AN APPLICATION OF FORMAC TO THE COMPUTATION OF COVERAGE FUNCTIONS

Samuel D. Oman*
The RAND Corporation
Santa Monica, California

Introduction

Coverage functions arise frequently in fields such as weapon systems analysis, where it is often required to evaluate the probability that a randomly distributed point target (or a fixed target with a distributed mass) will be destroyed by one or more weapons fired simultaneously at it. Such a probability of destruction (or expected fraction of the target destroyed) can generally be expressed in terms of a set of multiple integrations whose initial integrands contain as factors the density distribution of the target, the distribution of the point of detonation for each weapon, and the probability that a weapon detonating at a given point will destroy the target (or an element of the distributed target) located at another point.** The mathematical form of this last mentioned conditional probability, often referred to as the damage function of the weapon, has a very direct bearing on the ease with which the required integrations can be performed. Unfortunately, however, most realistic damage functions currently in use do not permit the expression of these integrals in closed form. Consequently their evaluation on a digital computer requires approximation or Monte Carlo techniques, and the mathematical formulation of more complicated problems involving these integrals is sometimes difficult.

This paper deals with a method of evaluating coverage functions, significantly different from existing methods in two respects: First, the method uses a new set of damage functions that are on the one hand empirically realistic, and on the other hand are sufficiently mathematically tractable to allow fairly complicated integrals to be evaluated exactly. Second, the method is implemented on the computer by means of FORMAC, the IBM written symbolic mathematical compiler (described in Reference 1). This second aspect, which will be the primary concern of this paper, is an interesting example of how FORMAC may be used when the application of a mathematical approach to an actual "real world" problem requires cumbersome and involved computations.

In Section 1 we discuss the relationship of the damage function to the coverage function, and mention some of the limitations of existing damage functions. Section 2 deals (very briefly) with the mathematics of the new method, which is due to Mario L. Juncosa of The RAND Corporation; a more detailed exposition of the method will be forthcoming from Dr. Juncosa at a later date. In Section 3 we describe the manner in which the method was implemented via FORMAC, together with some of the difficulties encountered. Finally, Section 4 is devoted to a sample problem, in which the method was used to determine the aimpoint of a pair of weapons that would optimize the probability that the target be destroyed.

Before proceeding any further, we establish some conventions regarding our notation. We shall think in two-dimensional terms, so that our targets will be assumed distributed in the plane. Moreover, we shall not consider factors such as the height above the plane at which a weapon detonates, so that the point of detonation of a weapon will be a two-dimensional random vector distributed about the aimpoint. Unless otherwise specified, the following notation will therefore be in effect:

x will denote the vector (x_1,x_2);

$\|x\| = (x_1^2 + x_2^2)^{\frac{1}{2}}$ will denote the length of x (so that $\|x - y\|$ is the distance between x and y); (0.1)

$\int f(x)dx$ will denote $\int_{R^2} f(x)dx = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f((x_1,x_2))dx_1dx_2$;

$\int f(x)P(X \in dx)$ will denote $\int f(x)h(x)dx$ when the random variable X has the probability density function h;

*Any views expressed in this paper are those of the author. They should not be interpreted as reflecting the views of The RAND Corporation or the official opinion or policy of any of its governmental or private research sponsors. Papers are reproduced by The RAND Corporation as a courtesy to members of its staff.

**It should be noted that problems in other fields can give rise to the same type of coverage function considered here. For example, the "weapons" could be radio transmitters, the "target" could be the population of a city, and the "expected fraction of the target destroyed" could be the expected population reached by the signals.

and $P(A|B)$ will denote the conditional probability of A, given B.

1. Coverage Functions and Damage Functions

Suppose we fire one weapon at a point target whose location is given by the (two-dimensional) random variable T, and suppose the point at which the weapon detonates is given by the random variable W. Then if we let D represent the event that the target is destroyed, we have

$$P(D) = \int\left[\int P(D|T=x,\ W=y)P(W \in dy)\right]P(T \in dx). \tag{1.1}$$

P(D) is often referred to as a coverage function.

If instead we simultaneously fire N weapons such that their points of detonation are independent random variables W_i, then we have (denoting the event that the target is destroyed by weapon i as D_i, and the event that it is not by D_i')

$$\begin{aligned} P(D) &= 1 - \int_{\mathbb{R}^2} P(D_1' \cap D_2' \cap \ldots \cap D_N'|T=x)P(T \in dx) \\ &= 1 - \int_{\mathbb{R}^2}\left\{\int_{\mathbb{R}^{2N}} \prod_{i=1}^{N}\left[1 - P(D_i|T=x,W_i=y_i)\right]\right. \\ &\qquad \left. P(W_1 \in dy_1, \ldots, W_N \in dy_N)\right\} P(T \in dx) \\ &= 1 - \int_{\mathbb{R}^2}\left\{\prod_{i=1}^{N}\int_{\mathbb{R}^2}\left[1 - P(D_i|T=x,W_i=y_i)\right]\right. \\ &\qquad \left. P(W_i \in dy_i)\right\} P(T \in dx), \end{aligned} \tag{1.2}$$

the last equality following from the independence of the W_i.

Let us now assume that T has a probability density function h, that the W_i have probability density functions f_i, and that the N weapons each have the same damage function; i.e., $P(D_i|T=x,W_i=y) = g(x,y)$ for all i. Then if we expand (1.2), we see that the coverage function P(D) may be expressed as a linear combination of integrals of the form

$$J = \int h(x)\left[\prod_{i=1}^{n}\int g(x,y)f_i(y)dy\right]dx, \tag{1.3}$$

where $n \le N$.

The assumption that the N weapons have the same damage function is fairly common; moreover, it is also not unusual to assume that the W_i and T are normally distributed. For simplicity's sake, we shall make these assumptions throughout this paper (although the mathematical approach described in the next section only requires the assumption of normality of the W_i).

Examination of (1.3) shows the essential role that the specification of the damage function g plays in calculating the coverage function P(D); for g must be not only empirically realistic, but also sufficiently mathematically tractable to allow this type of multiple integral to be calculated efficiently. As an illustration, consider the so-called "cookie-cutter" damage function given by

$$k_R(x,y) = \begin{cases} 1 & \|x-y\| \le R, \\ 0 & \text{otherwise.} \end{cases}$$

Using k_R for g in (1.3) results in

$$J = \int_{\mathbb{R}^2} h(x)\left[\prod_{i=1}^{n}\int_{\|x-y\| \le R} f_i(y)dy\right]dx.$$

Assuming normally distributed W_i, we see that an integral of the type in brackets is equivalent to the integral of a circular Gaussian distribution with mean zero over an offset ellipse $\left(\frac{x_1 - u_1}{a_1}\right)^2 + \left(\frac{x_2 - u_2}{a_2}\right)^2 \le r^2$. This integral (which cannot be expressed in closed form) has been tabulated fairly completely for the special case $a_1 = a_2$, and somewhat less completely for the case $a_1 \ne a_2$.[2] Moreover, for the case n = 1, an interchange of integration allows J itself to be expressed as such an elliptical probability coverage. For n > 1, however, this interchange does not work, and consequently these tables cannot be used to evaluate P(D). Another limitation of k_R is its somewhat unrealistic assumption of a totally discrete damage pattern for the weapon, i.e., its assumption that every point in the plane is either destroyed or not destroyed with probability one.

In an attempt to overcome these difficulties, several alternative types of damage functions have been proposed and used over the years; we refer the reader to Reference 3 for a comprehensive summary of the literature in this field.

2. A Different Class of Damage Functions

The class of damage functions and the resulting approach to the evaluation of coverage functions that we describe here are due to Mario L. Juncosa of The RAND Corporation.

Throughout this section we shall fix R > 0 and let $A = \pi R^2$. We first construct a sequence of functions g_m such that each g_m is decreasing in $\|x\|$, and such that $g_m \to k_R$ as $m \to \infty$. We shall then show that for any m, (1.3) may be evaluated exactly when the damage function g(x, y) is taken to be $g_m(x-y)$. Thus we shall obtain

a class of damage functions that are more realistic than the discrete "cookie-cutter," which nevertheless can be used to approximate k_R with any degree of accuracy, and which allow (1.3) to be expressed in closed form.*

We now define the g_m:

Definition 1 For $m \geq 0$, let

$$g_m(z) = e^{-\|\alpha_m z\|^2} \sum_{k=0}^{m} \frac{\|\alpha_m z\|^{2k}}{k!}, \quad z \in \mathbb{R}^2,$$

where $\alpha_m^2 = \frac{(m+1)\pi}{A}$.

The following lemma summarizes some of the properties of the g_m:

Lemma 1

(i) $0 \leq g_m \leq 1$.

(ii) $\int_{\mathbb{R}^2} g_m(z)dz = A$, all m.

If $0 < \rho_1 < R < \rho_2$, then

(iii) as $m \to \infty$, $g_m(z) \to 1$ uniformly for $\|z\| \leq \rho_1$,

and (iv) as $m \to \infty$, $g_m(z) \to 0$ uniformly $\|z\| \geq \rho_2$.

(v) $g_m(z)$ is a monotone decreasing function of $\|z\|^2$.

(We note that by (v) it suffices to show convergence for $\|z\| = \rho_1$ and $\|z\| = \rho_2$ to prove (iii) and (iv)).

Proof: (i) is obvious. For (ii) we have upon converting to polar coordinates that

$$\int_{\mathbb{R}^2} g_m(z)dz = 2\pi \sum_{k=0}^{m} \frac{1}{k!} \int_0^\infty e^{-\alpha_m^2 r^2} (\alpha_m r)^{2k} r \, dr$$

$$= 2\pi \sum_{k=0}^{m} \frac{1}{k! 2\alpha_m^2} \int_0^\infty e^{-t} t^k dt$$

$$= \frac{A}{m+1} \sum_{k=0}^{m} \frac{1}{k!} \cdot k!$$

$$= A.$$

To show (iii) and (iv) we let $t = \frac{\|z\|^2}{R^2}$ and write $g_m(z) = e^{-(m+1)t} \sum_{k=0}^{m} \frac{[(m+1)t]^k}{k!}$.

For the case $t < 1$ we note that

$$g_m(z) = 1 - e^{-(m+1)t} \sum_{k=m+1}^{\infty} \frac{[(m+1)t]^k}{k!},$$

and that

$$e^{-(m+1)t} \sum_{k=m+1}^{\infty} \frac{[(m+1)t]^k}{k!}$$

$$\leq t^{m+1} e^{-(m+1)t} \sum_{k=m+1}^{\infty} \frac{(m+1)^k}{k!}$$

$$\leq t^{m+1} e^{-(m+1)t} e^{(m+1)}$$

$$= e^{(m+1)(1-t+\log t)}.$$

Since $\log t < t - 1$ for $t \neq 1$, we have proven (iii).

For $t > 1$ we note that

$$g_m(z) \leq t^m e^{-(m+1)t} \sum_{k=0}^{m} \frac{(m+1)^k}{k!}$$

$$\leq e^{-\log t} e^{(m+1)(1-t+\log t)},$$

which establishes (iv).

Finally, (v) follows from

$$\frac{d}{d(\|z\|^2)} g_m(z) = \frac{1}{R^2} e^{-(m+1)t} \left[-m \sum_{k=0}^{m} \frac{[(m+1)t]^k}{k!} - \frac{[(m+1)t]^m}{m!} \right]$$

$$\leq 0.$$

Graphs of the g_m are given in Fig. 1; as can be seen, relatively small values of m provide fairly reasonable damage functions.

We now describe the type of integration that allows (1.3) to be evaluated exactly:

*Elliptical and rectangular "cookie-cutters" (i.e., set-indicator functions for ellipses and rectangles) can also be approximated by functions similar to the g_m; moreover, such functions (and sums of them) can be used to approximate target distributions as well as damage patterns. Although we shall not use these facts, the reader should note that the results given here hold in this more general case as well.

Lemma 2 Let p be a polynomial in $x_1, x_2, \ldots, x_n$, and let q be a negative definite quadratic form in $x_1, x_2, \ldots, x_n$. Then

$$\int_{-\infty}^{\infty} p(x_1,x_2,\ldots,x_n)e^{q(x_1,x_2,\ldots,x_n)}dx_1 = r(x_2,\ldots,x_n)e^{s(x_2,\ldots,x_n)},$$

where r is again a polynomial and s is a negative definite quadratic form.
(We recall that a negative definite quadratic form in $x_1, \ldots, x_n$ is a function $\sum_i \sum_j x_i a_{ij} x_j$ that is strictly negative for all values of $x_1, \ldots, x_n$.)

Proof: Denote the integral in question by I. We may write $q(x_1,\ldots,x_n) = ax_1^2 + bx_1 + c$ where $a < 0$ (otherwise q could be made non-negative by a suitable choice of x_1). We then obtain by "completing the squares" that

$$I = \sum_j \hat{p}_j e^{\frac{-b^2}{4a} + c} \int_{-\infty}^{\infty} e^{ax_1^2} x_1^j dx_1,$$

where $\hat{p}_j$ is the coefficient of x_1^j in $p(x_1 - \frac{b}{2a}, x_2, \ldots, x_n)$; we note that each $\hat{p}_j$ is thus a polynomial in $x_2, \ldots, x_n$.

Since $a < 0$ guarantees that each integral in the summation is a finite number, we need only show that $\frac{-b^2}{4a} + c$ is a negative definite quadratic form in $x_2, \ldots, x_n$. It is easily verified that $\frac{-b^2}{4a} + c$ is a quadratic form in $x_2, \ldots, x_n$, so it suffices to prove that it is always negative. Suppose not. Then there exist $\tilde{x}_2, \ldots, \tilde{x}_n$ such that $\frac{-b^2}{4a} + c$ is non-negative; in other words, the equation $ax_1^2 + bx_1 + c = 0$ has a real root, say $\tilde{x}_1$. But this means that $q(\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_n) = 0$, a contradiction. Hence $\frac{-b^2}{4a} + c$ must always be negative.

We are now in a position to express (1.3) in closed form:

Theorem 1 Fix $m \geq 0$. If g_m is as defined in Definition 1, and if h, $f_1, \ldots, f_n$ are normal density functions, then

$$P_D = \int h(x) \left[\prod_{i=1}^{n} \int g_m(x-y) f_i(y) dy \right] dx \qquad (2.1)$$

may be evaluated exactly by successive integrations.

Proof: If we let $v = (v_1,v_2,v_3,v_4) = (x_1,x_2,y_1,y_2)$, we see that $g_m(x-y) = p(v)e^{q(v)}$ and $f_i(y) = c_i e^{r_i(v)}$, where p is a polynomial in the v_j, c_i are constants, and q and the r_i are non-positive definite quadratic forms. But $r_i(v) = 0$ only if $y_1 = y_2 = 0$, and $q(v) = 0$ only if $x_1 = y_1$ and $x_2 = y_2$; hence $q(v) + r_i(v)$ is negative definite.

We may thus apply Lemma 2 twice to integrate each factor in the product in (2.1) with respect to y, thereby obtaining

$$P_D = \int h(x) \left[\prod_{i=1}^{n} p_i(x) e^{q_i(x)} \right] dx \quad .$$

We again may show that the exponent in the integrand is a negative definite quadratic form in (x_1,x_2), so applying Lemma 2 two more times completes our proof.

We note here that P_D will be of the form re^s, where r is a polynomial in the coefficients of the polynomials defining g_m, f_i, and h.

3. The FORMAC Implementation

Although our damage functions g_m allow (1.3) to be evaluated exactly, the necessary computations very quickly become very involved; the reader may verify this for himself by attempting to compute, for example,

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1^m x_2^n e^{q(x_1,x_2)} dx_1 dx_2,$$

when q is a quadratic involving an x_1x_2 term. Application of the method described in the last section therefore requires the use of a symbolic compiler. An attempt was made at RAND several years ago using the ALTRAN compiler, but this attempt failed due to size limitations. This problem of expression "blow-up" was overcome, however, in the FORMAC application we describe:

The first point to note is that the density and damage functions discussed in the last section can each be represented by two polynomials. Thus $g_m(x-y) = p(x-y)e^{q(x-y)}$ may be represented by the polynomials p and q in the variables x_1, x_2, y_1, y_2; similarly the normal $f_i(y) = p_i(y)e^{q_i(y)}$ may be represented by the quadratic polynomial q_i and the constant polynomial p_i.* The coefficients in these polynomials may either be numeric constants or polynomial expressions (in other variables) themselves; for example, in the problem discussed in Section 4, the coefficients in q_i were

*This is also true of the approximations to elliptical and rectangular "cookie-cutters", mentioned in the footnote in Section 2.

polynomials in λ, where λ was a parameter used in specifying the aimpoint of the i-th weapon. These polynomials are easily generated in FORMAC (the quadratics are especially easy to generate by using the EVAL routine).

We next note that multiplying our functions is then just a matter of "minding our p's and q's"; i.e., if f_1 is represented by p_1, q_1 and f_2 by p_2, q_2, then $f_1 f_2$ is represented by $p_1 p_2$, $q_1 + q_2$. These elementary polynomial operations are, of course, trivial in FORMAC (although multiplication does tend to produce large expressions).

From the proof of Theorem 1, it should be clear that the only operations necessary in order to evaluate (1.3) are multiplication and integration; and as we have just indicated, all functions arising in our computations are easily multiplied in FORMAC. Regarding the integration, we first observe that all the integrands which will arise in our computations can be integrated by the same algorithm. That is, all integrands will be of the form pe^q, and hence can be evaluated by an integration subroutine which accepts as input the (FORMAC polynomial) expressions P and Q in the (FORMAC) variable V, and returns as output the expressions R and S such that

$$\int_{-\infty}^{\infty} Pe^{Q}dV = Re^{S}. \qquad (3.1)$$

We therefore programmed a FORMAC routine (INTEGRA) to perform this integration. Since it is natural to integrate with respect to pairs of variables (x_1 and x_2, y_1 and y_2, etc.) etc.), we also programmed a "driver" routine DBINT which performs these double integrations simply by calling INTEGRA twice, once with respect to each variable.

INTEGRA was coded to calculate R and S in (3.1) as follows: let $P(V) = p_0 + p_1 V + \ldots + p_n V^n$ and $Q(V) = aV^2 + bV + c$. Then

$$S = -\frac{b^2}{4a} + c$$

and

$$R = \left(\frac{\pi}{-a}\right)^{\frac{1}{2}} \sum_{\substack{j=0 \\ j \text{ even}}}^{\tilde{n}} \frac{\tilde{p}_j F_j}{(-2a)^{j/2}}, \qquad (3.2)$$

where

$\tilde{n}$ is the largest even integer not exceeding n,

$\tilde{p}_j$ is the coefficient of V^j in $P(V + d)$,

$d = \frac{-b}{2a}$,

and

$$F_j = \begin{cases} (j-1)(j-3)\cdot\ldots\cdot 1 & (j=2,4,\ldots), \\ 1 & (j=0). \end{cases}$$

These formulas are easily verified by "completing the square" in Q and evaluating the integrals $I_j = \int_{-\infty}^{\infty} e^{aV^2} V^j dV$, as indicated in the proof of Lemma 2. For odd j, $I_j = 0$; for even j, I_j is evaluated recursively by integration by parts and equals $\left(\frac{\pi}{-a}\right)^{\frac{1}{2}} \frac{F_j}{(-2a)^{j/2}}$.

As a first attempt, INTEGRA was programmed to compute the coefficients $\tilde{p}_j$ by using the FORMAC routine EVAL to substitute V + d for V in P. This program worked fine when d was a numeric constant but rapidly ran out of core when d was a polynomial. Unfortunately, this will generally be the case. For example, d will always be a polynomial when computing $\int g_m(x-y)f(x)dx$: for when first integrating with respect to x_1, the coefficient of x_1 in the exponent Q will be a linear function of y_1 (and also of x_2, if the principal axes of the variance-covariance matrix of the normal distribution described by f are not parallel to the x_1 and x_2 axes). Also, after integrating out x_1, the coefficient of x_2 in the resulting exponential will still be a linear function of y_1. Moreover, if any of the parameters of the distribution are variable (for example the aimpoint, or mean of the distribution specified by f), then these will be additional variables in d, and will still be present in the exponential even during the final two integrations of (1.3).

The $\tilde{p}_j$ were therefore expressed in terms of the p_j and d by means of the binomial expansion, so that INTEGRA was left with computing S as in (3.2), and

$$R = \left(\frac{\pi}{-a}\right)^{\frac{1}{2}} \sum_{\substack{j=0 \\ j \text{ even}}}^{\tilde{n}} \frac{F_j}{(-2a)^{j/2}} \sum_{k=j}^{n} p_k d^{k-j} \binom{k}{j} \qquad (3.3)$$

when $d \neq 0$. Variables in d produced no problems with regard to size when this formula was used.

All that was then necessary in order to solve a problem was to represent the desired functions by generating the appropriate polynomials, and then to integrate combinations of these functions by repeatedly calling DBINT.

4. An Example

The method we have described was used to solve a targeting problem, which we now discuss.

This example illustrates how the analytical results of Section 2 may be applied using the FORMAC methods discussed in Section 3.

We consider the following problem: We are given a point target with an elliptical normal distribution with density function h about the origin, such that the major and minor axes of its distribution are parallel to the x_1- and x_2-axes respectively. We attempt to destroy it with two weapons, fired independently, each with the same circular "cookie-cutter" damage function k_R. The point of detonation W_i of each weapon is also elliptically normally distributed, and we assume furthermore that both W_i have the same variance-covariance matrix, with axes parallel to the x_1 and x_2 axes. The aimpoint of the first weapon (i.e., the mean of W_1) is the point $(\lambda,0)$, and the mean of W_2 is the point $(-\lambda,0)$ (See Fig. 2). All the parameters except λ of the distributions are given constants; our objective is to choose that value of λ (which we shall denote by λ^*) which will maximize the probability $P_D(\lambda)$ that the target is destroyed. We denote the density function of W_1 by f_λ, that of W_2 by $f_{-\lambda}$.

Intuitively we would expect λ^* to be positive and $P_D(\lambda)$ to be first monotone increasing in $[0,\lambda^*)$, and then monotone decreasing in $[\lambda^*,\infty)$. For as λ increases from 0, each weapon becomes less effective because it will tend to detonate at a point further away from the area where the target is most likely to be found. On the other hand, it is clear that the larger λ, the less the "lethal circles" of the two weapons will tend to overlap, and hence the area covered by their circles will tend to be greater. We would therefore expect, due to this second factor, that a small increase from 0 of λ would result in a higher probability of destroying the target. But too great an increase would result, because of the first factor, in a lower probability.

This problem was solved in the following manner: For a given m we 1) generated the damage function g_m as described in Section 2, 2) used g_m to compute the probability the target would be destroyed, $P_m(\lambda)$, by (1.2), and 3) obtained the λ_m which maximized P_m by setting $\frac{d}{d\lambda} P_m = 0$. This procedure was done for $m = 1, 2, \ldots$, until the λ_m converged.

Two simplifications immediately arose from the symmetry of the problem. First, P (target destroyed by weapon 1) = P(target destroyed by weapon 2). Second, if we let $\Phi_i((x_1,x_2)) = \Phi_i(x) = P(\text{target is destroyed by weapon } i \mid T = x)$, then $\Phi_1((x_1,x_2)) = \Phi_2((-x_1,x_2))$. Determining P_m was therefore reduced to the following set of computations:

$$\Phi_1(x) = \int f_\lambda(y) g_m(x-y)\,dy,$$

$$\Phi_2((x_1,x_2)) = \Phi_1((-x_1,x_2)), \qquad (4.1)$$

$$P_{one} = \int h(x)\Phi_1(x)\,dx,$$

$$P_{both} = \int h(x)\Phi_1(x)\Phi_2(x)\,dx,$$

and $$P_m = 2P_{one} - P_{both}.$$

The second computation was done with the FORMAC routine EVAL, so that DBINT only needed to be used three times. Computational simplifications similar to these can generally be expected to arise from the nature of the specific problem being considered.

The computations (4.1) were done for m = 1, 2, 3 and 4. The general form of P_m was

$$P_m(\lambda) = \left[2P_{one}(\lambda)\right] - \left[P_{both}(\lambda)\right]$$

$$= \left[e^{-\alpha_m\lambda^2} \sum_{j=0}^{2m} \beta_{mj}\lambda^{2j}\right] - \left[e^{-\gamma_m\lambda^2} \sum_{k=0}^{4m} \delta_{mk}\lambda^{2k}\right].$$

Values of $P_m(\lambda)$ for $\lambda = 0, .1, .2, \ldots, 1.0$ are given in Table 1. Table 2 has the optimizing λ_m and the corresponding maximum probabilities $P_m(\lambda_m)$.

We note several facts about each P_m. First, P_m has the shape (increasing in $[0,\lambda_m)$, decreasing in $[\lambda_m,\infty)$) we expected, confirming our intuition. Second, P_m is extremely flat in the interval $[0,\lambda_m)$. An attempt to produce a steeper curve by "stretching" the target (i.e., increasing the variance of its distribution in the x_1 direction), although increasing P_m to around .4 in $[0,\lambda_m)$, still resulted in a flat curve there. The flatness of P_m thus may stem more from the nature of the problem than the parameters of the particular distributions involved.

Table 1

VALUES OF $P_m(\lambda)$

λ	$P_1(\lambda)$	$P_2(\lambda)$	$P_3(\lambda)$	$P_4(\lambda)$
0.0	.2889	.2904	.2910	.2913
.1	.2901	.2919	.2927	.2932
.2	.2929	.2956	.2969	.2977
.3	.2958	.2994	.3012	.3024
.4	.2968	.3011	.3033	.3048
.5	.2941	.2988	.3013	.3028
.6	.2870	.2915	.2940	.2955
.7	.2756	.2796	.2817	.2830
.8	.2608	.2640	.2657	.2666
.9	.2437	.2462	.2474	.2482
1.0	.2253	.2272	.2281	.2286

Table 2

OPTIMIZING PARAMETERS λ_m AND CORRESPONDING OPTIMUM PROBABILITIES $P_m(\lambda_m)$

m	λ_m	$P_m(\lambda_m)$
1	.3813	.2968
2	.3983	.3011
3	.4063	.3034
4	.4109	.3048

This flatness is somewhat disappointing since it shows that spreading the weapons out a bit produces no dramatic increase in the probability of destroying the target. On the other hand, such information could be useful. For example, if the two weapons were missiles, the knowledge that they could just as effectively be both aimed at the same point might simplify guidance considerations.

If we examine Table 1, we see that the values of P_m at each λ do not change much with respect to m; in fact, the difference between the maximum probability for m = 1 and m = 4 is less than .008. Thus, although g_1 certainly is not shaped much like k_R (see Fig. 1), it still may be used to compute a fairly good approximation to P_D. This is because the difference between g_1 and k_R is "smoothed out" when the functions are multiplied by probability density functions and then integrated.

Regarding the optimal aimpoint λ^*, we see that the λ_m converge to λ^* fairly rapidly; in fact, the "smoothing out" process is sufficiently strong in this case that even λ_1 provides a good approximation to λ^*. The fact that λ_m increases in m is to be expected. The g_m are becoming more like k_R and hence tend to overlap more, so that it is desirable to increase the distance between the aimpoints of the two weapons.

In Table 3 we have presented figures showing the number of cpu seconds and the amount of core storage required to do the calculations (4.1) for m = 1, 2, 3 and 4. Also presented, for purposes of comparison, are the time and space required for the "fixed aimpoint case", i.e., for the case in which λ was set at the beginning of the program to a numeric constant (.5), instead of being carried throughout the calculations as a FORMAC atomic variable. Table 4 shows the same data for m = 2, 3, 4 with the time and core necessary for the case m = 1 subtracted; this gives a rough measure of the requirements to process the problems, less "overhead". These figures were obtained on an IBM System 360/65.

We first note that "expression swell", the generation of large expressions in the intermediate steps of the computations, was not a significant problem. From Table 4 we see that the amount of core required increased approximately linearly with m, the rate of increase being slightly faster in the variable than in the fixed aimpoint case. The increase with m in time, however, was much more rapid. In the fixed aimpoint case, the cpu seconds required increased approximately as m^5. The following rather tentative reasoning, although not purporting to be an explanation of this rapid increase, may offer some insight into what is going on. Equation (3.3) shows that much of the computing time will be spent in adding expressions together, and this in turn involves comparing the terms in the expressions. That is, if we wish to add $a_1 + a_2 + \ldots + a_r$ to $b_1 + b_2 + \ldots + b_s$ in such a manner that the terms in the result are combined whenever possible (which we need to do in order to prevent our expressions from becoming too large), then we need to compare each a_i with each b_j to see if they can be combined into one term; this requires rs comparisons. Now if the expressions we are adding have, say, mr and ms terms each, then there will be m^2rs comparisons to be made. Moreover, if the results of two such additions each result (due to a lack of simplification) in m^2rs terms, then again adding these two results together will necessitate $m^4(rs)^2$ comparisons. Since the amount of core required increases linearly with m, it may be safe to assume that the expressions to be added in (3.3) will typically satisfy these conditions, so that the double summation in (3.3) could account for an increase in time on the order of m^4.

Table 3

TIME AND SPACE REQUIREMENTS. TIME IS GIVEN IN SECONDS, STORAGE IN UNITS OF 1024_{10} BYTES

Fixed Aimpoint

m	CPU Time	Core Storage
1	5	154
2	15	160
3	67	168
4	298	186

Variable Aimpoint

m	CPU Time	Core Storage
1	5	154
2	33	170
3	254	204
4	1517	286

Examination of Table 4 also reveals that for each m ≥ 2, the time used in the variable aimpoint case was approximately (m+1) times the time required for the fixed aimpoint case. Equation (3.3) would at first lead one to believe that this is due to the fact that the presence of λ in the expression d will result in d containing more terms when λ is a FORMAC atomic variable than when λ is a constant, and

that this difference will be magnified when d is raised to a power. As it turns out, however, this is not the case: in all the integrations to be performed, the expression d (which is equal to $\frac{-b}{2a}$, where the integrand has the exponential $e^{av^2 + bv + c}$) will have the same number of terms, regardless of whether λ is a numeric constant or an unassigned variable. This ratio of the variable aimpoint time to the fixed aimpoint time must therefore be due to some factor other than the number of terms in d. One possible explanation would be that the comparisons just described take longer when some of the terms are variables than when they are constants.

Table 4

ADJUSTED TIME AND SPACE REQUIREMENTS. TIME IS GIVEN IN SECONDS, STORAGE IN UNITS OF 1024_{10} BYTES

Fixed Aimpoint

m	CPU Time	Core Storage
2	10	6
3	62	14
4	293	32

Variable Aimpoint

m	CPU Time	Core Storage
2	28	16
3	249	50
4	1512	132

Although the time required to compute P_m became prohibitive as m increased, we may be encouraged by two aspects of this particular application of our FORMAC approach: first, the frequently encountered phenomenon of "expression swell" did not prove to be a significant problem. Second, our results for the case m = 1 provided a fairly good approximation to the final answer; and the core storage and cpu time required to obtain these results were minimal. It remains to be seen whether this will be true of other applications as well.

References

1. Tobey, R., et al, "PL/1 FORMAC Symbolic Mathematics Interpreter," IBM Contributed Program Library, 360D-03.3.004, Hawthorne, New York, September 1969.

2. Ury, H. K.,"Approximate Methods for Computing Elliptical Probability Coverages," Civil Defense Research Project, CDRP-182-110, University of California--Berkeley Institute of Engineering Research, November 1962.

3. Guenther, W. C. and P. J. Terragno, "A Review of the Literature on a Class of Coverage Problems," Annals of Mathematical Statistics, No. 1, Vol. 35, 1964 (March), p. 232.

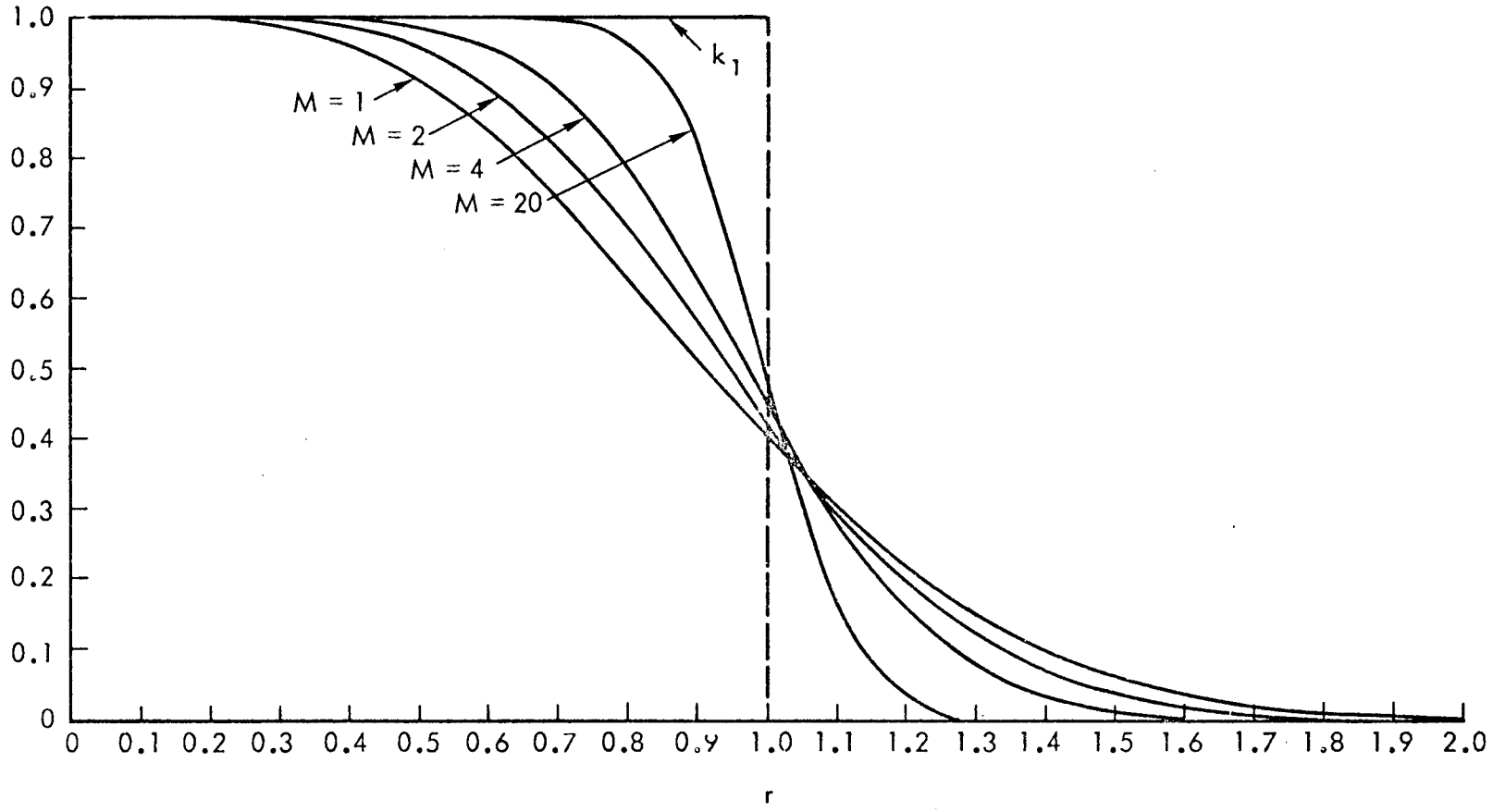

Fig. 1--Approximations g_m to a "cookie-cutter" k_1 of radius 1.

$$g_m(x) = e^{-(m+1)r^2} \sum_{k=0}^{m} \frac{[(m+1)r^2]^k}{k!}, \text{ where } r = \|x\|.$$

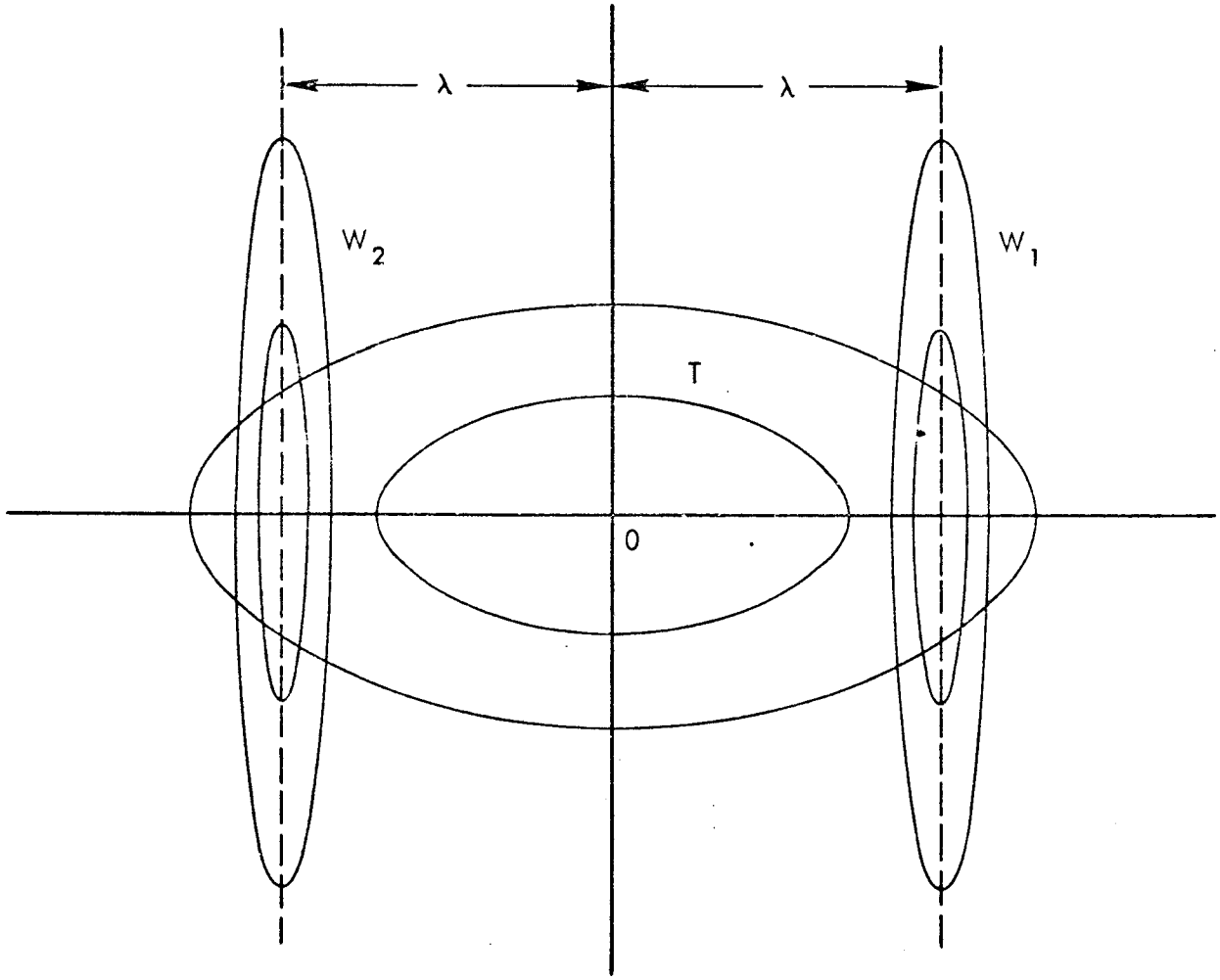

Fig. 2--Contours of equal probability for the random variables T, W_1, and W_2. Each variable is normal with the following parameters:

	T	W_1	W_2
Mean	(0,0)	(λ,0)	(-λ,0)
Variance-- Covariance Matrix	$\begin{bmatrix} 1 & 0 \\ 0 & 1/4 \end{bmatrix}$	$\begin{bmatrix} 1/64 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1/64 & 0 \\ 0 & 1 \end{bmatrix}$