VARIABLES')
      WRITE(15,1002)
 1002 FORMAT(/,' VARIABLE      VALUE')

      DO 15 J=1,NVAR
      WRITE(15,1003) J ,X(J)
 1003 FORMAT(1X,I5,1PE15.5)
   15 CONTINUE

      WRITE(15,1004)
 1004 FORMAT(///,' ANALYSIS OF DEVIATIONS FROM GOALS')
      WRITE(15,1005)
 1005 FORMAT(/,'  ROW',10X,'RHS-VALUES',5X,'POSITIVE DEVIATION',
     1   2X,'NEGATIVE DEVIATION')

      DO 16 I=1,NBROWS
      WRITE(15,1006) I,PRHS(I),POSD(I),RNEGD(I)
 1006 FORMAT(I5,3(1PE20.5))
   16 CONTINUE

C     THIS SECTION REPORTS ON PRIORITY LEVELS

      WRITE(15,1013)
 1013 FORMAT(///,' ANALYSIS OF THE OBJECTIVE FUNCTION')
      WRITE(15,1014)
 1014 FORMAT(//,' PRIORITY',9X,'UNDERACHIEVEMENT')
```

```
      KTOTAL=NBPRIORS + 1
      DO 52 K=1,NBPRIORS
      KVAL=KTOTAL - K
      M=KVAL
      IF (KTEST .EQ. 1) M=KVAL - 1
      ZVALUE=0.0
      DO 50 I=1,NBROWS
      IF (VALB(KVAL,I) .LE. 1.E-14) GO TO 50
      ZVALUE=ZVALUE + VALB(KVAL,I)*RHS(I)
   50 CONTINUE
      IF (KTEST .EQ. 0) GO TO 51
      IF (M .GT. 0) GO TO 51
      WRITE(15,1015) ZVALUE
 1015 FORMAT(' ARTIFICIAL',1PE20.5)

      IF (ZVALUE .GT. 1.E-5) THEN
        PRINT*,' '
        PRINT*,' '
        PRINT*,' ***********************************'
        PRINT*,' *                                 *'
        PRINT*,' *  WARNING !   WARNING !          *'
        PRINT*,' *                                 *'
        PRINT*,' *                                 *'
        PRINT*,' *  THE SOLUTION IS INFEASIBLE     *'
        PRINT*,' *                                 *'
        PRINT*,' *  CHECK THE SCENARIO INPUT DATA  *'
        PRINT*,' *                                 *'
        PRINT*,' *                                 *'
        PRINT*,' ***********************************'
        PRINT*,' '
        PRINT*,' '

        WRITE(3,*)' '
        WRITE(3,*)' '
        WRITE(3,*)' ***********************************'
        WRITE(3,*)' *                                 *'
        WRITE(3,*)' *  WARNING !   WARNING !          *'
        WRITE(3,*)' *                                 *'
        WRITE(3,*)' *                                 *'
        WRITE(3,*)' *  THE SOLUTION IS INFEASIBLE     *'
        WRITE(3,*)' *                                 *'
        WRITE(3,*)' *  CHECK THE SCENARIO INPUT DATA  *'
        WRITE(3,*)' *                                 *'
        WRITE(3,*)' *                                 *'
        WRITE(3,*)' ***********************************'
        WRITE(3,*)' '
        WRITE(3,*)' '
      ENDIF

      GO TO 52

   51 WRITE(15,1016) M,ZVALUE
 1016 FORMAT(I4,7X,1PE20.5)
   52 CONTINUE

      RETURN
      END
c             ! LGPRINT
```

```
      SUBROUTINE   PRTSOLU(pass)

C     ************************************************************
C     *  THIS ROUTINE WRITES OUT SUMMARIZED INFORMATION OBTAINED  *
C     *  FROM THE LGP REPORT FILE 15                              *
C     ************************************************************

      INCLUDE 'pair_2.inc'

      integer       pass
      character*1   ch_pass1
      character*2   ch_pass2
      character*14  filename

      DIMENSION     DP(MAXROWS), DN(MAXROWS)
      real          expend(MAXI,MAXJ)
      real          KG(MAXK), AG(MAXI), TIK(MAXI,MAXK)
      real          munition_req_cost(MAXJ), platform_att_cost(MAXI)
      real          tot_plat_att_cost
      real          NBRNDS

      CHARACTER*17 SP17
      CHARACTER*21 SP21

      DATA SP17/'                '/
      DATA SP21/'                    '/



      REWIND 15

      ITEST=0
      N=0

C     ***************************************
C     *  initialize THE MUNITION SUM ARRAY  *
C     ***************************************

   25 DO 27 J=1,NBJ
        RNDSUM(J)=0.0
        reqsum(J)=0
        DO 26 I=1,NBI
          expend(I,J)=0
          require(i,j)=0
   26   CONTINUE
   27 CONTINUE

C     ******************************************************
C     *  initialize THE TARGET KILL AND COST/KILL ARRAYS  *
C     ******************************************************

      DO 30 K=1,NBK
        TOTKILL(K)=0.0
        TOTCOST(K)=0.0
        do  I=1,NBI
          TIK(I,K) = 0.0
          do  J=1,NBJ
            TIJK(I,J,K) = 0.0
          enddo
        enddo
        DO 28 J=1,NBJ
          SUMKILL(J,K)=0.0
   28   CONTINUE
        DO 29 L=1,MAXL
          TOTLKILL(K,L)=0.0
   29   CONTINUE
   30 CONTINUE
```

```
      DO 33 I=1,NBI
        TOTATT(I)=0
   33 CONTINUE

      READ(15,1100) LINE
      READ(15,1100) LINE
      READ(15,1100) LINE
      READ(15,1100) LINE

   35 READ(15,1100) LINE
      IF (LINE(1:10) .EQ. '          ') GO TO 100

      READ(LINE,1900) MM,X1
 1900 FORMAT(I6,E15.5)

      IF (X1 .LT. 1.E-4) GO TO 35

      CALL XET(MM,I,J,K,L)

      xkilled  = X1                                          !!  here, decision variables are targets killed
      TIJK(i,j,k) = xkilled

      NBRNDS =  RPE(i,j,k) * STK(i,j,k,l) * xkilled

      expend(i,j) =  expend(i,j)  +  NBRNDS                  !!  sum up expenditures of munition J by platform I
      require(i,j)  =  R_E(j) * expend(i,j)                  !!  sum up requirements of munition J by platform I


C     ****************************************************************
C     *  SUM  munition requirement COST  TO KILL TARGETS OF TYPE K  *
C     ****************************************************************

      TOTCOST(K) = TOTCOST(K)  +  R_E(j) * NBRNDS * CM(J)


C     ******************************
C     *  SUM MUNITIONS OF TYPE J  *
C     ******************************

      RNDSUM(J) = RNDSUM(J) + NBRNDS
      reqsum(j) = R_E(j) * RNDSUM(j)

C     ***********************************
C     *  SUM TARGETS OF TYPE K KILLED  *
C     ***********************************

      IF (RPE(I,J,K) .EQ. 0) GO TO 35

c     XKILLED=NBRNDS/(RPE(I,J,K)*(STK(I,J,K,L) +             !!  here,  xkilled = X1,  see above
c    1         ZK*STDEV(I,J,K,L)))

      TOTKILL(K)    =  TOTKILL(K) + XKILLED
      TOTLKILL(K,L) =  TOTLKILL(K,L) + XKILLED
      SUMKILL(J,K)  =  SUMKILL(J,K) + XKILLED                !!  sum up type K targets killed by munition J
      TIK(I,K)      =  TIK(I,K) + XKILLED                    !!

C     ****************************************
C     *  SUM OF PLATFORMS OF TYPE I ATTRITED  *
C     ****************************************

c     XATTRIT=NBRNDS * ATT(I,J,K)/RPE(I,J,K)
      xattrit  =  ATT(i,j,k) * STK(i,j,k,l) * xkilled        !!
      TOTATT(I) = TOTATT(I) + XATTRIT

      GO TO 35

  100 READ(15,1100,END=105) LINE
 1100 FORMAT(A)
```

```
      IF (LINE(1:5) .NE. '  ROW') GO TO 100

      DO 103 I=1,NBROWS
        READ(15,*) R,VAL,DP(I),DN(I)
c       write(3,*) R,VAL,DP(I),DN(I)                              !!   row, RHS, pos_dev, neg_dev
  103 CONTINUE

  105 continue


c   -------------- !   -----------------


      IF (ITER500 .EQ. 1) THEN
        WRITE(3,*) ' '
        WRITE(3,*) ' ****************************************'
        WRITE(3,*) ' '
        WRITE(3,*) ' MESSAGE FROM SUBROUTINE   SIMPLEX :'
        WRITE(3,*) ' '
        WRITE(3,*) '   THERE HAVE BEEN MORE THAN 2500 ITERATIONS'
        WRITE(3,*) '   THE LGP SOLUTION IS QUESTIONABLE'
        WRITE(3,*) ' '
        WRITE(3,*) ' ****************************************'
      ENDIF


      L = 1                                                       !!

      CG = COSTGOAL + ( DP(1) - DN(1) )
      do  K=1,NBK
        K1 = 1 + K
        KG(K) = FTBK(K,L) * TQ(K) + ( DP(K1) - DN(K1) )
      enddo
      do  I=1,NBI
        I1 = 1 + NBK + I
        AG(I) = FATTR(I) * PQ(I) + ( DP(I1) - DN(I1) )
      enddo


      tot_mun_req_cost = 0.0
      do   J=1,NBJ
c       munition_req_cost(J) = 1.E3 * cm(J) * reqsum(J)           !!   in millions
        munition_req_cost(J) = cm(J) * reqsum(J)                  !!   in billions
        tot_mun_req_cost     = tot_mun_req_cost + munition_req_cost(J)
      enddo

      tot_plat_att_cost = 0.0
      do  I=1,NBI
c       platform_att_cost(I) = 1.E3 * platcost(I) * totatt(I)     !!   in millions
        platform_att_cost(I) = platcost(I) * totatt(I)            !!   in billions
        tot_plat_att_cost    = tot_plat_att_cost + platform_att_cost(I)
      enddo



c     ---   Write JMAP Output Summary Information

      write(3,700)
      write(3,701)  prepri(1), costgoal, tot_mun_req_cost
c    .             ,tot_mun_req_cost+tot_plat_att_cost          !!  platform attrition cost not in cost-goal
      write(8,730)  pass, tot_mun_req_cost, (totkill(K)/tq(K),k=1,NBK)  !!  write cost solution for each pass


      write(3,702)  prepri(2)
      L = 1                                                       !!
      do  K=1,NBK
        write(3,703)  targ(K), ftbk(K,L), totkill(K)/tq(K),
     .                1.E6 * totcost(K)/totkill(K)
      enddo
```

```
      write(3,704)  prepri(3)
      do  I=1,NBI
        write(3,703)  plat(I), fattr(I), totatt(I)/pq(I)
c     .               ,1.E3 * platform_att_cost(I)
      enddo
c     write(3,705)    1.E3 * tot_plat_att_cost

  700 format(/,50x,'JMAP  SUMMARY',///
     .         ,36x,'Priority       Goal     Solution',//)
  701 format(/,'Overall Cost   (Billions)',15x,i1,3x,3(5x,f7.4))
  702 format(/'Fraction of Targets Killed',14x,i1,42x,
     .       'Munition Cost per Kill',/,88x,'(Thousands $)')
c 703 format(10x,a15,19x,2(5x,f7.4),21x,f10.4)
  703 format(10x,a15,19x,2(5x,f7.4),18x,f13.4)
c 704 format(/,'Fraction of Platforms Attrited',10x,i1,47x,
c    .         'Attrition Cost',/,89x,'(Millions $)')
  704 format(/,'Fraction of Platforms Attrited',10x,i1,/)
  705 format(/,58x,'Total Platform Attrition Cost  ',f10.4)
  730 format(/,4x,i2,6x,f10.6,12x,4f8.4)



c     ---   Write Munition Expenditures, Requirements, and Cost

      write(3,710)
      do J=1,NBJ
        write(3,711)  mun(J), rndsum(J), reqsum(J), stock(J)-reqsum(J),
     .                munition_req_cost(J)
      enddo
      write(3,712)  tot_mun_req_cost
      write(3,705)  tot_plat_att_cost

  710 format(/////,20x,'Munition Expenditures, Requirements,',
     .             ' Left-in-Stock and Cost',//,
     . 40x,'Expenditures   Requirements   Left-in-Stock',11x,'Cost',/,
     . 89x,'(Billions $)')
  711 format(a15, 20x, 3(8x,f7.1), 12x, f7.4)
  712 format(/,68x,'Total Munition Cost',5x,f7.4)



c     ---   Write Munition Expenditures and Requirements by Platform

      tolerance = 0.00001
      write(3,720)
      do J=1,NBJ
        do I=1,NBJ
          if (require(I,J) .gt. tolerance) then
            write(3,721)  mun(J), plat(I), expend(I,J), require(I,J)
          endif
        enddo
      enddo

  720 format(///,15x,'Munition Expenditures and Requirements by'
     .           ,' Platform',//,40x,'Expenditures   Requirements',/)
  721 format(a15,5x,a15,2(8x,f7.1))



c     ---   Write JMAP+ target kills by Platform


      if (pass .lt. 10) then
        write(ch_pass1,321)   pass
        filename = 'j_t_out.'//ch_pass1
      else
        write(ch_pass2,322)   pass
        filename = 'j_t_out.'//ch_pass2
      endif
```

```
      open(4,file=filename)

      write(3,804)
      write(4,805)   filename
      write(3,806)  (plat(I), I=1,NBI)
      write(4,806)  (plat(I), I=1,NBI)
      do  K=1,NBK
        write(3,807)  targ(K), (TIK(I,K),I=1,NBI), totkill(K), tq(K),
     .                totkill(K)/tq(K)
        write(4,807)  targ(K), (TIK(I,K),I=1,NBI), totkill(K), tq(K),
     .                totkill(K)/tq(K)
      enddo

      WRITE(3,1000) ITER                                              !!
 1000 FORMAT(//,I6,'    ITERATIONS')

c     ---   Write JMAP+ Target Kills by Platform and Munition

      write(4,808)
      write(4,809)  (plat(I), I=1,NBI)
      do  J=1,NBJ
        write(4,810)  mun(J)
        do  K=1,NBK
          write(4,807)  targ(K), (TIJK(I,J,K),I=1,NBI), sumkill(J,K)
        enddo
      enddo

  321 format(i1)
  322 format(i2)
  804 format(///,40x,'JMAP+  Targets Killed by Platform')
  805 format(//,'File Name :  ',a14,
     .       /,40x,'JMAP+  Targets Killed by Platform')
  806 format(/,100x,'Total',6x,'Total',4x,'Fraction',/,
     .       10x,12(1x,a6),5x,'Killed',4x,'Available',3x,'Killed',/)
  807 format(3x,a5,12(1x,f6.2),2x,2(5x,f6.2),5x,f6.4)
  808 format(///,32x,'JMAP+  Targets Killed by Platform and Munition')
  809 format(/,10x,12(1x,a6),6x,'Total')
  810 format(/,a15)

      close(4)

      RETURN
      END
c             ! PRTSOLU




      INTEGER FUNCTION TEX(I,J,K,L)

c     ************************************************************
c     *  THIS FUNCTION RETURNS THE COLUMN INDEX OF X(I,J,K,L)  *
c     ************************************************************

      INCLUDE 'pair_2.inc'


      IF (MCOL(I,J,K) .EQ. 0) THEN
        TEX=0
      ELSE
        TEX=(L-1)*NIJK + MCOL(I,J,K)
      ENDIF


      RETURN
      END
c              ! TEX(I,J,K,L)
```

```
      SUBROUTINE   XET(M,I,J,K,L)

C     ******************************************************
C     *  GIVEN THE COLUMN INDEX OF X(I,J,K,L), THIS ROUTINE  *
C     *  RETURNS I,J,K,L                                     *
C     ******************************************************

      INCLUDE 'pair_2.inc'

      MM=M
      L=1

   10 IF (MM .GT. NIJK) THEN
        L=L+1
        IF (L .GT. 3) THEN
          PRINT*,' '
          PRINT*,' ******************************'
          PRINT*,' '
          PRINT*,' MESSAGE FROM SUBROUTINE   XET :'
          PRINT*,' '
          PRINT*,'    NON-EXISTENT COLUMN INDEX'
          PRINT*,' '
          PRINT*,' ******************************'
          PRINT*,' '
          PRINT*,' '
          STOP
        ELSE
          MM=MM - NIJK
          GO TO 10
        ENDIF
      ENDIF

      DO 20 I=1,NBI
        DO 30 J=1,NBJ
          DO 40 K=1,NBK
            IF (MM .EQ. MCOL(I,J,K)) THEN
              RETURN
            ENDIF
   40     CONTINUE
   30   CONTINUE
   20 CONTINUE

      PRINT*,' '
      PRINT*,' ******************************'
      PRINT*,' '
      PRINT*,' MESSAGE FROM SUBROUTINE   XET :'
      PRINT*,' '
      PRINT*,'    NON-EXISTENT COLUMN INDEX'
      PRINT*,' '
      PRINT*,' ******************************'
      PRINT*,' '
      PRINT*,' '
      STOP

      return
      END
c             ! XET(M,I,J,K,L)
```

```
subroutine   close_files

close(9)
close(10)
close(20)
close(30)
close(40)

close(15)
close(3)

return
end   ! close_files
```

JMAP/Threat Pair

Code File : tm_input.for

```
c     File Name :    tm_input.f

*********************************************************************
*                The Monte Carlo THREAT MODEL                       *
*                      (Input Module)                               *
*                                                                   *
*          ********************************************           *
*           g l o s s a r y   o f   v a r i a b l e s               *
*          ********************************************           *
*                                                                   *
*     common/theat/theatnames(6)                                    *
*       character*45 theatnames                                     *
*                                                                   *
*     common/misc/ncase,ncases,lun,beta,a,exwt,title                *
*       integer ncase, ncases, lun                                  *
*       real beta, a, exwt                                          *
*       character*45 title                                          *
*                                                                   *
*      ncase     :current case number indicating scenario, data,    *
*                 etc.                                              *
*      ncases    :number of cases to be processed for this run      *
*                 (maximum of 10)                                   *
*      lun       :logical unit number designating output device     *
*      beta      :polya parameter                                   *
*      a         :ratio of expenditures to shooters for current     *
*                 case                                              *
*      exwt      :mean (expected value) of distr. being processed   *
*      title     :weapon/shooter name for entire run                *
*                                                                   *
*   common/plat/nplat,iaptot,replen,ia,rp                           *
*       integer   nplat                                             *
*       real      iaptot, replen, rp, ia                            *
*                                                                   *
*      nplat     :number of shooters for current case               *
*      iaptot    :total initial loadout of shooters                 *
*      replen    :average shooters refill size                      *
*      ia        :average shooter initial loadout                   *
*      rp        :average shooter reorder point                     *
*                                                                   *
*     common/plt/ npt,np(40,10),mag(40),iap(40),nrp(40),rop(40),    *
*                 platnames(40)                                     *
*       integer  npt, np                                            *
*       real     mag, iap, nrp, rop                                 *
*       character*20 platnames                                      *
*                                                                   *
*      npt       :number of shooter types for this run (max of 40)  *
*      np        :number of shooters keyed by shooter type, case    *
*      iap       :initial loadouts for each shooter type            *
*      nrp       :refill size for each shooter type                 *
*      rop       :reorder point for each shooter type               *
*      platnames:name of shooter by type                            *
*                                                                   *
*     common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200) *
*       integer  ntt, rt                                            *
*       real     rnt, r                                             *
*       real*8   ext                                                *
*       character*20 targnames                                      *
*                                                                   *
*      ntt       :number of target types for this run (max of 200)  *
*      rnt       :number of targets keyed by target type, case      *
*      ext       :average expenditures needed to kill one target by *
*                 target type                                       *
*      rt        :number of targets for current case                *
*      r         :number of targets for each type for current case  *
*      targnames:name of targets by type                            *
*                                                                   *
*     common/percents/perckill(6),nperc                             *
*       integer nperc                                               *
```

```
*       real perckill                                                  *
*       perckill  :In evaluating the fraction of targets killed        *
*                  at different levels of confidence, perckill are     *
*                  the chosen confidences (actually in fractional      *
*                  form, not %)                                        *
*       nperc     :The number of confidences chosen                    *
*********************************************************************
```

```
      subroutine   get_inputs(iflag,file_unit,m_type)

c     ********************************************************************
c     * Get_inputs gets inputs for  m_type munition  Threat Model run.  *
c     * Referenced by:                          subroutine threat_model  *
c     ********************************************************************

      include 'pair_1.inc'

      integer       iflag, file_unit, m_type

      common/theat/theatnames(6)
        character*45 theatnames
      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
      common/plt/ npt,np(40,10),mag(40),iap(40),nrp(40),rop(40),
     &            platnames(40)
        integer  npt, np
        real     mag, iap, nrp, rop
        character*20 platnames
      common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200)
        integer  ntt, rt
        real     rnt, r
        real*8   ext
        character*20 targnames

      logical       having_target
      integer       i, j, k, L
      real          tolerance
      data          tolerance /0.00001/

      j      = m_type
      beta   = 1.0
      L      = 1                                                  !  single time interval
      ncases = 1                                                  !  single case

      npt  = 0
      ntt  = 0
      do  i=1,NBI
        k = 0
        having_target = .FALSE.
        do  k=1,NBK
          if (TIJK(i,j,k) .gt. tolerance) then
            having_target   = .TRUE.
            ntt             = ntt + 1
            ext(ntt)        = RPE(i,j,k) * STK(i,j,k,L)           ! rounds-to-kill
            rnt(ntt,ncases) = TIJK(i,j,k)                         ! targets
          endif
        enddo

        if (having_target) then
          npt = npt + 1
          iap(npt)       = IA(i,j)
          nrp(npt)       = RS(i,j)
          rop(npt)       = RP(i,j)
          mag(npt)       = iap(npt)
          np(npt,ncases) = pq(i)
        endif
      enddo

      call echo_threat_model_inputs(file_unit,m_type)

      iflag = -1

      return
      end
*     (*   get_inputs   *)
```

```
      subroutine   echo_threat_model_inputs(file_unit,m_type)

c     ************************************************************************
c     * This subroutine echos the inputs which will be used in the           *
c     * current case.                                                        *
c     * Referenced by:                              subroutine threat_model  *
c     ************************************************************************

      include 'pair_1.inc'

      integer       file_unit, m_type

      common/theat/theatnames(6)
        character*45 theatnames
      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
      common/plt/ npt,np(40,10),mag(40),iap(40),nrp(40),rop(40),
     &            platnames(40)
        integer  npt, np
        real     mag, iap, nrp, rop
        character*20 platnames
      common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200)
        integer  ntt, rt
        real     rnt, r
        real*8   ext
        character*20 targnames

      integer       n


      ncases = 1

      write(file_unit,100)    mun(m_type)
      write(file_unit,101)    npt
      write(file_unit,102)    ntt
      write(file_unit,103)
      do  n=1,ntt
        write(file_unit,104)  n, ext(n), rnt(n,ncases)
      enddo
      write(file_unit,105)
      do  n=1,npt
        write(file_unit,106)   n, np(n,ncases)
      enddo
      write(file_unit,107)
      do  n=1,npt
        write(file_unit,104)   n, iap(n), nrp(n), rop(n)
      enddo

      return
  100 format(/,a)
  101 format(3x,'Number of platform types  : ',i2)
  102 format(3x,'Number of platform/target type combinations  : ',i2)
  103 format(3x,'Rounds-to-Kill and number of targets  :')
  104 format(5x,i2,3x,3(3x,f7.3))
  105 format(3x,'Number of platforms  :')
  106 format(5x,i2,16x,i7)
  107 format(3x,'Initial-loadout, refill-size, and reorder-point  :')
      end

c     (*   echo_threat_model_inputs   *)
```

```
      subroutine   assign

c     ************************************************************************
c     * Initializes those quantities which depend on case.                   *
c     * Referenced by:                              subroutine threat_model *
c     ************************************************************************
      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia
      common/plt/ npt,np(40,10),mag(40),iap(40),nrp(40),rop(40),
     &            platnames(40)
        integer  npt, np
        real     mag, iap, nrp, rop
        character*20 platnames
      common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200)
        integer  ntt, rt
        real     rnt, r
        real*8   ext
        character*20 targnames
        integer  j

      real temp

      temp=0.0
      do 10 j=1,ntt
        temp=temp+rnt(j,ncase)
 10   continue
      if (temp.gt.0.0) then
         rt=max0(nint(temp),1)
      else
         rt=0
      endif

      replen = 0.
      iaptot = 0.
      nplat  = 0
      rp     = 0.
      do 20 j=1,npt
        iaptot = iaptot + iap(j)*np(j,ncase)
        replen = replen + nrp(j)*np(j,ncase)
        rp     = rp     + rop(j)*np(j,ncase)
        nplat  = nplat + np(j,ncase)
 20   continue

      if(nplat.gt.0) then
        replen = replen/float(nplat)
        rp     = rp/float(nplat)
        ia     = iaptot/nplat
        do 30  j=1,ntt
          r(j)=rnt(j,ncase)
 30     continue
        call amean(exwt)
        a = exwt/float(nplat)
      end if
      return
      end
c     (*   assign   *)
```

```
      subroutine   amean(ex)

c     ************************************************************************
c     * Amean calculates the expected expenditure necessary to kill all   *
c     * targets.                                                          *
c     * Referenced by:                                    subroutine assign *
c     ************************************************************************
      real  ex

      common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200)
        integer  ntt, rt
        real     rnt, r
        real*8   ext
        character*20 targnames
        integer  j

      ex = 0
      do 10 j=1,ntt
        ex = ex + r(j)*ext(j)
 10   continue
      if (ex.gt.0.0) ex=amax1(ex,1.0)
      return
      end
c     (*   amean   *)




      subroutine   display

c     ************************************************************************
c     * Display pauses to allow output to be viewed at the console.       *
c     * Referenced by:                                 subroutine threat_model *
c     ************************************************************************
      character*1 idummy

      write(*,10)
      read(*,20) idummy
      return
 10   format(/,' < press return to continue >' )
 20   format(a1)
      end
c    (*   display   *)
```

```
      subroutine   echo

c     ************************************************************************
c     * This subroutine echos the inputs which will be used in the           *
c     * current case.                                                        *
c     * Referenced by:                                 subroutine threat_model *
c     ************************************************************************
      common/theat/theatnames(6)
        character*45 theatnames
      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia
      common/plt/ npt,np(40,10),mag(40),iap(40),nrp(40),rop(40),
     &            platnames(40)
        integer  npt, np
        real     mag, iap, nrp, rop
        character*20 platnames
      common/tar/ ntt,rnt(200,10),ext(200),rt,r(200),targnames(200)
        integer  ntt, rt
        real     rnt, r
        real*8   ext
        character*20 targnames
      real       ex
      integer    j

      write(lun,5)
      write(lun,10) ncase, title, theatnames(ncase), rt,
     &              nplat, beta
      if (nplat.ge.1) then
        write(lun,20)
        write(lun,30) (targnames(j),r(j),ext(j),j=1,ntt)
        if (rt.ge.1)  then
          ex = exwt/rt
          write(lun,40) ex,exwt
          write(lun,50)
          write(lun,60) (platnames(j),np(j,ncase),mag(j),iap(j),
     &                  np(j,ncase)*iap(j),nrp(j),rop(j),j = 1,npt)
          write(lun,70) replen,rp,ia,iaptot
        end if
      end if

      return
    5 format(//////,' Monte Carlo Threat Model',/)
   10 format(' Case Number ',i2,//,
     &       ' Weapon  : ',a45,/
     &       ' Theater : ',a45,//
     &       ' Number of Targets     =',i4,/,
     &       ' Number of Shooters    =',i4,/,
     &       ' Polya Parameter       = ',f6.2)
   20 format(//' Target Data:'/
     & ' Name                  Number  Expenditure/Kill')
   30 format(1x,a20,f8.2,f13.3)
   40 format(/' Average Expenditure Per Kill = ',f8.2,/
     &       ' Average  Total  Expenditures = ',f8.2)
   50 format(//,' Shooter Data:                    ',
     & '          Initial  Initial',/,
     & ' Name                  Number   ',
     & '  MagSz   Loadout  TotLoad    RefSz   RordPt')
   60 format(1x,a20,i7,f11.2,f10.2,3f9.2)
   70 format(/' Average Shooter Replenishment Size = ',f10.3,/,
     &       ' Average Shooter Reorder Point       = ',f10.3,/,
     &       ' Average Shooter Initial Loadout     = ',f10.3,/,
     &       ' Total Shooter Initial Loadouts      = ',f10.3)
      end
c     (*   echo   *)
```

```
      logical FUNCTION ans_yes(i)

c     ************************************************************************
c     * Referenced by:           subroutines   chk_scaling and get_iter  *
c     ************************************************************************

      integer    i
      character*1 answer

      ans_yes = .true.
      if (i.eq.1) then
        write(*,10)
        read(*,20) answer
        if (answer.eq.'N' .or. answer.eq.'n') ans_yes = .false.
      else
        write(*,30)
        read(*,20) answer
        if (answer.ne.'Y' .and. answer.ne.'y') ans_yes = .false.
      endif
 10   format('& (<CR>=yes,n=no) --> ')
 20   format(a1)
 30   format('& (y=yes,<CR>=no) --> ')
      return
      end
c     (*   ans_yes   *)
```

JMAP/Threat Pair

Code File : tm_monte.for

```
c     File Name :    tm_monte.for


*********************************************************************
*                  Monte Carlo THREAT MODEL                         *
*                   (Calculational Module)                          *
*                                                                   *
*          ***********************************************          *
*           g l o s s a r y   o f   v a r i a b l e s               *
*          ***********************************************          *
*                                                                   *
*  common/ddown/prdd(0:2301),lldd,ludd,ddmean                       *
*    integer lldd, ludd                                             *
*    real prdd, ddmean                                              *
*                                                                   *
*      prdd(i)  :probability that i-lld refill demands are made on  *
*                the depot for the current case                     *
*      lldd     :lower bound on the prdd distribution               *
*      ludd     :upper bound on the prdd distribution               *
*      ddmean   :mean of the prdd distribution                      *
*                                                                   *
*  common/supply/prsup(0:2301,30),llsu(30),lusu(30),smean(30)       *
*    integer llsu, lusu                                             *
*    real prsup, smean                                              *
*                                                                   *
*     prsup(i,j):probability that i-llsup(j) expenditures are       *
*                supplied at the j-th inventory for the current case*
*      llsup(j) :lower bound on the prsup(*,j) distribution         *
*      lusup(j) :upper bound on the prsup(*,j) distribution         *
*      smean(j) :mean of the prsup(*,j) distribution                *
*                                                                   *
*  common/runout/rmean(30)                                          *
*    real rmean                                                     *
*                                                                   *
*      rmean(j) :average fraction of shooters that run out at the   *
*                j-th inventory                                     *
*                                                                   *
*  common/target/targ(0:601,30),llt(30), lut(30), tmean(30)         *
*    integer llt, lut                                               *
*    real targ, tmean                                               *
*                                                                   *
*      targ(i,j):probability that i-llt(j) targets are killed at    *
*                the j-th inventory for the current case            *
*      llt(j)   :lower bound on the targ(*,j) distribution          *
*      lut(j)   :upper bound on the targ(*,j) distribution          *
*      tmean(j) :mean of the targ(*,j) distribution                 *
*                                                                   *
*  common/inventy/ia_inv(30),rdep(30),inv(30),n_inv                 *
*    integer n_inv                                                  *
*    real ia_inv, rdep, inv                                         *
*                                                                   *
*       ia_inv(i):initial allowance at the i-th inventory for the   *
*                 current case                                      *
*       rdep(i)  :refills (of size replen) in the depot at the i-th *
*                 inventory for the current case                    *
*       inv(i)   :i-th inventory for current case                   *
*                                                                   *
*  common/dimens/maxex,maxtar,maxplat                               *
*    integer maxex, maxtar, maxplat                                 *
*       maxex    :dimension of the first index of the prsup array.  *
*                 Should be greater or equal to the demand, iwt.    *
*       maxtar   :dimension of the first index of the targ array.   *
*                 Should be greater or equal to the no. of targets. *
*       maxplat  :dimension of the nu array.                        *
*                 Should be greater or equal to the no. of platforms*
*                                                                   *
*     common/percents/perckill(6),nperc                             *
*       integer nperc                                               *
```

```
*       real perckill                                                  *
*       perckill  :In evaluating the fraction of targets killed        *
*                  at different levels of confidence, perckill are     *
*                  the chosen confidences (actually in fractional      *
*                  form, not %)                                        *
*       nperc     :The number of confidences chosen                    *
*********************************************************************


      BLOCK DATA initial

*********************************************************************
* Store dimensions of arrays for expenditures, targets, and         *
* platforms.                                                        *
*                                                                   *
*  The following arrays must have dismension maxex as shown:        *
*  common/supply/       : prsup(0:maxex,30)                         *
*  common/ddown/        : prdd(0:maxex)                             *
*  subroutine calc_mean : x(0:maxex)                                *
*  subroutine cdf       : pr(0:maxex)                               *
*  subroutine cumsum    : a(0:maxex), b(0:maxex)                    *
*  subroutine dist      : a(0:maxex)                                *
*  subroutine getbd     : a(0:maxex)                                *
*  subroutine print_dist: a(0:maxex)                                *
*  subroutine zero      : pr(0:maxex)                               *
*  subroutine zerod     : dpr(0:maxex)                              *
*                                                                   *
*  The following arrays must have dismension maxtar as shown:       *
*  common/target/       : targ(0:maxtar,30)                         *
*  function   fval      : t(0:maxtar)                               *
*  subroutine monte     : tartemp2(0:maxtar), tartemp1(0:maxtar)    *
*                         n_dd(maxex)                               *
*  subroutine targ80    : targ(0:maxtar)                            *
*  subroutine target    : pr(0:maxtar)                              *
*                                                                   *
*  The following arrays must have dismension maxplat as shown:      *
*  subroutine analyze   : nu(maxplat),nz(2,maxplat)                 *
*  subroutine box       : nu(maxplat)                               *
*  subroutine monte     : nu(maxplat),nz(2,maxplat)                 *
*  subroutine refills   : nu(maxplat),nz(2,maxplat),n_dd(maxex)     *
*  subroutine ref_setup : nu(maxplat),nz(2,maxplat),n_dd(maxex)     *
*********************************************************************
      common/dimens/maxex,maxtar,maxplat
        integer maxex, maxtar, maxplat

        data maxex/2301/
        data maxtar/601/
        data maxplat/2001/
      end
*     (*   block data initial   *)
```

```
      SUBROUTINE main_up(ntar,iflag)

*********************************************************************
* Referenced by:                 subroutine model (file: input.f) *
* For the current case (ncase), determine the inventories for      *
* which evaluations will take place and then for each of these     *
* inventories compute the distributions of supply, targets killed, *
* and shooters running out.                                        *
*********************************************************************
      integer ntar, iflag

      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia

      integer iwt, ntar_hold, nplat_hold, iter
      real    ran, randm, scale

c     write(*,*) 'Working on case ',ncase
      ran = randm(0)                          !initialize random number generator
      ntar_hold = ntar
      nplat_hold = nplat
      call chk_scaling(exwt,ntar,nplat,lun,ncase,scale)
      iwt = int(exwt+0.5)                     !round expenditures to integer
      call inventory(iwt)                         !determine inventories
      call get_iter(iter,iwt,iflag)               !get number if iterations
      if (iter.gt.0) then
        call initialize(iwt,ntar,nplat,replen)    !initialize arrays
        call monte(iwt,ntar,iter)                 !perform monte carlo analysis
        ran = randm(2)
        call summary(title,nplat_hold,ntar_hold,lun,iter,ncase) !print distributions
        call moe(nplat,ntar,lun,scale,title)      !evaluate requirements for
      endif                                       !           different MOEs.
      return
      end
* (* subroutine main_up *)
```

```
      SUBROUTINE analyze(i,nballs,ref,nurns,nu,nz)

*********************************************************************
* Referenced by:                                  subroutine monte *
* Extract from the current iteration the amount supplied to the     *
* targets (sup) and the number of shooters that runout (cout) at    *
* the i-th inventory. Accumulate the results in the appropriate     *
* distributions.                                                    *
*********************************************************************
      integer i,nu(2001),nz(2,2001),nurns,nballs
      real ref

      common/supply/prsup(0:2301,30),llsu(30), lusu(30), smean(30)
        integer llsu, lusu
        real prsup, smean
      common/runout/rmean(30)
        real rmean
      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer n_inv
        real ia_inv, rdep, inv

      real    sup, frac, avail, cout
      integer iu, nsup

      sup    = 0.0
      cout   = 0.0
      do 50 iu = 1,nurns
        avail = ia_inv(i)+nz(1,iu)*ref      !available supply
        if  (avail .gt. nu(iu)) then
          sup = sup + nu(iu)                !  Shooter does not run out
          if (nz(2,iu).gt.0)
     &       cout = cout + amax1(0.,1.-(avail-nu(iu)))
        else
          sup = sup + avail                 !  Shooter runs out
          cout = cout + 1.
        endif
   50 continue
      if (sup .ge. nballs) then
        prsup(nballs,i) = prsup(nballs,i) + 1
        lusu(i) = nballs
      else
        nsup = int(sup)
        frac = sup - nsup
        prsup(nsup,i) = prsup(nsup,i) + (1 - frac)
        prsup(nsup+1,i) = prsup(nsup+1,i) + frac
        if (sup.gt.lusu(i)) lusu(i) = nsup + 1
      endif
      rmean(i) = rmean(i) + amin1(float(nurns),cout)
      return
      end
* (* subroutine analyze *)
```

```
      SUBROUTINE box(nballs,nurns,nu)

*****************************************************************
* Referenced by:                                  subroutine monte *
* Distribute demands(nballs) over platforms (nurns) in a polya     *
* fashion.                                                         *
*  nu(i)   = number of demands made on the i-th platform.          *
*****************************************************************
      integer nballs, nurns, nu(2001)

      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title
*                                Note: n_d has dimension = exp+plat
      integer   n_d(4500), index, j, k
      real      nbeta, ran, randm

      do 10 j = 1,nurns
        n_d(j)   = j
        nu(j)    = 0
   10 continue
      nbeta    = nurns*beta
      do 20 k = 1,nballs
        ran = randm(1)*(nbeta + k - 1)
        if (ran.le.nbeta) then
          index = int(ran/beta + 1)
        else
          index = int(ran-nbeta+nurns+1)
        endif
        n_d(nurns+k) = n_d(index)
        nu(n_d(index)) = nu(n_d(index)) + 1
   20 continue
      return
      end
* (* subroutine box *)
```

```
      SUBROUTINE calc_mean(x,ex,var,llx,lux,case)

*********************************************************************
* Referenced by:                              subroutine print_dist *
* Compute mean and variance of the cumulative distribution stored   *
* in array x.                                                       *
*********************************************************************
      integer llx, lux, case
      real    x(0:2301,30), ex, var

      real      ex2
      integer   i

      ex  = 0.0
      ex2 = 0.0
      do 10 i=0,lux-llx
        ex  = ex + 1- x(i,case)
        ex2 = ex2 + (1-x(i,case))*(i+llx)
 10   continue
      ex2 = ex2 + llx*(llx-1)*.5
      ex  = ex  + llx
      var = 2*ex2 + ex - ex*ex
      if (sign(1.,var).lt.0 .and. abs(var).gt.0.1) write(*,*)          !#2
c     &              'xxmean: variance negative - var = ',var          !
*     std = sign(1.,var)*sqrt(abs(var))
      return
      end
* (* subroutine calc_mean *)




      SUBROUTINE cdf(pr,llpr,lupr,case)

*********************************************************************
* Referenced by:                              subroutine print_dist *
* Cdfr converts a pdf to a cdf.                                     *
*********************************************************************

      integer     llpr, lupr, case
      real        pr(0:2301,30)
      integer     j

      do 10 j=1,lupr-llpr
        pr(j,case)=pr(j,case)+pr(j-1,case)
 10   continue
      return
      end
* (* subroutine cdf *)
```

```
      SUBROUTINE cumsum(b,llb,lub,a,lla,lua,prob,case)

*********************************************************************
* Referenced by:                                    subroutine monte *
* Accumulate the distribution a into distribution b weighted by    *
* the probability prob.                                            *
*********************************************************************
      integer lla, lua, llb, lub, case
      real    prob
      real*8  a(0:2301,30), b(0:2301,30)
      integer j

      if (llb.gt.lla .or. lub.lt.lua) then
        write(*,20) lla,lua,llb,lub
 20     format(' Error in cumsum!',/,
     &         '    lla = ',i5,'    lua = ',i5,/
     &         '    llb = ',i5,'    lub = ',i5)
        stop
      else
        do 10 j = lla,lua
          b(j-llb,case) = b(j-llb,case) + a(j-lla,case)*prob
 10     continue
      endif
      return
      end
* (* subroutine cumsum *)
```

```
      SUBROUTINE chk_scaling(exp,ntar,nplat,lun,ncase,scale)

*********************************************************************
* Referenced by:                                  subroutine main_up *
* Make sure that the number of expenditures, targets, and platforms *
* do not exceed the dimensions of the appropriate arrays. If they   *
* do exceed the arrays, compute how much the problem must be scaled *
* down so that it fits the existing arrays.                         *
*********************************************************************
      integer ntar, nplat, lun, ncase
      real exp, scale

      common/dimens/maxex,maxtar,maxplat
        integer maxex, maxtar, maxplat

      integer  flag, iwt
      real     scalemax
      logical  ans_yes

      save flag

      if (ncase.eq.1) then
*4      write(*,5)
*       write(*,6)
*       write(*,7)
*       read(*,*,err=4) flag
*       if (flag.lt.1 .or. flag.gt.3) goto 4
        flag=3
      endif

      scale = amin1(1.,maxex/exp,float(maxtar)/ntar,
     &              float(maxplat)/nplat)

      if (scale.lt.1.) then
        iwt = int(exp + .5)
*       write(*,10) ncase,iwt,maxex,nplat,maxplat,ntar,maxtar,scale
*       write(lun,10) ncase,iwt,maxex,nplat,maxplat,ntar,maxtar,scale
*       write(*,25) int(exp*scale+.5),int(nplat*scale+.5),
*    &             int(ntar*scale+.5), scale
      else
        if (flag.eq.1) write(*,30) ncase
      endif
      if (flag.eq.1 .or. (flag.eq.2 .and. scale.lt.1)) then
        scalemax = scale
        write(*,20)
        if (.not.ans_yes(1)) then
 50       write(*,40)
          read(5,*,err=50) scale
          if (scale.gt.scalemax) then
            write(*,55) scalemax
            goto 50
          endif
          write(*,25) int(exp*scale+.5),int(nplat*scale+.5),
     &              int(ntar*scale+.5),scale
          write(*,20)
          if (.not.ans_yes(1)) goto 50
        endif
      endif

      if (scale.lt. 1.0) then
        exp = exp*scale
        nplat = int(nplat*scale+.5)
        ntar  = int(ntar*scale +.5)
*       write(lun,25) int(exp+.5), nplat, ntar, scale
      endif
      return

  5   format(///,15x,'********* Scaling **********',//
     &  5x,'Array sizes in the code may be too small for the given ',/
```

```
     &  5x,'inputs.  If this is the case, the model will scale down ',/
     &  5x,'the inputs to fit in the existing arrays and then scale ',/
     &  5x,'the output back up to correspond to the original inputs.',/
     &  5x,'The user is also given the option to scale down inputs',/
     &  5x,'even when array sizes are adequate. Note, however, this',/
     &  5x,'process only gives reliable results when the inputs are',/
     &  5x,'"large".'//
     &  5x,'Choose one of the following options:',//)
  6   format(
     &  5x,' 1. Prompt user for a scaling factor at the beginning of ',/
     &  5x,'     every case.',/
     &  5x,' 2. Prompt user for a scaling factor at the beginning of',/
     &  5x,'     only those cases for which the array sizes are not',/
     &  5x,'     adequate.',/
     &  5x,' 3. Let the model automatically decide whether scaling',/
     &  5x,'     is needed.')
  7   format(/' Enter choice. (1, 2, or 3) --> ')
 10   format(/' Dimension of array is too small in case ',i2,' :',/,
     &        '   Expenditures = ',i8,'    Max allowed=',i8,/,
     &        '   Platforms    = ',i8,'    Max allowed=',i8,/,
     &        '   Targets      = ',i8,'    Max allowed=',i8,//,
     &        ' A scaling factor of at least ',f6.3,
     &           ' is required to run the case.',/)
 25   format( '   The Model will be run with the following inputs:',/,
     &        '         Expenditures = ',i8,/
     &        '         Platforms    = ',i8,/
     &        '         Targets      = ',i8,/
     &        '         Scale Factor = ',f6.3,/)
 20   format('  Is this oK ? ')
 30   format(/' Array sizes in case ',i3,
     &        ' are adequate, scaling will not be done. ')
 55   format(//'    Factor must be less than ',f6.3,' ! Try again.'//)
 40   format(' Enter the amount by which inputs will be scaled down. ')
      end
* (* chk_scaling *)
```

```
      SUBROUTINE dist(a,lla,lua,npoint,case)

*********************************************************************
* Referenced by:                                  subroutine monte *
* Convert from number of events to distribution of events, that    *
* is, divide all points in array from a(0) to a(lua-lla) by npoint.*
*********************************************************************
      integer   lla, lua, npoint, case
      real      a(0:2301,30)
      integer   j

      do 10 j=lla,lua
        a(j-lla,case) = a(j-lla,case)/npoint
  10  continue
      return
      end
* (* subroutine dist *)




      real FUNCTION fval(t,llt,lut,k,case)

*********************************************************************
* Referenced by:                                    subroutine moe    *
*                                                   subroutine targ80 *
* Obtain the cumulative probability at k stored in array t. The    *
* cum is zero if k is below the lower bound and it is one if it is *
* above the upper bound.                                           *
*********************************************************************
      integer llt,lut,k,case
      real t(0:2301,30)
      if (k.lt.llt) then
        fval = 0.
      else if (k.gt.lut) then
        fval = 1.
      else
        fval = t(k-llt,case)
      endif
      return
      end
* (* function fval *)
```

```
      SUBROUTINE getbd(a,lla,lua,case)

*********************************************************************
* Referenced by:                                 subroutine print_dist *
* Find upper and lower limits of cumulative distribution stored     *
* in array a. Shift distribution so that lower limit is in a(0).    *
*********************************************************************
      integer   lla,lua,llanew, case
      real      a(0:2301,30)
      integer   k

      do 10 k=lla,lua
        llanew = k
        if (a(k-lla,case).ge. 0.0001) goto 20
 10   continue
      write(*,*) 'Error in getbd'
      stop
 20   do 30 k = llanew,lua
        if (llanew.ne.lla) a(k-llanew,case) = a(k-lla,case)
        if (a(k-llanew,case).gt. .9999 ) then
           lua = k
           a(lua-llanew,case) = 1.0
           goto 40
        endif
 30   continue
 40   lla = llanew
      return
      end
* (* subroutine getbd *)
```

```
      SUBROUTINE get_iter(iter,iwt,iflag)

*********************************************************************
* Referenced by:                                  subroutine main_up *
* Determine the number of iterations to perform. Currently, this    *
* subroutine calculates the number of iterations needed to make     *
* approximately 400,000 random draws. The recommended number of     *
* iterations is then taken to be the minimum of 100 and this        *
* calculated number.                                                *
*********************************************************************
      integer iter, iwt, iflag

      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer n_inv
        real    ia_inv, rdep, inv

      real      maxref
      logical   ans_yes

      if (iflag.ge.0) then
        iter = iflag
      else
        maxref  = rdep(n_inv-1)               !maximum # of refills
        iter = max0(int(5.0e5/(iwt+maxref)),500)
        iter = min0(iter, 2000)
c       write(*,10) iter                                          !!
        if (iflag.eq.-2) then
          write(*,15)
          if (.not.ans_yes(1)) then
            write(*,30)
            read(*,*) iter
          endif
        endif
      endif
      return

 10   format('    The default number of iterations is ',i10)
 15   format('    Is this ok?')
 30   format(
     &  '    Enter the number of iterations for this case. --> ')
      end
* (* subroutine get_iter *)
```

```
      SUBROUTINE initialize(iwt,ntar,nplat,replen)

*********************************************************************
* Referenced by:                                  subroutine main_up *
* Initialize variables and arrays in calculations.                   *
*********************************************************************
      integer iwt, ntar, nplat
      real    replen

      common/ddown/prdd(0:2301,30),lldd,ludd,ddmean
        integer lldd, ludd
        real prdd, ddmean
      common/supply/prsup(0:2301,30),llsu(30), lusu(30), smean(30)
        integer llsu, lusu
        real prsup, smean
      common/runout/rmean(30)
        real rmean
      common/target/targ(0:2301,30),llt(30), lut(30), tmean(30)
        integer llt, lut
        real targ, tmean
      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer n_inv
        real ia_inv, rdep, inv

      integer    j

      lldd    = 0
      ludd    = 0
      call zero(prdd,int(iwt/replen),1)

      llsu(1) = 0
      lusu(1) = 0
      prsup(0,1) = 1.0

      llsu(n_inv) = iwt
      lusu(n_inv) = iwt
      prsup(0,n_inv) = 1.0

      rmean(1) = float(nplat)
      rmean(n_inv) = 0.

      llt(1)  = 0
      lut(1)  = 0
      targ(0,1) = 1.0

      llt(n_inv)  = ntar
      lut(n_inv)  = ntar
      targ(0,n_inv) = 1.0

      do 10 j = 2, n_inv-1
        llsu(j) = 0
        lusu(j) = 0
        rmean(j) = 0.
        llt(j)  = 0
        lut(j)  = 0
        call zero(prsup,iwt,j)
        call zero(targ,ntar,j)
 10   continue
      return
      end
* (* subroutine initialize *)
```

```
      SUBROUTINE inventory(exp_total)
*************************************************************************
* Referenced by:                                        subroutine mainsub *
*                                                                          *
* Determine the inventories at which to evaluate distributions.  At        *
* most 50 inventories are allowed and must be split between combatant      *
* only (depot empty) and combatant+depot.  Note that at least 6            *
* inventories are required for later parts of the program to work.         *
*                                                                          *
* Referenced:                                             function smint *
*************************************************************************
      integer         max_inv,min_inv
      parameter       (max_inv=30,min_inv=6)

      integer         exp_total

      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer n_inv
        real    ia_inv, rdep, inv
      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia

      integer         max_steps,step_size_dep,num_steps_dep
      integer         step_size_plat,num_steps_plat,smint,i
      real            dep_rounds,plat_rounds

      dep_rounds=amax1((exp_total+1)-ia,0.0)
      step_size_plat=max0(exp_total/nplat,1)
      max_steps=max_inv-(smint(ia/step_size_plat)+1)
      if ((dep_rounds.gt.0.0).and.(max_steps.lt.1)) max_steps=1
      step_size_dep=max0(smint(dep_rounds/(max_steps*replen))-1,0)
10    continue                                                !from 20
         step_size_dep=step_size_dep+1
         num_steps_dep=smint(dep_rounds/(step_size_dep*replen))
20    if ((float(num_steps_dep)/max_inv.gt.
     &     dep_rounds/(exp_total+1)).and.(num_steps_dep.gt.1)) goto 10
* This test ensures that a disproportionate number of steps are not      *
* set aside for the depot.                                                *
      plat_rounds=amin1(ia,float(max0(exp_total+1,min_inv-1)))
      max_steps=max_inv-num_steps_dep-1
      step_size_plat=min0(smint(plat_rounds/max_steps),step_size_plat)
      num_steps_plat=smint(amin1(ia,plat_rounds)/step_size_plat)
      n_inv=min0(num_steps_plat,max_steps)+1
      do 50 i=1,n_inv
         ia_inv(i)=amin1(float((i-1)*step_size_plat),ia)
         rdep(i)=0
50    continue
      ia_inv(n_inv)=plat_rounds
      num_steps_dep=max0(num_steps_dep,min_inv-n_inv)
      do 70 i=n_inv+1,n_inv+num_steps_dep
         ia_inv(i)=ia
         rdep(i)=amin1(float((i-n_inv)*step_size_dep),dep_rounds/replen)
70    continue
      n_inv=n_inv+num_steps_dep
      do 80 i=1,n_inv
         inv(i)=(nplat*ia_inv(i))+(rdep(i)*replen)
80    continue
      return

      end
*  (* subroutine inventory *)
```

```
      SUBROUTINE   moe(nplat,ntar,lun,scale,title)

*********************************************************************
* Referenced by:                                  subroutine main_up *
* Determine and print for each inventory evaluated, the average     *
* shooters that do not run out, the probability that at least       *
* 80% of the targets are killed, and the probability that all the   *
* targets are killed.                                               *
* Interpolate to obtain the inventories needed so that:             *
*   1.) X% of the shooters not to run out on average,               *
*   2.) at least 80% of the targets are killed with X% confidence,  *
*   3.) all of the targets are killed with X% confidence,           *
* for X% = 75%, 80%, 85%, 90%, 95%, and 99%.                        *
*********************************************************************
      integer nplat, ntar, lun
      real scale
      character*45 title

      common/EXPECTED_REQ/ expected_req_array(6)                    ! shared with
cal_thrt_model_reqirements_for_munition_type
        real        expected_req_array
      common/runout/rmean(30)
        real        rmean
      common/target/targ(0:2301,30),llt(30), lut(30), tmean(30)
        integer     llt, lut
        real        targ, tmean
      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer     n_inv
        real        ia_inv, rdep, inv
      common/percents/perckill(6),nperc
        integer     nperc
        real        perckill

      integer   j, k, l
      real ave_sh(30), ave_ta(30), conf_ta(30,6)
      real inv_sh(6), inv_ata(6)
      real inv_conf(6,6)
      real perc(6), killperc, fval
      data perc/.75,.8,.85,.9,.95,.99/


c     write(lun,40) 'Evaluated:   ', (100*perckill(j),j=1,nperc)      !!
c     write(lun,41)                                                   !!
      do  5 k=1, n_inv
c       ave_sh(k)  = 1. - rmean(k)/nplat
        ave_ta(k) = tmean(k)/ntar
        do 90 j = 1, nperc
          if (perckill(j).ge.0.99999) then
            conf_ta(k,j)  =
     &           1. - fval(targ,llt(k),lut(k),ntar-1,k)
          else
            conf_ta(k,j)  =
     &          killperc(perckill(j),targ,llt(k),lut(k),ntar,k)
          endif
 90     continue
c       write(lun,50) int(inv(k)/scale +.5), ave_sh(k), ave_ta(k),     !! RL   v   uncommented
c    &               (conf_ta(k,j),j=1,nperc)
  5   continue

c     call interpsetup(n_inv,inv,ave_sh)
c     do 10 j = 1, 6
c        call interp(n_inv,inv,ave_sh,inv_sh(j),perc(j))

c       print*,'do 10 :   j = ', j                                    !#

c10   continue

      call interpsetup(n_inv,inv,ave_ta)
      do 35 j = 1, 6
```

```
        call interp(n_inv,inv,ave_ta,inv_ata(j),perc(j))
 35   continue                                                    !!       ^

c     do 30 l = 1, nperc
c       call interpsetup(n_inv,inv,conf_ta(1,l))
c       do 20 j = 1, 6                                          !! RL   6 <- 2
c         call interp(n_inv,inv,conf_ta(1,l),inv_conf(j,l),perc(j))
c20     continue
c30   continue

c     write(lun,60) ' Interpolated:', (100*perckill(j),j=1,nperc)   !! RL  v  uncommented
c     write(lun,61)
      do 80 k = 1, 6
c       write(lun,70) int(100*perc(k)+.5),int(inv_sh(k)/scale+.5),
c    &    int(inv_ata(k)/scale+.5),
c    &    (int(inv_conf(k,i)/scale+.5),i=1,nperc)
        expected_req_array(k) = inv_ata(k)/scale                  !! Average requirements needed for
                                                                  !! k  fractions of the targets killed
 80   continue

      write(lun,75)
      write(lun,76)  (expected_req_array(k), k=1,6)               !! Average requirements needed for
                                                                  !! k  fractions of the targets killed

c     if (inv_conf(2,1)/scale.gt.ia_inv(n_inv)*nplat) then
c       write(lun,69) title(1:11),nint(inv_conf(2,1)/scale),'*'
c     else
c       write(lun,69) title(1:11),nint(inv_conf(2,1)/scale),' '
c     endif                                                       !! RL  ^

      return
69    format(a11,i7,3x,a1)
 40   format(/,a13,//,
     &   '          Average Fraction of:         ',
     &      'Probability that at Least Y ',/,
     &   '           --------------------          ',
     &      'of the Targets are Killed ',/,
     &   '            Shooters                     ',
     &      '      for Y =',/,
     &   '          not Running  Targets    ---------',
     &      '--------------------------------',/,
     &   'Inventory     Out      Killed    ',6(f6.1,'%'))
 41   format(
     &   '---------  -------------------  ',
     &    ' ----------------------------------------')
 50   format(i8,f9.2,4x,f6.2,6x,f6.2,7f7.2)

 60   format(/,a13,//,
     &   '          Inventory Needed     ',/,
     &   '         so that on Average    ',/,
     &   '            X of the:          ',
     &      '   Inventory Needed to Kill at Least  ',/,
     &   '        --------------------  ',
     &      '   Y of the Targets with X Confidence',/,
     &   '          Shooters   Targets    ',
     &      '              for Y =',/,
     &   '           Do Not     are       ',
     &      '-------------------------------------------',/,
     &   '  X      Run Out    Killed   ',6(f7.1,'%'))
 61   format(
     &   ' -----   --------------------  ',
     &      '-------------------------------------------')
 70   format(i4,'%',i11,i10,i11,5i8)
 75   format('Average Requirements Needed for  X %  Targets Killed :',/,
     .  10x,'X :',4x,'75%',7x,'80%',7x,'85%',7x,'90%',7x,'95%',7x,'99%')
c 76   format(10x,6i10)
 76   format(10x,6f10.3)
      end
*  (* subroutine moe *)
```

```
      SUBROUTINE monte(iwt,ntar,iter)

*********************************************************************
* Referenced by:                                  subroutine main_up *
* This is the subroutine that controls the Monte Carlo calculations.*
*********************************************************************
      integer iwt, ntar, iter

      common/ddown/prdd(0:2301,30),lldd,ludd,ddmean
        integer lldd, ludd
        real prdd, ddmean
      common/supply/prsup(0:2301,30),llsu(30), lusu(30), smean(30)
        integer llsu, lusu
        real prsup, smean
      common/runout/rmean(30)
        real rmean
      common/target/targ(0:2301,30),llt(30), lut(30), tmean(30)
        integer llt, lut
        real targ, tmean
      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer n_inv
        real ia_inv, rdep, inv
      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia

      integer   nu(2001),nz(2,2001),lltp, lutp, n_dd(2301), ns, nzt
      integer   i, k, js, it
      real*8    tartemp2(0:2301,30), tartemp1(0:2301,30)

      do 10 ns = 1,iter
c       if (mod(ns,max0(1,iter/50)).eq.0)
c    &     write(*,*)'     Working on iteration #',ns,
c    &               '  (',ns*100./iter,'% )'
        call box(iwt,nplat,nu)
        call ref_setup(nu,nz,nzt,n_dd)
        prdd(nzt,1) = prdd(nzt,1) + 1.
        if (nzt.gt.ludd) ludd = nzt

        do 20 i = 2, n_inv-1
          if (rdep(i).gt.0) call refills(i,rdep,nz,nzt,n_dd)
          call analyze(i,iwt,replen,nplat,nu,nz)
 20     continue
 10   continue
      write(*,*)  '   Finished  ',iter, ' iterations.'                !!

      write(*,*)  '   Calculating target distributions.'
      call dist(prdd,lldd,ludd,iter, 1)
c     write(*,70) 1,100./(n_inv-1.)                                   !!
      do 30 k = 2, n_inv-1
c       if (mod(k,max0(1,(n_inv-1)/5)).eq.0)                          !!
c    &    write(*,70) k,k*100./(n_inv-1.)                            !!
        call zerod(tartemp2,0,ntar,k)
        call dist(prsup,llsu(k),lusu(k),iter,k)
        rmean(k) = rmean(k)/iter
        do 40 js = llsu(k),lusu(k)
          if (prsup(js,k).gt.1.0e-15) then
            call calctarget(iwt,js,ntar,tartemp1,lltp,lutp,k)
            if (lutp.gt.lut(k)) lut(k) = lutp
            call cumsum(tartemp2,0,lut(k),tartemp1,lltp,lutp,
     &           prsup(js,k),k)
          endif
 40     continue
        do 60 it=0,lut(k)
          if (tartemp2(it,k).gt.1.0e-37) then
            targ(it,k) = tartemp2(it,k)
          else
            targ(it,k) = 0.0
```

```
         endif
 60     continue
 30    continue

       return
 70    format('      Inventory #',i3,' (',f6.2,'% )')
       end
*      (*   subroutine monte   *)



       real FUNCTION killperc(perc,targ,llt,lut,ntar,case)

*********************************************************************
* Referenced by:                                    subroutine moe *
* Linear interpolate to get the probability of killing at least    *
* perc of the targets.  (perc is expressed as a fraction)          *
*********************************************************************
       integer llt, lut, ntar,case
       real    perc, targ(0:2301,30)

       real    tperc, f, fval
       integer itperc

       tperc = perc*ntar
       itperc = int(tperc)
       f = tperc - itperc
       killperc = f*(1.-fval(targ,llt,lut,itperc,case)) +
      &         (1.-f)*(1.-fval(targ,llt,lut,itperc-1,case))
       return
       end
* (* function killperc *)



       SUBROUTINE print_dist(name,a,lla,lua,mean,lun,case)

*********************************************************************
* Referenced by:                                 subroutine summary *
* Print distribution arry to output device. Calculate mean and     *
* variance.                                                        *
*********************************************************************
       integer lla, lua, lun, case
       real a(0:2301,30), mean, var
       character*4 name

       call cdf(a,lla,lua,case)
       call getbd(a,lla,lua,case)
       call calc_mean(a,mean,var,lla,lua,case)

* Print out Distributions:

*      write(lun,10) name, lla, lua, mean, var, sqrt(var)
*      write(lun,20) (a(i,case),i=0,lua-lla)
*      write(lun,20) lua-lla+1,float(lla),a(0,case)
*      write(lun,21) (float(i),a(i-lla,case)-a(i-lla-1,case),i=lla+1,lua)
 10    format(/,'***** ',a4,' Distribution ****',/,
     &         '  Lower Bound: ',i5,/,
     &         '  Upper Bound: ',i5,/,
     &         '  Mean       : ',f10.5,/,
     &         '  Variance   : ',f10.5,'  StDev:',F10.5/)
 20    format(10f8.4)
*20    format(i5,' points in data set :   name',/,2f23.6)
 21    format(2f23.6)
       return
       end
* (* subroutine print_dist *)
```

```
      SUBROUTINE refills(i,rdep,nz,nzt,n_dd)

*********************************************************************
* Referenced by:                                    subroutine monte *
* For the i-th inventory, distribute the refills available in      *
* depot, rdep(i), over the shooters who ask for refills. Note      *
* that the refills available in the depot at the (i-1)th inventory,*
* rdep(i-1), have already been distributed so that we only need to *
* distribute the remaining refills.                                *
*********************************************************************
      integer  nz(2,2001), i, nzt, n_dd(2301)
      real     rdep(30)

      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia

      integer     l, index, iurn
      real        randm

      do 30 l = int(rdep(i-1)+.001)+1, min0(nzt, int(rdep(i)+.001))
        index =  int(randm(1)*(nzt - l + 1) + 1)
        iurn = n_dd(index)
        nz(2,iurn) = nz(2,iurn) - 1
        nz(1,iurn) = nz(1,iurn) + 1
        n_dd(index) = n_dd(nzt - l + 1)
  30  continue
      return
      end
*     (*   subroutine refills   *)



      SUBROUTINE ref_setup(nu,nz,nzt,n_dd)

*********************************************************************
* Referenced by:                                    subroutine monte *
*  nzt     = total number of refills demanded by the shooters      *
*            for the current iteration.                            *
*  nz(1,i) = of the total refills demanded by the i-th platform,   *
*            this is number of refills that were satisfied.        *
*  nz(2,i) = of the total refills demanded by the i-th platform,   *
*            this is number of refills that were unsatisfied.      *
* Initially set all the refills as unsatisfied. As refills are     *
* made available in the depot, refills in nz(2,*) will be trans-   *
* ferred over to nz(1,*). This transfer occurs in subroutine       *
* refills.                                                         *
*********************************************************************
      integer nu(2001), nz(2,2001), nzt, n_dd(2301)

      common/plat/nplat,iaptot,replen,ia,rp
        integer   nplat
        real      iaptot, replen, rp, ia

      integer     i, ic, m

      nzt  = 0
      ic = 0
      do 20 m = 1, nplat
        nz(1,m) = 0
        nz(2,m) = max0(0,int((nu(m)-ia+rp+replen)/replen+.01))
        nzt = nzt + nz(2,m)
        do 10 i = 1,nz(2,m)
          ic = ic + 1
          n_dd(ic) = m
  10    continue
    20 continue
```

```
      return
      end
* (* subroutine ref_setup *)



      integer FUNCTION smint(x)
************************************************************************
* Referenced by:                                    subroutine inventory *
*                                                                        *
* Determine the smallest integer that is greater than or equal to x.     *
************************************************************************
      real            x

      if (x.lt.0.0) then
         if (ifix(x)-x.gt.0.99999) then
            smint=nint(x)
         else
            smint=ifix(x)
         endif
      else
         if (x-ifix(x).lt.0.00001) then
            smint=nint(x)
         else
            smint=ifix(x)+1
         endif
      endif
      return

      end
* (* function smint *)



      SUBROUTINE starttar(ndup,nsup,ntar,nr,pr1)

************************************************************************
* Referenced by:                                    subroutine target *
* Calculate the probability given ndup and nsup that  nr targets   *
* are killed out of ntar targets.                                        *
*                                                                        *
* pr1 = (nsup take nr)*(ndup-nsup take ntar-nr)/(ndup take ntar)   *
*                                                                        *
************************************************************************
      integer ndup, nsup, ntar, nr
      real*8  pr1

      integer   i, j

      pr1 = 0.0
      if (nsup.ne.ndup) then
        do 10 j=0,nr-1
           pr1=pr1+dlog(dble(nsup-j)*(ntar-j)/(nr-j)/(ndup-j))
  10    continue
        do 20 i=0,ntar-nr-1
           pr1 = pr1+dlog(dble(ndup-nsup-i)/(ndup-nr-i))
  20    continue
      endif
      pr1 = dexp(pr1)
      return
      end
* (* subroutine starttar *)
```

```
      SUBROUTINE summary(title,nplat,ntar,lun,iter,ncase)

*********************************************************************
* Referenced by:                                 subroutine main_up *
* Print the distribution of refill demands on the depot, DDWN.      *
* Print out for each inventory the distribution of supply, the      *
* distribution targets killed, and the distribtion of platforms     *
* running out.                                                      *
*********************************************************************
      integer nplat, ntar, lun, iter, ncase
      character*45 title
c     real temp(0:2301,30)

      common/theat/theatnames(6)
        character*45 theatnames
      common/ddown/prdd(0:2301,30),lldd,ludd,ddmean
        integer lldd, ludd
        real prdd, ddmean
      common/supply/prsup(0:2301,30),llsu(30), lusu(30), smean(30)
        integer llsu, lusu
        real prsup, smean
      common/runout/rmean(30)
        real rmean
      common/target/targ(0:2301,30),llt(30), lut(30), tmean(30)
        integer llt, lut
        real targ, tmean
      common/inventy/ia_inv(30),rdep(30),inv(30),n_inv
        integer n_inv
        real ia_inv, rdep, inv

      integer  k

*     write(lun,5) ncase, iter
 5    format(//'Monte Carlo Threat Analysis for case ',i2,
     &         ' using ',i6,' iterations'/)
*     write(lun,30) title, theatnames(ncase), ntar, nplat
 30   format('Weapon  : ',a45,/
     &       'Theater : ',a45,//
     &       'Number of Targets     =',i4,/,
     &       'Number of Shooters    =',i4)
      call print_dist('DDWN',prdd,lldd,ludd,ddmean,lun,1)
      do 10 k = 1, n_inv
*       write(lun,20) inv(k),ia_inv(k),rdep(k)
 20     format(//' Inventory                  = ',f10.2,/,
     &          ' Shooter initial allowance  = ',f10.2,/
     &          ' Refills in the depot       = ',f10.2,/)

        call print_dist('SUP ', prsup,llsu(k),lusu(k),smean(k),lun,k)
        call print_dist('TARG',targ,llt(k),lut(k),tmean(k),lun,k)

 10   continue
      return
      end
* (* subroutine summary *)
```

```
      SUBROUTINE calctarget(ndup,nsup,ntar,pr,llpr,lupr, case)

*********************************************************************
* Referenced by:                                    subroutine monte *
* This subroutine computes the distribution of targets killed       *
* conditioned on the total demand, ndup, and the total supply,      *
* nsup.                                                             *
*********************************************************************
      integer ndup,nsup,ntar,llpr,lupr, case
      real*8  pr(0:2301,30)

      integer j, mlpr, ix
      real*8 cum

      if (ndup.eq.nsup) then         !all targets killed
        llpr  = ntar
        lupr  = ntar
        pr(0,case) = 1.
      else
        call zerod(pr,0,ntar,case)
        j = ndup - nsup
        llpr = max0(0,ntar - j)
        lupr = min0(ntar,nsup)
        mlpr = nsup*ntar/ndup
        call starttar(ndup,nsup,ntar,mlpr,pr(mlpr-llpr,case))
        cum = pr(mlpr-llpr,case)
        do 40 ix= mlpr+1,lupr
          pr(ix-llpr,case) = pr(ix-1-llpr,case)*(nsup-ix+1)*(ntar-ix+1)/
     &                  ix/(j-ntar+ix)
          cum = cum + pr(ix-llpr,case)
          if (pr(ix-llpr,case).lt. 1.0d-15) goto 45
 40     continue
 45     do 50 ix = mlpr-1,llpr,-1
          pr(ix-llpr,case) = pr(ix+1-llpr,case)*(ix+1)*(j-ntar+ix+1)/
     &                  (nsup-ix)/(ntar-ix)
          cum = cum + pr(ix-llpr,case)
          if (pr(ix-llpr,case).lt. 1.0d-15) goto 55
 50     continue
 55     if (cum.lt. 0.99 .or. cum.gt. 1.001) then
          write(*,*) 'Error in subroutine target:  cum = ', cum
          stop
        endif
      endif
      return
      end
* (* subroutine target *)



      SUBROUTINE zero(pr,max,case)

*********************************************************************
* Referenced by:                                 subroutine initialize *
* Zero single precision array.                                      *
*********************************************************************
      integer max, case
      real    pr(0:2301,30)

      integer  j

      do 10 j=0,max
        pr(j,case) = 0.0
 10   continue
      return
      end
* (* subroutine zero *)
```

```
      SUBROUTINE zerod(dpr,ll,lu,case)

*********************************************************************
* Referenced by:                                  subroutine monte  *
*                                                 subroutine target *
* Zero double precision array.                                      *
*********************************************************************
      integer ll, lu, case
      real*8 dpr(0:2301,30)

      integer  j

      do 10 j=0,lu-ll
        dpr(j,case) = 0.0
 10   continue
      return
      end
* (* subroutine zerod *)
```

JMAP/Threat Pair

Code File : tm_inter.for

```
c     File Name :    tm_inter.for


*********************************************************************
*                   Monte Carlo THREAT MODEL                        *
*                    (Interpolation Module)                         *
*                                                                   *
* This module computes a monotone piecewise interpolation of data.  *
* The three main subroutines are interpsetup, and interp.           *
* Interpsetup computes the coefficients of the interpolated         *
* piecewise polynomial. Once this subroutine has been called then   *
* subroutines interp may be called any number of times.             *
* Poly interpolates to get a y value for a given x value and        *
* interp interpolates to get an x value for a given y value.        *
*********************************************************************

      SUBROUTINE   interpsetup(n,x,y)

*********************************************************************
* This subroutine determines the coefficients of a piecewise cubic  *
* monotone interpolation to a set of n points (x(i),f(i)), i=1,n    *
* This method was taken from a method described in the article:     *
* "A Method for Constructing Local Monotone Piecewise Cubic         *
*  Interpolants", F. N. Fritsch and J. Butland, Siam J., Stat.      *
*  Comput., Vol 5, No. 2, June 1984                                 *
* The cubic polynomial is given by:                                 *
* p(x) = f(i)*H1(x) + f(i+1)*H2(x) + d(i)*H3(x) + d(i+1)*H4(x)      *
* where       H1(x) = phi( (x(i+1)-x)/h(i) )                        *
*             H2(x) = phi( (x-x(i))/h(i) )                          *
*             H3(x) = -h(i)*psi( (x(i+1)-x)/h(i) )                  *
*             H4(x) = h(i)*psi( (x-x(i))/h(i) )                     *
*             h(i)  = x(i+1) - x(i)                                 *
*             phi(t) = 3*t^2 - 2*t^3                                *
*             psi(t) = t^3 - t^2                                    *
*             d(i)   = p'(x(i))                                     *
* The purpose of this algorithm is to determine the derivatives     *
* d(i). This is done as follows:                                    *
* At the endpoints:                                                 *
*    d(1) and d(n) are determined using a three point               *
*    approximation. In this case we assume the function is          *
*    montonically increasing, therefore, if the three point         *
*    approx give a negative derivative then this derivative is      *
*    set to zero.                                                   *
* At the internal points:                                           *
*    Define      S(i) = (f(i+1)-f(i))/(x(i+1)-x(i))                 *
*    Then                                                           *
*      d(i) = g( S(i-1),S(i),h(i-1),h(i) )                          *
*                                                                   *
*             ¦ S(i-1)*S(i)/(a*S(i)+(1-a)*S(i-1), if s(i-1)*s(i)>0  *
*          = ¦                                                      *
*             ¦                 0               , otherwise       *
*       and                                                         *
*             a = (h(i-1)+2h(i))/(3(h(i-1)+h(i)))                   *
* Referenced by:                                    subroutine moe  *
*********************************************************************
      integer n
      real x(50), y(50)

      common/coef/a(50),b(50),c(50),e(50)
        real*8 a, b, c, e
      common/deriv/h(50),m(50),d(50)
        real*8 h, m, d

      integer i, ip
      real*8 g, threept

      do 10 i = 1,n-1
        h(i) = x(i+1) - x(i)
```

```
        m(i) = (y(i+1) - y(i))/h(i)
 10   continue

      d(1) = dmax1(dble(0.),threept(x,y,1,1))
      d(n) = dmax1(dble(0.),threept(x,y,n,n-2))
      do 30 i = 2, n-1
        d(i) = g(m(i-1),m(i),h(i-1),h(i))
 30   continue

* To solve for a given p(x) we must rewrite
* p(x) = f(i)*H1(x) + f(i+1)*H2(x) + d(i)*H3(x) + d(i+1)*H4(x)
* as p(x) = a(i) + b(i)*(x-x(i)) + c(i)*(x-x(i))^2 + e(i)*(x-x(i))^3
      do 50 ip = 1, n-1
        e(ip) = (2*(y(ip)-y(ip+1))+h(ip)*(d(ip)+d(ip+1)))/h(ip)**3
        c(ip) =
     &    (-3*y(ip)+3*y(ip+1)-2*h(ip)*d(ip)-h(ip)*d(ip+1))/h(ip)**2
        b(ip) = d(ip)
        a(ip) = y(ip)
 50   continue
      return
      end
*     (*  interpsetup   *)



      real*8 FUNCTION   threept(x,y,i,j)

**********************************************************************
* Use three point formula to estimate derivative at point x(i)       *
* using the three points x(j), x(j+1), and x(j+2)                    *
* Referenced by:                              subroutine interpsetup *
**********************************************************************
      integer   i, j
      real      x(50),y(50),x0,x1,x2,xk

      x0 = x(j)
      x1 = x(j+1)
      x2 = x(j+2)
      xk = x(i)
      threept = y(j)*(2*xk - x1 - x2)/(x0 - x1)/(x0 - x2) +
     &          y(j+1)*(2*xk - x0 - x2)/(x1 - x0)/(x1 - x2) +
     &          y(j+2)*(2*xk - x0 - x1)/(x2 - x0)/(x2 - x1)
      return
      end
*     (*   threept   *)



      real*8 FUNCTION   g(s1,s2,h1,h2)

**********************************************************************
* Function used to compute derivatives at data points (except        *
* endpoints).                                                        *
* Referenced by:                              subroutine interpsetup *
**********************************************************************
      real*8 s1, s2, h1, h2

      real*8 alpha

      if (s1*s2 .le. 0) then
        g = 0.
      else
        alpha  = (h1 + 2*h2)/(h1+h2)/3.
        g = s1*s2/(alpha*s2 + (1.-alpha)*s1)
      endif
      return
      end
*     (*   g   *)
```

```
      SUBROUTINE   interp(n1,x,f,xval,fval)

*********************************************************************
* For fval find the interpolated value of x:                        *
* Solve the cubic equation for ss(j)=fval  where                    *
*               f(j)=< fval < f(j+1)                                *
*                                                                   *
* ss(j) = f(j) + b(j)*(x-x(j)) + c(j)*(x-x(j))^2 + d(j)*(x-x(j))^3  *
*                                                                   *
* Find the discriminant using reduced for, calculated from          *
* the normal form of the cubic equation.                            *
*   normal form:                                                    *
*   x^3 + rx^2 + sx + t = 0 , where r = c/d, s= b/d, and t=(f-ss)/d *
*   reduced form:                                                   *
*   y^3 + py + q = 0, where p=(3s - r^2)/3, q=2r^3/27 - rs/3 + t.   *
* Referenced by:                                    subroutine moe *
*********************************************************************
      integer n1
      real x(50) ,f(50), fval, xval

      common/coef/a(50),b(50),c(50),d(50)
        real*8 a, b, c, d

      real*8     r, s, t, p, q, dc, rho, phi, u, v, sol(3), pi
      real       const, delx, error
      integer    j, k

c     if (fval.le.f(1)) then
      if (fval.lt.f(1)) then
        write(*,*) 'fval too small'
        print*, 'fval = ', fval,'  f(1) = ', f(1)
        stop
      endif
      do 20  k = 1, n1
        if (fval .eq. f(k)) then
          xval = x(k)
          return
        else if (fval .lt. f(k))  then
          goto 30
        endif
   20 continue
   30 j = k - 1
      if (d(j).eq.0.) then
        if (c(j).eq.0) then
          if (b(j).eq.0.) then
            write (6,*) 'problem in interp -- x can be anything'
          else                                        !linear equation
            xval = (fval-f(j))/b(j) + x(j)
          endif
        else                                          !quadratic equation
          dc = b(j)*b(j)-4.*(f(j)-fval)*c(j)
          if (dc.lt.0) then
            write (6,*) 'problem in interp -- no solution'
          else if (dc.eq.0) then
            sol(1) = -b(j)/2./c(j) + x(j)
            call checkrng(sol,1,x(j),x(j+1),xval)
          else
            sol(1) = (- b(j) + dsqrt(dc))/2./c(j) + x(j)
            sol(2) = (- b(j) - dsqrt(dc))/2./c(j) + x(j)
            call checkrng(sol,2,x(j),x(j+1),xval)
          endif
        endif
      else                                            !cubic equation
        r = c(j)/d(j)
        s = b(j)/d(j)
        t = (f(j)-fval)/d(j)
        p = (3.*s - r*r)/3.
        q = (2.*(r*r*r))/27. - (r*s)/3. + t
        dc = (p/3.)*(p/3.)*(p/3.) + (q/2.)*(q/2.)
```

```
        if (dc .gt. 0) then                    ! one real solution
          u = -q/2. + dsqrt(dc)
          u = dsign(dabs(u)**(1./3.),u)
          v = -q/2. - dsqrt(dc)
          v = dsign(dabs(v)**(1./3.),v)
          sol(1) = (u + v - (r /3.0)) + x(j)
          call checkrng(sol,1,x(j),x(j+1),xval)
        else if (dc .eq. 0) then               ! two real solutions
          u = -q/2. + dsqrt(dc)
          u = dsign(dabs(u)**(1./3.),u)
          sol(1) = (2.*u - (r /3.0)) + x(j)
          sol(2) = (-u - (r /3.0)) + x(j)
          call checkrng(sol,2,x(j),x(j+1),xval)
        else                                   ! three real solutions
          rho = dsqrt(-p*p*p/27.)
          phi = dacos(-q/2./rho)
          pi  = 4.*datan(dble(1.))
          const = 2.*dsign(dabs(rho)**(1./3.),rho)
          sol(1) = (const * dcos(phi/3.)           - r/3.) + x(j)
          sol(2) = (const * dcos(phi/3. + 2.*pi/3.) - r/3.) + x(j)
          sol(3) = (const * dcos(phi/3. + 4.*pi/3.) - r/3.) + x(j)
          call checkrng(sol,3,x(j),x(j+1),xval)
        endif
      endif

* Check solution
      delx = xval-x(j)
      error = dabs(f(j)-fval + b(j)*delx + c(j)*delx*delx +
     &           d(j)*delx*delx*delx)
      if (error.gt.0.001) then
        write(*,*) ' Problem in interp: Error = ',error
      endif

      return
      end
*     (*   interp   *)


      SUBROUTINE   checkrng(sol,n,xl,xu,solution)

**********************************************************************
* Find all solutions in correct range. If the number of solutions  *
* is greater of less than one, signal error.                       *
* Referenced by:                                  subroutine interp *
**********************************************************************
      integer n
      real    xl, xu, solution
      real*8  sol(3)
      integer nsol, j

      nsol = 0
      do 10 j = 1, n
        if (sol(j) .ge. xl*.999995 .and. sol(j).le. xu*1.000005) then
          solution = sol(j)
          nsol  = nsol + 1
        endif
 10   continue

      if (nsol.eq.0 .or. nsol.gt.1) then
        write(*,*)  ' Problem in interp: Calculated solutions are: '
        write(*,*)  (sol(j),j=1,n)
        write(*,*)  '   Allowed range = ', xl,' to ',xu
        if (nsol.eq.0) write(*,*) ' No solutions '
        if (nsol.gt.1) write(*,*) ' Too many solutions '
        stop
      endif
      return
      end
*     (*   checkrng   *)
```

```
      SUBROUTINE   poly(n,x,y,xd,yd)

*********************************************************************
* For a given xd value return the function value yd .               *
* Referenced by:                                    subroutine moe *
*********************************************************************
      integer n
      real x(50), y(50), xd, yd

      common/deriv/h(50),m(50),d(50)
        real*8 h, m, d

      real*8    phi, psi, xu, xl, t
      integer   ip, j

      phi(t) = t*t*(3. - 2*t)
      psi(t) = t*t*(t-1.)

      ip = 0
      do 20 j=1,n-1
        if (xd.ge.x(j) .and. xd.le.x(j+1)) then
          ip = j
          goto 10
        endif
 20   continue

 10   if (ip.eq.0) then
        print *,'x out of range in subroutine poly'
        stop
      else if (xd.eq.x(j)) then
         yd = y(j)
      else if (xd.eq.x(j+1)) then
         yd = y(j+1)
      else
        xu = (x(ip+1) - xd)/h(ip)
        xl = (xd - x(ip))/h(ip)
        yd = y(ip)*phi(xu) + y(ip+1)*phi(xl) -
     &         d(ip)*h(ip)*psi(xu)  +  d(ip+1)*h(ip)*psi(xl)
      endif
      return
      end
*     (*   poly   *)
```

JMAP/Threat Pair

Code File : tm_rand.for

```
c     File Name :    tm_rand.for



      real  FUNCTION randm(icheck)

c     ************************************************************************
c     * Pick a random number between zero and one.                           *
c     * Referenced by:                              subroutine box           *
c     *                                             subroutine refills       *
c     *                                             subroutine main_up       *
c     ************************************************************************

      integer icheck

      common/misc/ncase,ncases,lun,beta,a,exwt,title
        integer ncase, ncases, lun
        real beta, a, exwt
        character*45 title

      real rmean
      integer y, ndr
      save y, ndr, rmean

      if (icheck.eq.0) then
*       write(lun,*) ' Random Number Generator: 2^16 cycle'
        y = 1103                    !seed
        ndr = 0
        rmean = 0.
      else if (icheck.eq.2) then
*       write(lun,*)'# of random draws = ',ndr,'   mean = ',rmean/ndr
      else
        ndr = ndr + 1
        y = y*25173 + 13849
        y = mod(y,65536)
        randm = float(y)/65535
        rmean = rmean + randm
      endif

      return
      end
*     (*   randm   *)
```

JMAP/Threat Pair

Code File : pair_1.inc

```
c     File Name :   pair_1.inc


      integer       maxi, maxj, maxk, maxl, maxrows
      PARAMETER (MAXI=25,MAXJ=25,MAXK=25,MAXL=3,MAXROWS=1000)


      COMMON /ATT/ ATT(MAXI,MAXJ,MAXK)                                  !  ATT <- EXCHANGE
        real       ATT
      COMMON /CM/ CM(MAXJ)                                              !  Munition cost
        real       CM
      COMMON /COSTGOAL/ COSTGOAL                                        !  Cost goal computed by the Threat Model
        real       COSTGOAL
      common /CG_REDUCTION/ costgoal_reduction_factor                   !  Cost goal reduction factor
        real        costgoal_reduction_factor
      common /DELTA/ delta(MAXI,MAXK), TIK_0(MAXI,MAXK)                 !  Overlapping bounds for targets to be killed
        real       delta                                                !     and initial target splits by platform
      common /EJ/ EJ(MAXJ), EJ_old(MAXJ)                              ! Threat Model expenditures of two consecutive passes
        real        EJ, EJ_old
      COMMON /FATTR/ FATTR(MAXI)                                        !  Platform attrition fractions
        real       fattr
      COMMON /FTBK/ FTBK(MAXK,MAXL)                                     !  Minimum fractions of targets to be killed
        real       FTBK
      common /IA_RS_RP/ IA(MAXI,MAXJ), RS(MAXI,MAXJ), RP(MAXI,MAXJ)    !  Initial Allowance, Refill-Size, Reorder-Point
        real       ia, rs, rp
      common /JMAP_SUM/ totatt(MAXI), tot_mun_req_cost, totkill(MAXK)  ! JMAP output - platform attrition, munition cost
        real      totatt, tot_mun_req_cost, totkill                   !                  targets killed
      common /KC_0/ KC_0(MAXJ), KC(MAXJ),DIFF(MAXJ)                    !  Kill Criterion by Munition Type
        real       KC_0, KC, DIFF
      common /MAX_PASS/ max_pass                                        !  Maximum number of pair-iterations
        real       max_pass
      COMMON /NAMES/ PLAT(MAXI),MUN(MAXJ),TARG(MAXK)                    !  Names of platform, munition, and target types
        CHARACTER*15 PLAT,MUN,TARG
      COMMON /NBIJKL/ NBI,NBJ,NBK,NBL,NIJK                              !  # of types of platform, munition, and target
        integer    nbi, nbj, nbk, nbl, nijk
      COMMON /PLATCOST/ PLATCOST(MAXI)                                  !  Platform cost
        real       PLATCOST
      COMMON /PQ_TQ/ PQ(MAXI),TQ(MAXK)                                  !  # of platforms and targets of each type
        real       pq, tq
      common /PREPRE/ prepri(MAXROWS)                                   !  priority level of cost goal, kill goals,
        integer    prepri                                               !     and attrition goals
      common /R_E/ R_E(MAXJ)                                            !  Requirement-to-Expenditure ratio
        real       R_E
      common /REQ/ reqsum(maxj)                                         !  JMAP+ requirements by munition type
        real       reqsum
      COMMON /RND/ RNDSUM(MAXJ)                                         !  JMAP+ expenditures by munition type
        real       RNDSUM
      COMMON /RPE/ RPE(MAXI,MAXJ,MAXK)                                  !  Rounds-per-Engagement,  RPE <- SALVO
        real       rpe
      COMMON /STK/ STK(MAXI,MAXJ,MAXK,MAXL)                             !  Salvo-to-Kill,   STK  <-  SALVOTOK
        real       stk
      common /STOPPING/ stopping_tolerance                              !  Stopping tolerance
        real       stopping_tolerance
      common /TIJK/ TIJK(maxi,maxj,maxk)                                !  Targets killed
        real       tijk
      common /T_OVLP_CONF/ target_overlap_confidence                    !  Target overlap confidence
        real          target_overlap_confidence
```

JMAP/Threat Pair

Code File : pair_2.inc

```
c     File Name :   pair_2.inc


      integer       maxi, maxj, maxk, maxl
      PARAMETER (MAXI=25,MAXJ=25,MAXK=25,MAXL=3)
      PARAMETER (MAXX=25000,MAXROWS=1000,MAXDEVS=2*MAXROWS)
      PARAMETER (MAXPLATS=10000,MAXTARGS=10000)
      PARAMETER (MAXCFS=25000,MAXCFS2=2*MAXCFS)
      PARAMETER (MAXCOLS=2500,MAXPRI=20)


      COMMON /ATT/ ATT(MAXI,MAXJ,MAXK)                              !   ATT <- EXCHANGE
        real       ATT
      COMMON /CM/ CM(MAXJ)                                          !   Munition cost
        real       CM
      COMMON /COSTGOAL/ COSTGOAL                                    !   Cost goal computed by the Threat Model
        real       COSTGOAL
      common /CG_REDUCTION/ costgoal_reduction_factor               !   Cost goal reduction factor
        real       costgoal_reduction_factor
      common /DELTA/ delta(MAXI,MAXK), TIK_0(MAXI,MAXK)             !   Overlapping bounds for targets to be killed
        real       delta                                            !       and initial target splits by platform
      common /EJ/ EJ(MAXJ), EJ_old(MAXJ)                          ! Threat Model expenditures of two consecutive passes
        real       EJ, EJ_old
      COMMON /FATTR/ FATTR(MAXI)                                    !   Platform attrition fractions
        real       fattr
      COMMON /FTBK/ FTBK(MAXK,MAXL)                                 !   Minimum fractions of targets to be killed
        real       FTBK
      common /IA_RS_RP/ IA(MAXI,MAXJ), RS(MAXI,MAXJ), RP(MAXI,MAXJ)  !  Initial Allowance, Refill-Size, Reorder-Point
        real       ia, rs, rp
      common /JMAP_SUM/ totatt(MAXI), tot_mun_req_cost, totkill(MAXK)  ! JMAP output - platform attrition, munition cost
        real       totatt, tot_mun_req_cost, totkill                !                  targets killed
      common /KC_0/ KC_0(MAXJ), KC(MAXJ), DIFF(MAXJ)                !   Kill Criterion by Munition Type
        real       KC_0, KC, DIFF
      common /MAX_PASS/ max_pass                                    !   Maximum number of pair-iterations
        real       max_pass
      COMMON /NAMES/ PLAT(MAXI),MUN(MAXJ),TARG(MAXK)                !  Names of platform, munition, and target types
        CHARACTER*15 PLAT,MUN,TARG
      COMMON /NBIJKL/ NBI,NBJ,NBK,NBL,NIJK                          !   # of types of platform, munition, and target
        integer    nbi, nbj, nbk, nbl, nijk
      COMMON /PLATCOST/ PLATCOST(MAXI)                              !   Platform cost
        real       PLATCOST
      COMMON /PQ_TQ/ PQ(MAXI),TQ(MAXK)                              !   # of platforms and targets of each type
        real       pq, tq
      common /PREPRI/ prepri(MAXROWS)                               !   priority level of cost goal, kill goals,
        integer    prepri                                           !       and attrition goals
      common /R_E/ R_E(MAXJ)                                        !   Requirement-to-Expenditure ratio
        real       R_E
      common /REQ/ reqsum(maxj)                                     !   JMAP+ requirements by munition type
        real       reqsum
      COMMON /RND/ RNDSUM(MAXJ)                                     !   JMAP+ expenditures by munition type
        real       RNDSUM
      COMMON /RPE/ RPE(MAXI,MAXJ,MAXK)                              !   Rounds-per-Engagement,  RPE <- SALVO
        real       rpe
      COMMON /STK/ STK(MAXI,MAXJ,MAXK,MAXL)                         !   Salvo-to-Kill,   STK  <-  SALVOTOK
        real       stk
      common /STOPPING/ stopping_tolerance                          !   Stopping tolerance
        real       stopping_tolerance
      common /TIJK/ TIJK(maxi,maxj,maxk)                            !   Targets killed
        real       tijk
      common /T_OVLP_CONF/ target_overlap_confidence                !   Target overlap confidence
        real       target_overlap_confidence


      COMMON /CES/ C(5),E(5)
      COMMON /COLROWS/ COLS(MAXCFS),ROWS(MAXCFS)
      COMMON /CRR/ RMIN(MAXI,MAXJ),RMAX(MAXI,MAXJ)
      COMMON /CT/ COEFFS(MAXCFS),T(MAXCFS2)
      COMMON /CTYPE/ CTYPE(MAXROWS)
        CHARACTER*1 CTYPE
```

```
      COMMON /DOCTRINE/ DOCTRIN(MAXJ)
      common /FOM/ FOM_P(MAXI), FOM_T(MAXK)                    !    Figure-of-Merits for platforms & targets
      COMMON /FRACLOST/ FRACLOST
      COMMON /I1/ IBASIC(MAXROWS),JCOL(MAXCOLS)
      COMMON /ISIGN/ ISIGN(MAXROWS)
      CHARACTER*1 ISIGN
      COMMON /ITER500/ ITER500
      COMMON /LOADOUT/ LOADOUT(MAXI,MAXJ)
      COMMON /MCOL/ MCOL(MAXI,MAXJ,MAXK)
      COMMON /MPSS/ MPSCOLS,MPSRHS,MPSTOT
      COMMON /NB2030/ NBF20,NBF30
      COMMON /NBCNROWS/ NBC4ROWS,NBC5ROWS,NBC6ROWS,NBC7ROWS,NBC8ROWS,
     1                  NBC9ROWS,NBCAROWS
      COMMON /NCONS/ NC1,NC2,NC3,NC4,NC5,NC6,NC7,NC8,NC9,NC10
      COMMON /NEGDEVS/ DNEGROW(MAXDEVS),DNEGPRI(MAXDEVS),DNEGWT(MAXDEVS)
      COMMON /NEGDEVS0/ DNEGROW0(MAXDEVS),DNEGPRI0(MAXDEVS),
     1   DNEGWT0(MAXDEVS)
      COMMON /NRC/ NBROWS,NBCOLS,NVAR,NBPRIORS,KTEST,ITER
      COMMON /POSDEVS/ DPOSROW(MAXDEVS),DPOSPRI(MAXDEVS),DPOSWT(MAXDEVS)
      COMMON /POSDEVS0/ DPOSROW0(MAXDEVS),DPOSPRI0(MAXDEVS),
     1   DPOSWT0(MAXDEVS)
      COMMON /TARGCHK/ TARGCHK(MAXTARGS)
      common /REQUIRE/ require(maxi,maxj)                      !   munition requirements
      COMMON /S/ S
         INTEGER S
      COMMON /STDEV/ STDEV(MAXI,MAXJ,MAXK,MAXL)
      COMMON /STOCK/ STOCK(MAXJ)
      COMMON /R1/ BASIS(MAXROWS,MAXCOLS)
      COMMON /R2/ VALC(MAXPRI,MAXCOLS),VALB(MAXPRI,MAXROWS)
      COMMON /R3/ PRHS(MAXROWS),RHS(MAXROWS)
      COMMON /RNN/ ROW,NBNZX,NCOEFFS,NPLATS
         INTEGER ROW
      COMMON /RSTEP/ RNGMIN,RNGMAX,STEP
      COMMON /ROF/ ROF(MAXI,MAXJ)
      COMMON /STKCOST/ STKCOST
      COMMON /TARGTYPE/ TARGTYPE(MAXK)
         CHARACTER*5 TARGTYPE
      COMMON /TOTIME/ TOTIME
      COMMON /TITLE/ TITLE
      COMMON /TOTARGS/ TOTARGS
      COMMON /ZK/ ZK
      COMMON /ALPHA/ ALPHA
      COMMON /TIME/ TIME(MAXL)

      DOUBLE PRECISION BASIS,VALC,VALB,PRHS,RHS

      DIMENSION TOTCOST(MAXK)
      DIMENSION TOTLKILL(MAXK,MAXL),TOTLK(MAXL)
      DIMENSION BCRNDS(MAXJ),BCFK(MAXK),BCCPK(MAXK)
      DIMENSION SUMKILL(MAXJ,MAXK)

      CHARACTER*1 YESNO
      CHARACTER*80 TITLE
      CHARACTER*132 LINE, CLINE
```