NWC TP 5416

A Ballistic Trajectory Algorithm for Digital Airborne Fire Control

by

Arthur A. Duke
Thomas H. Brown
Kenneth W. Burke
Richard B. Seeley
Weapons Development Department

Naval Weapons Center
CHINA LAKE, CALIFORNIA ■ SEPTEMBER 1972

[redacted]

ABSTRACT

A method of numerical integration of the equations of motion of unguided air-to-surface weapons is developed. This algorithm, suitable for real-time solution by airborne digital computers, yields accurate trajectory parameters, provides great flexibility in release condition and weapon type, and minimizes computer memory requirements.

NWC Technical Publication 5416

Published by	Weapons Development Department
Manuscript	40/MS 72-78
Collation	Cover, 28 leaves, DD Form 1473, abstract cards
First printing	360 unnumbered copies
Security classification	UNCLASSIFIED

Naval Weapons Center
AN ACTIVITY OF THE NAVAL MATERIAL COMMAND
W. J. Moran, RADM, USN .... Commander
H. G. Wilson .... Technical Director

FOREWORD

The investigations leading to this report were motivated by a need for a flexible, general-purpose algorithm, suitable for near real-time solution by current airborne digital fire-control computers, that would provide accurate weapon trajectory parameters for unguided air-to-surface weapons delivery.

The initial investigations were performed by the Boeing Company, Seattle, Wash., under Naval Weapons Center Contracts N00123-69-C-1344 and N00123-70-C-0829, authorized and funded by AirTask A365-33348/216-1/S-171-00-00, from March 1969 to September 1970. The final work was accomplished by NWC Code 4074 under AirTask A510-5103-216-2/1235-000-143.

This report has been reviewed for technical accuracy by Paul B. Homer, Weapons Development Department, and Edward Y. Mikami, Research Department. It is released at the working level for information only.

Released by
M. M. ROGERS, *Head*
Weapons Systems Analysis Div.
15 August 1972

Under authority of
F. H. KNEMEYER, *Head*
Weapons Development Dept.

CONTENTS

ACKNOWLEDGMENT

The authors would like to acknowledge the assistance of Jack R. Basden, Richard Nuckles, and Allen L. Russell who contributed significantly to the preparation of this report.

INTRODUCTION

Since the first airborne digital computers were introduced into operational Navy attack aircraft some 15 years ago, a basic computational requirement imposed upon these computers has been the prediction of weapon range and time-of-fall, given sensor-supplied release conditions. Because of the nature of the problem, each aircraft system developed in this interim period (e.g., A-6A, ILAAS, A-7E, etc.) has had concurrently developed with it a "new" set of weapon ballistic equations for this air-to-ground fire control use. While all the "sets" of weapon ballistic equations are developed from the same basic physical considerations, their final formulation is subject only to the ingenuity of the mathematician, and often the equation sets bear no resemblance to each other. Each of these sets of equations has to be "proofed" through extensive developmental flight tests, and each has its own idiosyncrasies.

As the processing speed and capacity of these airborne digital computers has increased, a direct method of solution of the ballistic weapon trajectories has become more and more attractive. This method, numerical integration, has heretofore required excessive time at the processing speeds available to meet the near real-time requirements of airborne weapon delivery systems.

This report presents a method of numerical integration of ballistic weapon trajectories that is suitable for current operational airborne digital computers in that it is fast, accurate, flexible, and efficient. It is hoped that this algorithm, or modifications thereof, will become the standard method of trajectory computation for future airborne digital weapon delivery systems.

BACKGROUND

BALLISTIC PROJECTILES

Most aircraft unguided air-to-ground weapons can be described as ballistic projectiles. That is, the only forces acting on them after release from the aircraft are gravity and aerodynamic drag. Bullets, streamlined bombs, drogued (retarded) bombs, cluster munitions, and unguided rockets (after burnout) are all ballistic projectiles. Guided weapons and weapons developing lift are not ballistic projectiles.

To successfully release a ballistic weapon from an aircraft so that it impacts at a desired point requires a measurement of--or predictions concerning--the following quantities:

1. Position of the target relative to the aircraft
2. Velocity of the aircraft in the air mass
3. Direction and magnitude of gravity
4. Velocity of the aircraft relative to the ground } (or total rela-
5. Velocity of the target relative to the ground } tive velocity)
6. An *a priori* prediction of the weapon trajectory (for example, horizontal range, time of flight)

given known initial (release) conditions in the air mass. This includes assumptions concerning the structure of the air mass containing the trajectory and of the ballistic (drag) characteristics of the weapon. The first five of these quantities are normally supplied in current operational airborne weapon delivery systems by a variety of aircraft sensors, e.g., inertial platforms, air data sensors, radar or laser rangers, target-tracking devices, etc. While the accuracy with which a ballistic weapon can be delivered against a target depends greatly upon the accuracy of this sensor-supplied information, equally important to the problem is the *a priori* prediction of the weapon's trajectory based upon the sensor-supplied instantaneous weapon release parameters.

BALLISTIC PROJECTILE TRAJECTORY SOLUTION

The solution of a ballistic trajectory in the air mass has frustrated mathematicians for several centuries. The equations of motion governing the trajectory of a ballistic projectile are a simple-appearing set of second-order differential equations (see p. 8). However, when a reasonably accurate model of the aerodynamic drag caused by the projectile's motion through the air mass is included in these equations, they are rendered extremely nonlinear, and no general solution in closed form has been found that is satisfactory for solution by an airborne fire-control computer.

In the past, several techniques have been used to evolve ballistic trajectory equations (solutions) that were adequate for airborne fire-control use. Most of these techniques involved replacing the accurate model of projectile drag with simplifying approximate expressions that render the equations integrable in closed form. These approximate equations, of course, yield solutions that are incorrect under release conditions (or projectile drag regimes) for which the original approximation was not nearly true. To correct these first solutions, empirical functions are determined that modify the approximate solutions to yield sufficiently accurate results. As the weapon release envelopes have expanded and the number of types of ballistic weapons has increased, these empirical functions have become more and more complex. The introduction of the dispenser weapons, with their discontinuous drag change and resultant "broken-back" trajectory, has further compounded the problem.

Normally, these approximate solutions and their modifying empirical functions are developed to satisfy the stated release envelopes and solution accuracy of a predefined set of ballistic weapons. If a new weapon is introduced, a modification to an existing weapon occurs, or a change in release envelope is required, it is not unlikely that the original empirical function will have to be redetermined to satisfy the new requirement.

All these factors have led to considerable complexity in the ballistic trajectory equations resident in most current aircraft weapon delivery systems that use a digital computer for real-time data processing. This complexity, of course, leads to large resident program memory requirements, while the basic formulation often restricts the weapon release envelope over which accurate solutions are possible. Additionally, the equation formulation often does not provide sufficient flexibility to allow easy incorporation of desired changes in weapon type or release envelope limits.

Since the introduction of the large ground-based digital data processors, all bombing and ballistic tables have been calculated by a process known as "numerical integration." In this process, the total path is divided into many pieces, and within each piece the approximation is made that all forces remain constant for short time periods. A mathematical formula with constant forces permits easy calculation of the position and velocity of the projectile at the end of the short time period, e.g., 0.1 second. The forces on the projectile are then calculated for the new point and used for the next 0.1 second, and the process is continued in a repetitive manner. A typical calculation may divide the fall (trajectory) of a weapon into 1,000 segments. While yielding very accurate solutions, the time required to complete the computations may range up to a few seconds even on a high-speed, ground-based digital data processor. This numerical integration process is relatively straightforward, and requires only that the following be known: an accurate mathematical representation of the atmosphere containing the trajectory, the drag characteristics of the weapon, and the release (initial) conditions.

The intrinsic simplicity, accuracy, and flexibility of this method of computing ballistic trajectories made it an extremely attractive candidate for use in airborne digital computers for trajectory computation. However, a major obstacle to overcome was the near real-time solution requirement for airborne applications; i.e., given that present release conditions were measured at time zero, a trajectory solution must be available within the computer based on those measurements on the order of 0.05 to 0.1 second later, if delivery accuracy is to be retained under dynamic flight conditions. Even though the processing speeds of airborne digital computers have been ever-increasing, the numerical integration methods commonly used by ground-based digital processors require far more time to provide a solution than this maximum acceptable time.

The ballistic integrator algorithm discussed in detail in the remainder of this report is designed to minimize these computation time requirements, while still retaining excellent accuracy and great flexibility. It also has been designed to minimize computer program storage requirements.

BASIC CALCULATION METHOD

The algorithm uses a time-base, second-order, Runge-Kutta numerical integration process with a fixed number of time (integration) steps for all weapons. To keep calculation time small and fixed (a requirement for airborne use) the number of integration steps must be small and fixed. Good accuracy is maintained for all weapons using 10 integration steps. To provide greater accuracy, or to expand the release envelope, more steps could be used at the expense of increased computation time.

In the initial work on this algorithm, the equations or data used in the description of the atmosphere (i.e., density, speed of sound, etc.), gravity acceleration, and drag coefficient were those used as standard in the computation of official ballistic tables. Subsequent work has developed new equations that yield the same results, but which minimize both storage and computation time. At no point in the basic equations (the models of the physical world) has any approximation been used that is significantly different from those accepted and used in the computation of ballistic tables.

Figure 1 illustrates how the numerical integration process of the algorithm works. Figure 1A shows a defined release point (speed, altitude, and dive angle). Everything necessary is known about the release point. The impact range and time-of-flight are to be calculated.

On first-pass (first calculation of weapon trajectory) the size of calculation intervals must be estimated. One way of obtaining this estimate is to use the vacuum trajectory in finding the time-of-flight and dividing it by the number of integration steps to be used. This works well for low-drag bombs; however, this estimate may be off as much as a factor of two or three in the high-drag bomb case. Even after making a vacuum guess, in order for the integration technique to be stable the calculation intervals must not be larger than the maximum allowable. It turns out that it is best to use maximum step size allowable to start the first-pass. Estimating the time-of-flight after first-pass will not be a problem. Since the time-of-flight will be calculated repetitively, one will always have a good estimate once the process is started.

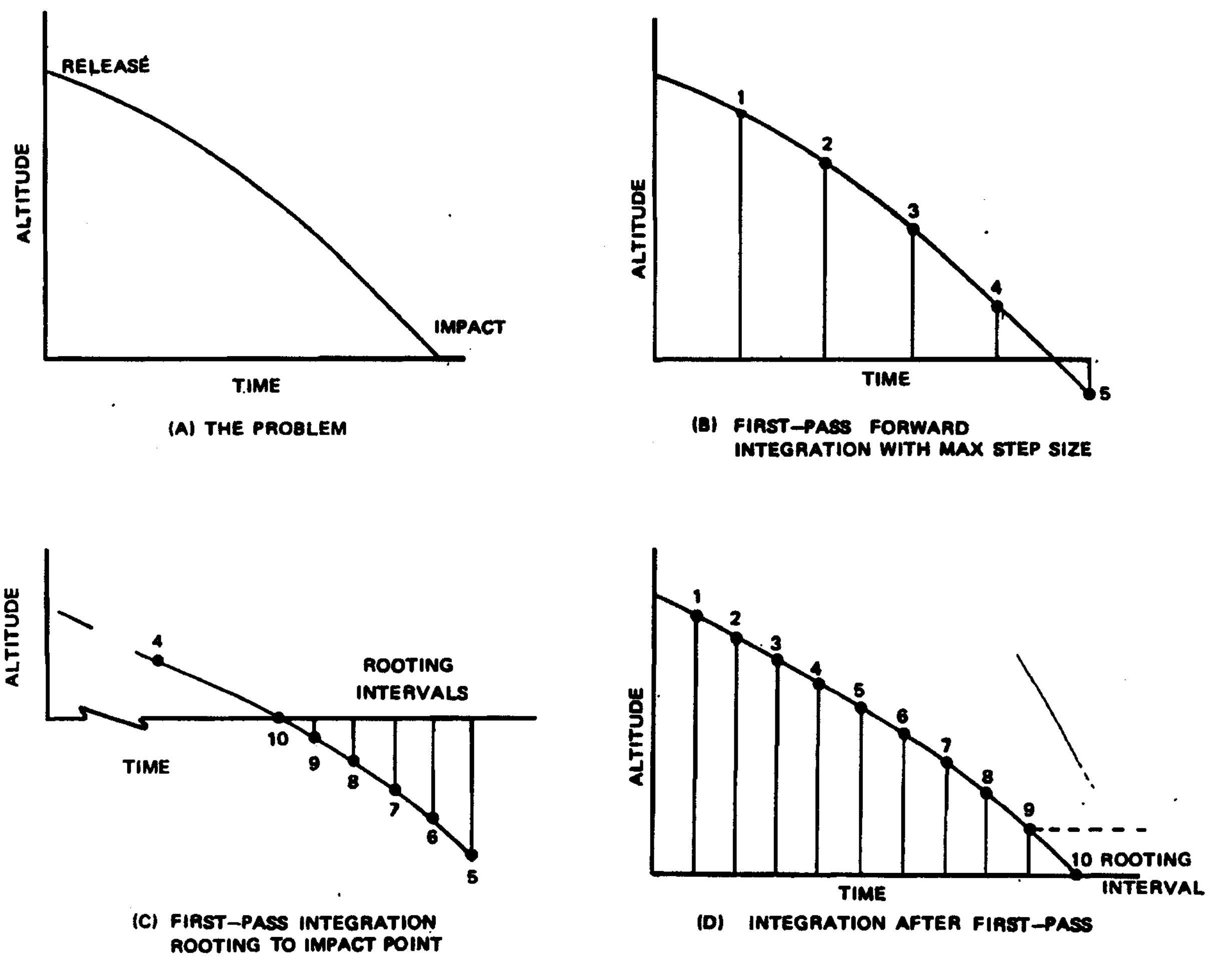
ALTITUDE
RELEASE
IMPACT
TIME
(A) THE PROBLEM
ALTITUDE
1
2
3
4
5
TIME
(B) FIRST-PASS FORWARD INTEGRATION WITH MAX STEP SIZE
ALTITUDE
4
ROOTING INTERVALS
10
9
8
7
6
5
TIME
(C) FIRST-PASS INTEGRATION ROOTING TO IMPACT POINT
ALTITUDE
1
2
3
4
5
6
7
8
9
10 ROOTING INTERVAL
TIME
(D) INTEGRATION AFTER FIRST-PASS

FIG. 1. Numerical Integration.

Starting at release point, the position, velocity, etc., of the projectile is calculated as it reaches the time corresponding to the end of the first step. (See Fig. 1B, point "1".) No trajectory calculations are made at any point between the ends of the steps. To calculate "1", begin with the release point and calculate the average value of the forces acting on the projectile between release and "1". The simplest estimate would be that the forces are the same throughout the interval as at release. This estimate is good only if the interval is very short or if the forces change slowly. A better estimate is needed to use long, but few, steps. The one used in the algorithm is the second-order, Runge-Kutta technique of numerical integration. This technique and why its choice are discussed in Appendix A (Ref. 1 and 2).

The average values of the forces in the interval, expressed as functions of time, are multiplied by the length of the step to find a close approximation of "1". The same practice is repeated to go from "1" to "2" and so on until "5" is reached.

At "5" the algorithm notes that the calculated position is underground (below target altitude). It is not known where between "4" and "5" the projectile struck the ground. Since we are using 10 integration steps total, the remaining 5 steps are used for rooting the projectile to impact point (see Fig. 1C). The size of integration interval for "6" is obtained by dividing the altitude at "5" by the product of the vertical velocity component of the projectile at "5" and the remaining five integration steps. The process is repeated for "7" and so on until we reach "10". Finally, a small straight-line correction is made, if needed, for time-of-flight and impact range so the projectile will be at target altitude.

Figure 1D shows the numerical integration process after first-pass of the same release condition.

The basic integration method applies to any ballistic projectile, but some adaption is required for each type of weapon. The adaptions included in the algorithm are described briefly below.

CLASSES OF WEAPONS

Streamlined Bombs and Bullets

The process described fits streamlined bombs exactly. After being fired, a bullet is really just a small, streamlined bomb and is handled as such. The muzzle velocity of a particular gun-bullet combination is added to the aircraft velocity.

Drogued Bombs

These bombs are more difficult to model accurately. Their common characteristic is that they are released in a relatively low-drag configuration. At some time after release they deploy vanes or a parachute which greatly increases total drag. There is no basic difficulty in calculating bomb trajectory before the drogue is deployed. Nor is there a basic difficulty in calculating the trajectory after the drogue is deployed. The problem is to break the calculation correctly into two parts, one using each drag function.

Cluster Bombs

These weapons are released as fairly large, low-drag containers. At a predetermined point in their fall, they use some mechanism to dispense smaller weapons. These smaller weapons are higher drag than the container and may also develop lift. The fire-control computer should predict the impact location of the center of the pattern rather than individual positions of the small bombs. The method of applying the algorithm is to calculate the trajectory of the container to the point where it dispenses the small weapons; switch to a drag function which describes the motion of the pattern center; and integrate down to the target altitude. Once again, the important factor is providing the logic to switch the drag function at the right place.

Unguided Rockets

Rockets present several problems. They change weight during flight; they have thrust (at varying levels) as well as drag; and they slew around just after firing, because their launches are not lined up with the aircraft direction of flight. The algorithm treats unguided rockets like a cluster bomb with the first stage having an average thrust as well as an average thrust time. This provides sufficient accuracy and saves computer storage that would be taken up storing the thrust profile of each rocket.

NWC–BOEING TRAJECTORY ALGORITHM

Basic Differential Equations

The development of an effective weapons release system is inherently dependent upon obtaining solutions of the equations for the motion of a projectile within the atmosphere. This is, generally, a difficult mathematical problem which has not been solved completely. The major difficulty stems from the nonlinearities introduced by the atmospheric effects on a falling weapon.

In choosing the mathematical model, two considerations have been kept in mind. The major objective of the mathematical analysis is to yield the weapon impact point. The main effect of this is that the weapon mass can be assumed to be a point mass. Also, the choice of the model is dictated by the need to evaluate results against some standard. Since the armed forces published range tables for various weapons, the model is chosen to conform as closely as possible to the model used for these tables.

The equations of motion are developed assuming the projectile is a point mass acted on only by the force of gravity and the retardation forces due to air resistance. The trajectory can be restricted to a plane by ignoring crosstrack effects such as winds. For practical applications, the effect of winds can be accounted for in a straight-forward manner.

The assumptions adopted are summarized below:

1. The Earth is flat and nonrotating.
2. The gravitational attraction is constant.
3. The projectile is a point mass.
4. The projectile is not powered and has a constant mass. (Rocket applications are discussed in the next section.)

Under these assumptions, the differential equations of motion (Ref. 3) have the following form.

$$\frac{d^2X}{dt^2} + H\frac{dX}{dt} = 0 \qquad (1)$$

$$\frac{d^2Y}{dt^2} + H\frac{dY}{dt} + G = 0$$

where X, Y, t, and G denote downrange, altitude, time, and gravitational attraction, respectively. The coefficient H, which is the drag function, is given by

$$H = \frac{\rho}{W} d^2 \frac{\pi}{8} C_D V \qquad (2)$$

where ρ is the atmospheric mass density, W is the bomb mass, d is the bomb diameter, C_D is the weapon coefficient of drag, and V is the velocity in air mass. The atmospheric density ρ is given as a function of altitude which is fitted to measured values of atmospheric density. C_D is empirically derived and given in tabular form as a function of Mach number. A constant gravitational acceleration of 32.174 ft/sec^2 is quite adequate for most bombing applications and helps simplify the differential equations.

The above differential equations are not analytically integrable, if an accurate model of H is used, because an accurate model would render them extremely nonlinear.

The two second-order, differential equations given in Eq. 1 will now be rewritten as four first-order, differential equations. This is done to get the differential equations in a form that is more suitable to the integration process used. Two new variables V_x and V_y are defined by

$$\frac{dX}{dt} = V_x \qquad \frac{dY}{dt} = V_y \tag{3}$$

Substituting the above expressions in Eq. 1 results in

$$\frac{dV_x}{dt} = -H\,V_x \qquad \frac{dV_y}{dt} = -H\,V_y - G \tag{4}$$

The four first-order, differential Eq. 3 and 4, are the desired equations with time as the independent variable. The Runge-Kutta integration formulas provide a step-by-step method of finding dependent variable values at given intervals of the independent variable. This is discussed in detail in Appendix A. Figure 2 shows the salient features in the x-y plane.

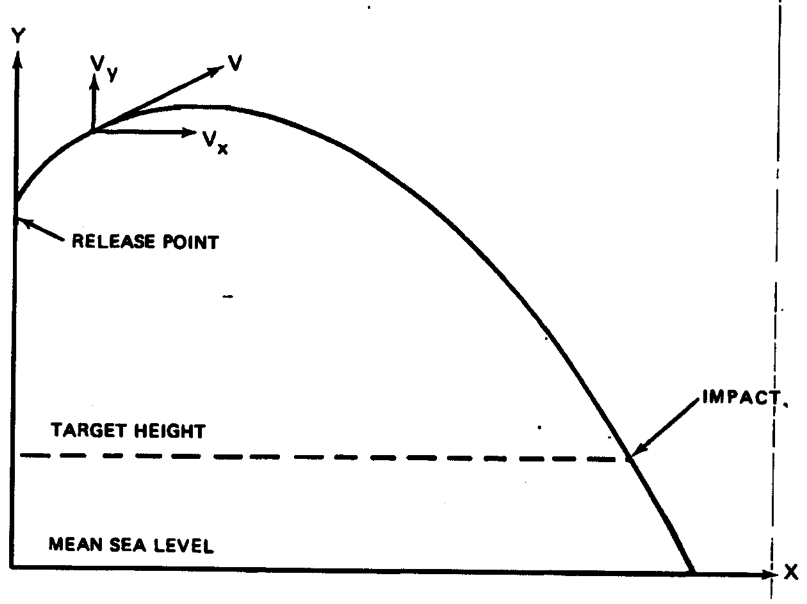

FIG. 2. Trajectory in the x-y Plane.

LOGIC AND DESCRIPTIVE FLOW DIAGRAMS

The computer logic developed for implementing the weapon delivery algorithm into an airborne digital computer is outlined in Fig. 3. This includes the logic for the:

1. Starting procedure
2. Repetitive computations
3. Specification of the integration interval in each of the above cases
4. Monitoring of the state of the trajectory computations, i.e., whether the computation of a given trajectory or part of a trajectory is completed.

Figure 4 is a descriptive flow diagram of Fig. 3. This will aid the reader in understanding the details of the algorithm.

The scheme used to classify weapons into different types will become apparent in the section starting on p. 13. The ITYPE number for each weapon is carried along with the weapon-dependent constants. This number controls the course of the logic shown in Fig. 3. The values for MSTG and IREG help specify the form and region of the C_D representation to be used in the integration process.

To recapitulate, the logic shown in Fig. 3 specifies the integration interval size, D, and the form of C_D for each integration step and performs the integration in computing the weapon trajectory. A more detailed explanation of the logic flow diagram in Fig. 3 is included in the section referred to in the preceding paragraph.

Definition of Program Terms

Terms relating to this program are defined as follows:

A	Runge-Kutta parameter (0.7 for 10 integration steps)
AA	0.5/A
AD	A × D
AN1, AN2	Runge-Kutta variables
AP1, AP2	Runge-Kutta variables
A1, B1, C1	RHO coefficients
A2, B2	CM coefficients
CC	Matrix (3x3x2) of drag curve coefficients
CF	CKDG stretch
CFORM1	CKDG stretch factor for first stage
CFORM2	CKDG stretch factor for second stage
CKDG	Bomb coefficient times drag coefficient
CM	Mach number
D	Integration step size
DKG	Shift in CKDG
DKG1	Shift in CKDG for first stage
DKG2	Shift in CKDG for second stage
DM	Shift in Mach number
DMAX	Largest integration step size allowable
DM1	Shift in Mach number for first stage
DM2	Shift in Mach number for second stage
DS	Integration step size for first stage of ITYPE ≠ -1 weapons except for retarded Snakeye. DS is a coefficient in the fin deployment time calculation for retarded Snakeye weapons.
FN	Equal to TH for first stage
FRACT	Parameter between zero and one (0.5 for 10 integration steps)
G	Acceleration due to gravity
HH	Total drag function
I	Index counter for Runge-Kutta integration

ICALC	Initial trajectory calculation indicator
IREG	Mach region index
ITYPE	Weapon classification parameter
KFLAG	Flag for weapon delivery envelope. KFLAG equals zero for inside the envelope and plus outside. On first-pass KFLAG is meaningless.

MSTG	Weapon stage index
	MSTG = 1 for first stage
	MSTG = 2 for second stage

NL	Maximum number of integration steps
NLM1	NL - 1
NSTEP	Number of integration steps for first stage of ITYPE $\neq -1$ weapons. NSTEP = NLM1 for ITYPE = -1 weapons.
NT	Number of deceleration steps for ITYPE $\neq -1$ weapons
NUM	Integration step counter
RHO	Air density (slugs/ft^3)
SL	Slope coefficient for retarded Snakeye fin deployment time; SL = 0 for other weapons
T	Elapsed time from weapon release
TH	See Eq. 13
TL	Small correction in T after final integration step
TS	Time-of-flight of the previous trajectory calculation
U	Aircraft speed (ft/sec)
V	Total velocity of weapon (ft/sec)
VX	Velocity component of V in X direction
VXA	VX at weapon release
VXO	VX at start of present integration step
VY	Velocity component of V in Y direction
VYA	VY at weapon release
VYO	VY at start of present integration step
X	Weapon groundrange from release (ft)
XNSTEP	NSTEP + FRACT
XNLM1	NLM1 + FRACT
Y	Weapon altitude above sea level (ft)
YA	Highest altitude in the weapon trajectory (ft)
YO	Y at start of present integration step
YT	Target altitude above sea level (ft)

EXPLANATION OF LOGIC FLOW DIAGRAM

The logic flow diagram in Fig. 3 is best explained by an example of each of the four types of weapons and with the aid of Fig. 4. Given a weapon, its type, ITYPE, is determined as follows:

ITYPE	Type of weapon
-1	Single-stage weapons with no deceleration computation (e.g., Mk 81 low-drag, Mk 117)
0	Retarded parachute weapons (e.g., retarded Mk 43)
1	Dual-stage weapons which require two drag curves (e.g., Rockeye II, Sadeye, Retarded Snakeye)
2	Single-stage weapons with the deceleration computation (e.g., Mk 106) and dual-stage weapons which require only one drag curve (e.g., rockets)

Computations for an ITYPE -1 Weapon

The logic flow starts in the top left corner of Fig. 3 from DECODE (see Appendix C). If the computations are being done for the first time (first-pass), the weapon-dependent constants and certain other variables are set up. Otherwise, this task is bypassed in DECODE.

The position and velocity variables are initialized to correspond to the beginning of a trajectory and several other variables are reset in block RK1. The type of weapon, ITYPE, is tested in block RK2. This sends the logic to block RK3, where the integration step size, D, is calculated. Note that TS is not known if the calculations are being done for the very first time; however, in this case, D is calculated later in block RK7.

The value of ICALC is tested next. If first-pass, ICALC will equal zero and the logic will be routed to block RK7, where D is set equal to the maximum allowable step size, DMAX. If not first-pass, ICALC will be plus and the logic will go to block RK6, where a test is made to make sure that D is not larger than DMAX. If D is greater than DMAX, the logic will be routed to block RK7.

We are now ready to perform the first step of Runge-Kutta integration of the trajectory in blocks RK8-RK18. These blocks are explained in the descriptive flow diagram of Fig. 4.

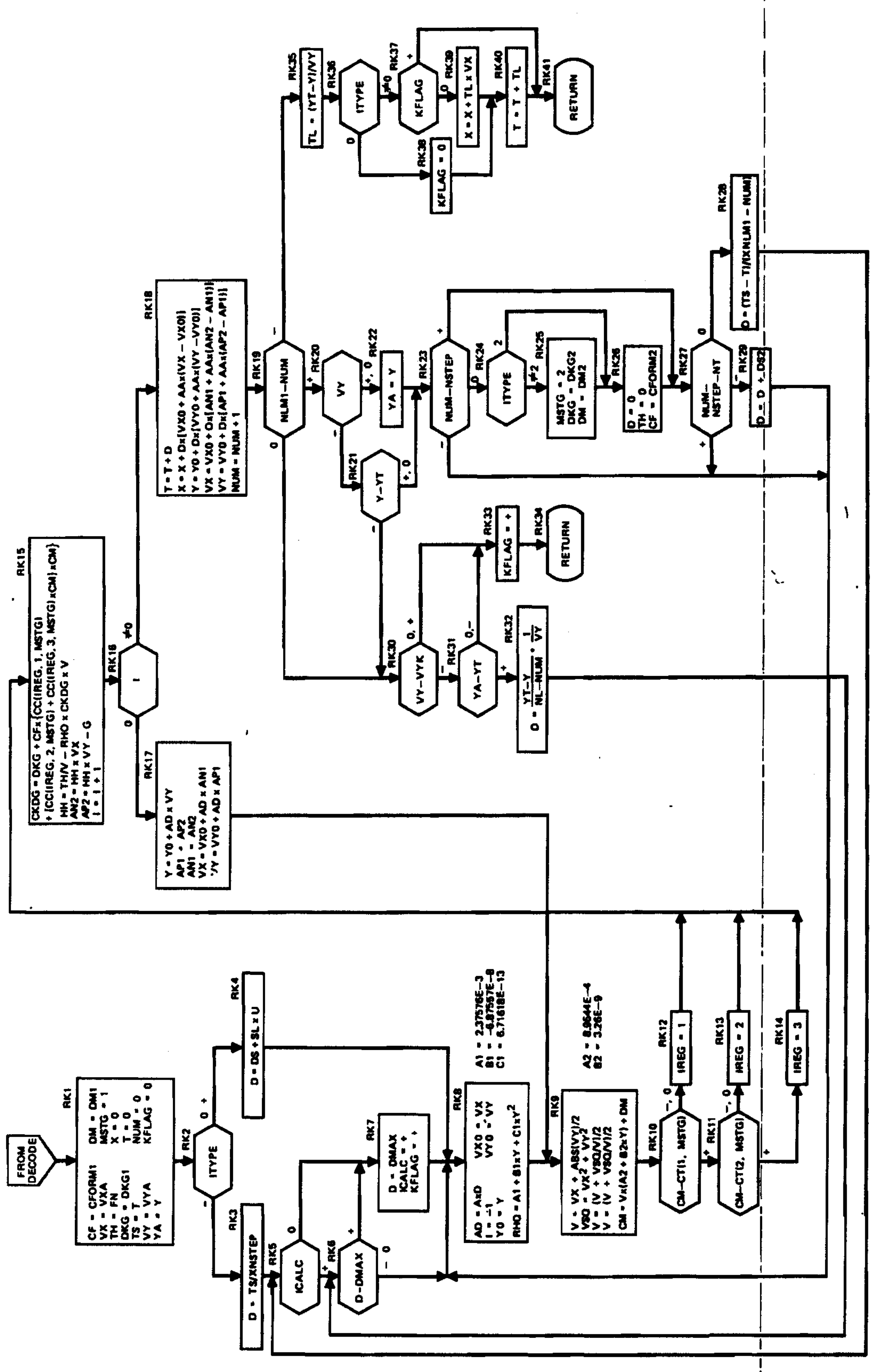

FIG. 3. Logic Flow Diagram of the Algorithm.

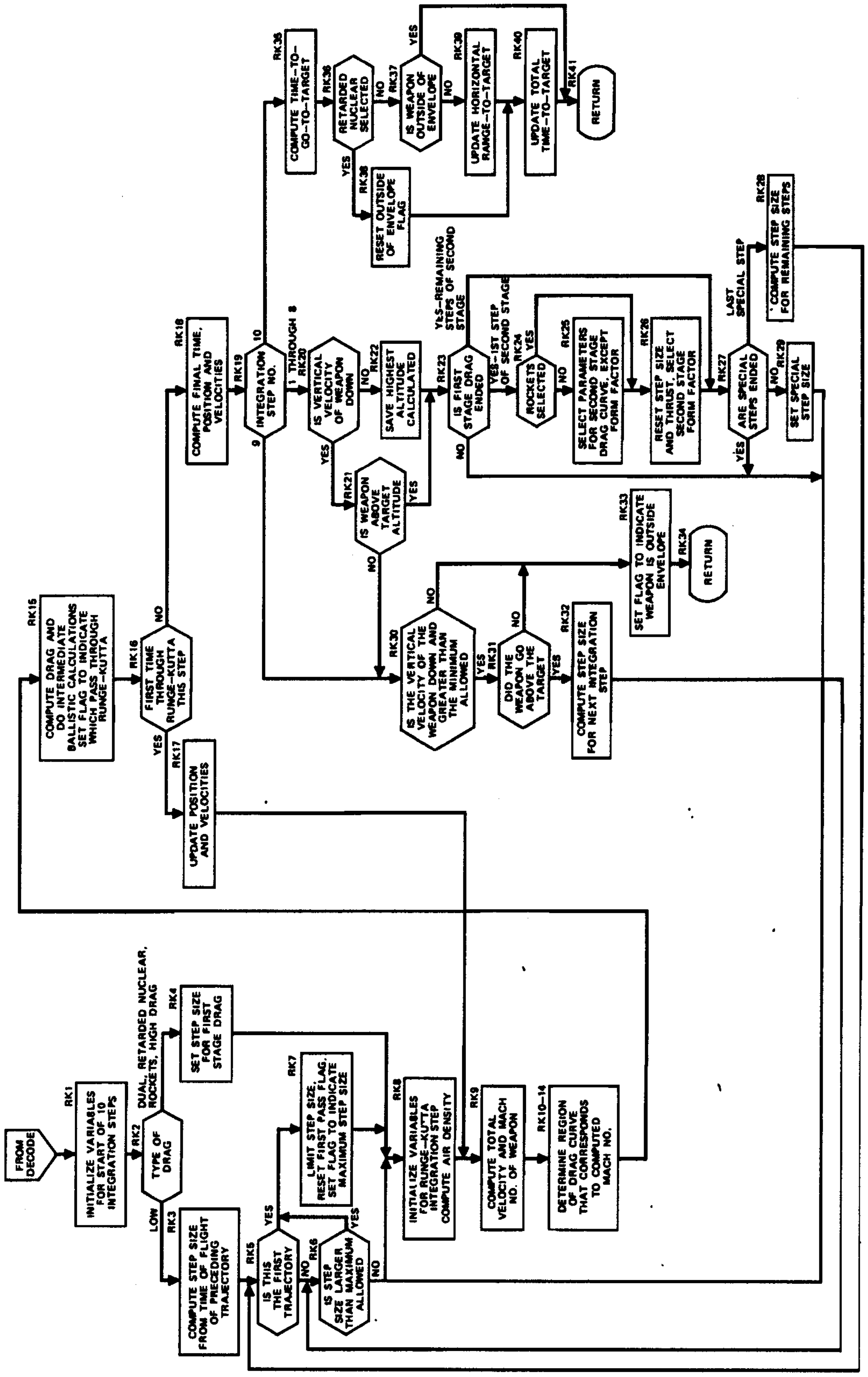
FROM DECODE
RK1 INITIALIZE VARIABLES FOR START OF 10 INTEGRATION STEPS
RK2 TYPE OF DRAG
DUAL, RETARDED NUCLEAR, ROCKETS, HIGH DRAG
LOW
RK3 COMPUTE STEP SIZE FROM TIME OF FLIGHT OF PRECEDING TRAJECTORY
RK4 SET STEP SIZE FOR FIRST STAGE DRAG
RK5 IS THIS THE FIRST TRAJECTORY
YES
NO
RK6 IS STEP SIZE LARGER THAN MAXIMUM ALLOWED
RK7 LIMIT STEP SIZE, RESET FIRST PASS FLAG, SET FLAG TO INDICATE MAXIMUM STEP SIZE
RK8 INITIALIZE VARIABLES FOR RUNGE-KUTTA INTEGRATION STEP COMPUTE AIR DENSITY
RK9 COMPUTE TOTAL VELOCITY AND MACH NO. OF WEAPON
RK10-14 DETERMINE REGION OF DRAG CURVE THAT CORRESPONDS TO COMPUTED MACH NO.
RK15 COMPUTE DRAG AND DO INTERMEDIATE BALLISTIC CALCULATIONS SET FLAG TO INDICATE WHICH PASS THROUGH RUNGE-KUTTA
RK16 FIRST TIME THROUGH RUNGE-KUTTA THIS STEP
RK17 UPDATE POSITION AND VELOCITIES
RK18 COMPUTE FINAL TIME, POSITION AND VELOCITIES
RK19 INTEGRATION STEP NO.
9
10
1 THROUGH 8
RK20 IS VERTICAL VELOCITY OF WEAPON DOWN
RK21 IS WEAPON ABOVE TARGET ALTITUDE
RK22 SAVE HIGHEST ALTITUDE CALCULATED
RK23 IS FIRST STAGE DRAG ENDED
YES-REMAINING STEPS OF SECOND STAGE
YES-1ST STEP OF SECOND STAGE
RK24 ROCKETS SELECTED
RK25 SELECT PARAMETERS FOR SECOND STAGE DRAG CURVE, EXCEPT FORM FACTOR
RK26 RESET STEP SIZE AND THRUST, SELECT SECOND STAGE FORM FACTOR
RK27 ARE SPECIAL STEPS ENDED
LAST SPECIAL STEP
RK28 COMPUTE STEP SIZE FOR REMAINING STEPS
RK29 SET SPECIAL STEP SIZE
RK30 IS THE VERTICAL VELOCITY OF THE WEAPON DOWN AND GREATER THAN THE MINIMUM ALLOWED
RK31 DID THE WEAPON GO ABOVE THE TARGET
RK32 COMPUTE STEP SIZE FOR NEXT INTEGRATION STEP
RK33 SET FLAG TO INDICATE WEAPON IS OUTSIDE ENVELOPE
RK34 RETURN
RK35 COMPUTE TIME-TO-GO-TO-TARGET
RK36 RETARDED NUCLEAR SELECTED
RK37 IS WEAPON OUTSIDE OF ENVELOPE
RK38 RESET OUTSIDE OF ENVELOPE FLAG
RK39 UPDATE HORIZONTAL RANGE-TO-TARGET
RK40 UPDATE TOTAL TIME-TO-TARGET
RK41 RETURN

FIG. 4. Descriptive Flow Diagram of Fig. 3.

The integration step counter, NUM, is updated at the end of block RK18. Since NUM equals one after the first integration step, the test at block RK19 sends the logic to RK20. Assuming a normal dive release condition above the target, the logic will be routed to block RK21, then RK23, and finally back to RK5 for the beginning of the next integration step.

During first-pass in a normal dive trajectory, this process will continue until the trajectory is below the target altitude. At this point the target altitude test in block RK21 fails and the logic is routed through blocks RK30-RK32 for routing back to impact point. After the last integration step, blocks RK35-RK40 make a small straight-line correction in time-of-flight and impact range.

The second and all succeeding trajectory calculations will be similar, except ICALC will be plus. The calculations for a given trajectory end with a return at block RK34 or RK41. Also, KFLAG will be set to zero or plus depending on whether the trajectory is inside or outside the envelope, respectively.

Computations for an ITYPE 1 Weapon

The discussion in this section will not repeat the details discussed in the last section when describing the calculations for an ITYPE 1 weapon (e.g., Rockeye II). The process for this case is similar to that of the ITYPE -1 weapon until block RK2 of Fig. 3 at which time the logic is then routed to block RK4 where D is calculated. The value of D is equal to the time necessary for the dispenser (the fin for retarded Snakeye) to open divided by the number of steps, NSTEP, that are used for the first stage.

After NSTEP integration steps, the test in block RK23 routes the logic through the necessary blocks RK24-RK29 in order to select the parameters for the second-stage drag curve and reset the integration step size for the remaining steps of the trajectory calculation.

For some ITYPE 1 weapons (e.g., Sadeye, retarded Snakeye) the deceleration after dispenser or fin opening at the beginning of the second stage is so great that special step sizes are needed to slow the weapon down the first NT steps in the trajectory calculations. Blocks RK27 and RK29 serve this purpose. Block RK28 resets the integration step size after the NT deceleration steps, if any, to the time remaining, TS-T, divided by approximately the number of remaining integration steps.

Computations for an ITYPE 2 Weapon

ITYPE 2 weapons are similar to ITYPE 1 weapons except only one drag curve is needed for both stages. For example, rockets only need a change in bomb coefficient using the same drag curve for the second stage when thrust is zero. Similarly, very high-drag bombs like the Mk 106 need to be decelerated the first few steps of the weapon trajectory but need only one drag curve. This qualifies the Mk 106 for an ITYPE 2 weapon.

Computations for an ITYPE 0 Weapon

The ITYPE 0 weapon is a special case ITYPE 1 weapon. For example, the retarded Mk 43 parachute bomb has two drag curves like the ITYPE 1 weapon and requires a large number of deceleration steps to handle the opening of the parachute and its effect on slowing the weapon down to a near-constant velocity. After that, the horizontal velocity component of the bomb is very small and the vertical velocity component remains almost constant due to the parachute.

The differentiating factor of the ITYPE 0 weapon from the ITYPE 1 weapon is the way the straight-line correction for time-of-flight and impact range is handled in blocks RK35-RK40. No correction in impact range is made after the final integration step. However, time-of-flight is corrected like all other weapons. Good accuracy is obtained over almost an unlimited envelope by this technique for retarded Mk 43, Mk 57, and Mk 61 parachute bombs.

Sizing and Timing of Algorithm

A study was made to determine the number of computer words and computational time required for the algorithm using as a reference the IBM 4-Pi Model TC-2 airborne computer. To obtain a count for the number of computer words and the type of instructions needed, a machine language program was written. Without scaling, a core size of 551 words was estimated to accommodate the present program and the choice of 28 weapons currently implemented in the A-7E aircraft. The breakdown of the word estimate is:

Constants for 28 weapons	174 words
Instructions to locate requested weapon and load appropriate constants	129 words
Runge-Kutta integration part of the algorithm including the calculation of air density, Mach number, and the square root of $V_x^2 + V_y^2$	122 words
Control logic of the algorithm	126 words

The IBM 4-Pi Model TC-2 computer technical reference description gives execution time (in microseconds) of the following instructions:

Instruction	Time, μsec
Load	5.0
Store	5.0
Add	5.0
Subtract	5.0
Multiply	20.0
Divide	21.0
Branch	2.5
Complement	2.5
Shift	5.0 plus number of positions shifted

These instructions and corresponding times were used in the above-mentioned machine language program.

For 10 steps to compute a weapon trajectory, the maximum time is 16.5 milliseconds and the minimum is 16.3 milliseconds without scaling. It should be pointed out that a smaller number of integration steps could be used to decrease the computation time at the expense of a smaller weapon envelope.

MODES OF WEAPON DELIVERY

The algorithm logic is very general and can be used for dive, toss, loft, and over-the-shoulder weapon delivery. The logic handles the cases where target altitude, YT, is above or below the present aircraft altitude at weapon release. The algorithm is stable at any altitude, velocity, dive angle, and pullup maneuver the present A-7E aircraft is capable of for the 28 weapons listed in Appendix B.

As mentioned earlier, KFLAG will be zero if the weapon trajectory is inside the envelope and plus if outside the envelope. The envelope for a particular weapon can be enlarged with an increase in total number of integration steps in the algorithm at the expense of more computation time. Likewise, computation time can be cut down at the expense of a smaller envelope. Ten integration steps will cover all the current type weapons with their envelopes.

COMPRESSION OF DRAG DATA

The behavior of a particular weapon's drag coefficient, C_D, as a function of Mach number must be known to calculate impact ranges for that weapon. Normally, this information is given in tabular form with some weapons having more than 100 points in the table. Most aircraft have the capability to carry many different types of weapons. It is desirable to store all the C_D information of an aircraft's weapon repertoire in the airborne computer to avoid loading the C_D data for different combinations of weapons. This requires storing drag coefficient data in a more efficient form than a table.

The choice of a scheme to approximate the drag coefficients for a given set of weapons implemented in a given aircraft is influenced by the following factors:

1. Computer storage
2. Computer computation time
3. Delivery envelope for each weapon
4. Allowable downrange and time-of-flight errors.

It would be ideal to use one general expression to fit all the drag tables for the weapon repertoire of a given aircraft. This would require a fairly powerful expression, because weapons like guns and rockets need C_D values for Mach numbers much higher than free-fall bombs. Consequently, extra storage would be spent on weapons that do not require the extra capability.

Several free-fall weapons have drag coefficient curves that differ only by a multiplicative factor. The Mk 80 low-drag series are one such group of weapons with the Mk 84 drag curve conventionally taken as the reference. Similarly, the Mk 106 Mod 2 and 3, CBU bomblet, and Sadeye bomblet are another group with the Garve drag curve as the reference. The region of interest of many more drag curves can be approximated by multiplying these two reference drag curves by a factor and then translating them along the C_D and M axes until they match the original drag curves.

To further explain approximating weapon drag curves by this method, let $C_D(M)$ and M be the drag coefficient and Mach number of a weapon drag curve. Let $\overline{C}_D(M)$ be the drag coefficient of a reference drag curve and let $\overline{A}$, $\overline{B}$, and DM be the multiplicative factor, drag coefficient translation, and Mach number translation, respectively, that when applied to the reference drag curve will approximate the weapon drag curve. That is,

$$C_D(M) \approx \overline{A} \times \overline{C}_D(M + DM) + \overline{B} \qquad (5)$$

A hypothetical weapon drag curve, C_D, and reference curve, $\overline{C}_D$, are shown in Fig. 5. For simplicity of discussion, the Mach number intervals $M_2 - M_1$ and $M_4 - M_3$ are equal. Suppose the $\overline{C}_D$ curve is multiplied by an $\overline{A}$ so that the C_D and $\overline{A} \times \overline{C}_D$ curves are nearly identical but for a C_D - axis translation, B, and an M-axis translation, DM. Thus C_D could be expressed as Eq. 5.

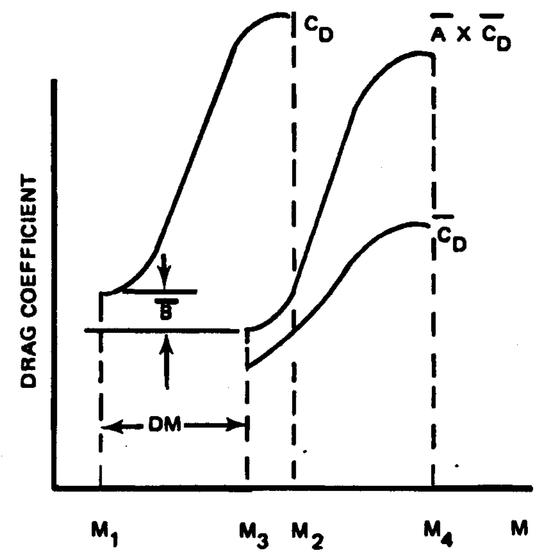

FIG. 5. Curve Fitting C_D.

For many weapons, Eq. 5 will not give an accurate fit over a wide interval of Mach number; instead, $\overline{A}$, $\overline{B}$, and DM should be chosen so that the C_D fit is best for the Mach number interval that corresponds to the delivery envelope for a particular aircraft.

Considering the weapons implemented in the A-7E, both the Mk 84 and Garve reference drag curves are necessary to handle the free-fall weapons. Instead of extending these curves to handle guns and rockets that have a much larger Mach number range, a rocket-gun reference drag curve is used in order to save computer storage and computation time.

CURVE FITTING THE DRAG COEFFICIENTS

The last section discussed what reference curves were needed and how they were used to approximate weapon drag curves for use in the algorithm. The next step is to represent the reference drag curves versus Mach number for the algorithm. The form of the function used should minimize computer storage, logic, and computation time for the allowable error in downrange and time-of-flight for the different weapons. It was decided that each of the three reference drag curves be divided into three regions with $\overline{C}_D$ expressed as a second-order polynomial in Mach number for each region. Thus, for a particular reference drag curve,

$$\overline{C}_D(M) = a_0 + a_1 \times M + a_2 \times M^2 \tag{6}$$

where coefficients, a_0, a_1, and a_2 depend on the region the Mach number M is in. Using Eq. 5 and 6, the drag coefficient for a particular weapon is expressed as

$$C_D(M) \approx \overline{A} \times [a_0 + a_1 \times (M + DM) + a_2 \times (M + DM)^2] + \overline{B} \tag{7}$$

In order to save computer storage and reduce the computation time for the algorithm, another step was taken in the compression of drag data. The computation C x ($\pi/8\ C_D$), CKDG, is made in the Runge-Kutta part of the algorithm where C is the bomb factor, d^2/W, where d is the weapon diameter in ft and W is the mass in slugs. One constant per weapon in computer storage and two multiplications per integration step is saved by expressing CKDG as

$$CKDG = CF \times \overline{CKDG} + DKG \tag{8}$$

where

$$CF = \overline{A} \times (d/\overline{d})^2 \times \overline{W}/W \tag{9}$$

$$\overline{CKDG} = \overline{d}^2/\overline{W} \times \pi/8 \times \overline{C}_D \tag{10}$$

and

$$DKG = \frac{d^2}{W} \times \frac{\pi}{8} \times \overline{B} \quad . \qquad (11)$$

In Eq. 9 and 10, $\overline{d}$ and $\overline{W}$ are the assumed diameter and mass for the reference curve $\overline{C}_D$. Finally, the form of CKDG in the algorithm is

$$CKDG(M) = CF \times [b_0 + b_1 \times (M + DM) + b_2 \times (M + DM)^2] + DKG \qquad (12)$$

Appendix B gives a list of b_0, b_1, and b_2 for each region of the three reference curves and the values of CF, DM, and DKG for 28 weapons implemented in the A-7E.

In the algorithm, thrust appears in the total drag function

$$HH = TH/V - \rho \times CKDG \times V \qquad (13)$$

where

TH = thrust/W

ρ = air density

V = weapon velocity

and CKDG is defined above in Eq. 8 through 11. Figure 6 shows thrust versus time for a typical rocket approximated by a constant, thrust = total impulse/T1, where T1 is decreased from actual motor burn time in order to make thrust equal to the average thrust of region 2. The mass of the rocket, W, obviously decreases during rocket burn time. However, it is adequate for W to be approximated by a constant by varying W until downrange error is minimized. The second stage of the rocket, time after weapon release greater than T1, is free fall and TH will be zero.

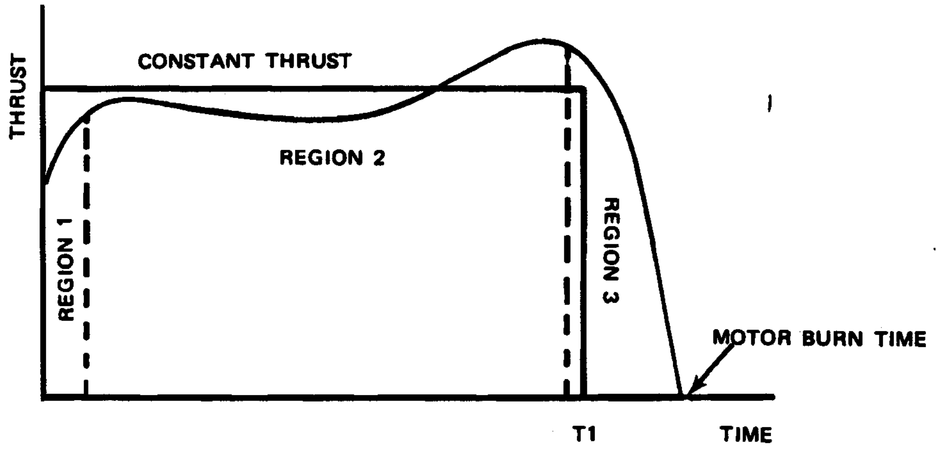

FIG. 6. Thrust Versus Time Curve for a Typical Rocket.

CONCLUSIONS

The technique described in this document can be used to predict the impact point and time-of-flight of any unguided weapon currently in the inventory. There is no restriction in the basic algorithm as to mode of weapon delivery. This wide applicability stems directly from the fact that an accurate representation of the differential equations of motion is numerically integrated to near-perfect accuracy by the algorithm. Furthermore, the algorithm can be extended to handle the following problems:

1. Variable wind profile
2. Nonstandard air density
3. Altitude and slant range fuzed weapons
4. Shrike and other similar guided rockets

It has been established that the basic algorithm can be coded successfully for an airborne digital computer. This coding induces a nominal requirement on computer memory and provides a continuously computed impact point in near real time.

There is no doubt that the algorithm can provide accuracy, flexibility, and efficiency in airborne weapon control applications.

Appendix A
OPTIMAL SECOND-ORDER RUNGE-KUTTA FORMULAS

BACKGROUND

For the numerical solution of ordinary differential equations, a large class of methods can properly be called Runge-Kutta methods. To illustrate the use and characteristics of these methods, attention will be restricted to the scalar-valued differential equation problem

$$y' = F(x,y) \qquad y(x_0) = y_0 \tag{14}$$

Later this problem will be interpreted in the case where y is vector-valued (i.e., for a system of n differential equations).

The fundamental R-K technique is to replace the result of truncating a Taylor series expansion of the form

$$y_{n+1} = y_n + h\, y_n' + \frac{h^2}{2!}\, y_n'' + \frac{h^3}{3!}\, y_n''' + \ldots \tag{15}$$

by an approximation in which y_{n+1} is computed from a formula of the type

$$y_{n+1} = y_n + h[a_0\, F(x_n,y_n) + a_1\, F(x_n + \alpha_1 h,\ y_n + \beta_1 h)$$
$$+ \ldots + a_p\, F(x_n + \alpha_p h,\ y_n + \beta_p h)] \tag{16}$$

Here, the a's, α's, and β's are to be determined so that, if the right side of Eq. 16 were expanded in powers of the integration step h, the coefficients of a certain number of the leading terms would agree with the corresponding coefficients of Eq. 15. The number of terms to which this expansion agrees with the Taylor series is called the order of the method. There is a fundamental difference between truncating Eq. 15 and using a formula like Eq. 16. In the Taylor series method, all required information is obtained at the point x_n, and the truncated series is used to advance the solution over an interval of length h to the point x_{n+1}. On the other hand, the R-K formulas use information obtained at certain points located in a disk centered at x_n. In a sense, Eq. 16 is a linear extrapolation of x_n to x_{n+1} over an interval of length h, using a weighted average of the slopes F(x,y) at specified points in the disk. In practical applications, the R-K methods of lower order are almost always more efficient than the Taylor-series approach. The second-order R-K methods are particularly simple. The third, fourth, fifth, and so on, order formulas are successively more accurate, but increase greatly in complexity for the higher orders. For the fifth and higher order formulas, the gain in accuracy is usually far outweighed by the increased computation time required for each integration step.

The choice of the "best" R-K formula for a particular problem is not obvious. The lower order formulas are simple, but require more integration steps than the more accurate higher order formulas. On the other hand, the complexity of the higher order formulas may require so much computer time that this becomes the overriding consideration. Generally speaking, it is safer to use the lower order formulas when discontinuities in the coefficients of the differential equation can be expected. Such discontinuities frequently occur in bomb trajectories (e.g., drogue deployment or rocket thrust termination). The second-order R-K formulas also afford the greatest flexibility in the distribution and size of the integration steps. This is helpful when a change is made in coordinate systems, as in the bombing algorithm. These considerations, along with their simplicity, have led to the choice of the second-order R-K method as the most appropriate for the bombing algorithm.

Once it is decided which order of formula will be used, there is still the nontrivial problem of selecting the "best" of the second-order methods. As will be shown below, the general second-order R-K formula contains a free parameter that may be chosen arbitrarily. The free parameter is usually chosen so as to either yield symmetrical formulas or simplify the appearance of the resulting expressions. These considerations are more important if the computations are to be done by hand and much less important when a high-speed digital computer is used. The classical methods of Heun, Euler, modified Euler, etc., are generated for certain choices of the parameter. Certain other choices of this parameter yield what are called optimal R-K formulas (Ceschino and Kuntzmann, Ref. 4, or Ralston, Ref. 5). In this context, the words "best" and "optimal" must be regarded with some degree of caution. For instance, the optimal formulas obtained by Ralston have been developed to minimize the coefficients of certain terms appearing in the expression from the truncation error of the formula. There are differential equations for which this kind of optimization yields worse results than one of the classical methods. (See Ref. 4.) This can happen because the optimization procedures used are directed at an arbitrary first-order differential equation. In what follows, optimal second-order R-K formulas are developed specifically for the differential equations involved in the bombing problem. In this context, the word "optimal" can be interpreted as meaning very accurate for the bombing problem.

ANALYSIS

The process of determining the constants in Eq. 16 for the second-order R-K methods is simplified if Eq. 16 is written in the equivalent form

$$y_{n+1} = y_n + a_0K_0 + a_1K_1 \tag{17}$$

where

$$\begin{aligned} K_0 &= h\ F(x_n, y_n) \\ K_1 &= h\ F(x_n + ch,\ y_n + bK_0) \end{aligned} \tag{18}$$

It will be shown that the parameters a_0, a_1, c, and b can be selected so that Eq. 17 and 18 constitute a second-order method. The idea is to expand K_1 in a two-dimensional Taylor series of the form

$$\begin{aligned} K_1 = h\Bigg[F_n &+ ch \left.\frac{\partial F}{\partial x}\right|_n + bK_0 \left.\frac{\partial F}{\partial y}\right|_n + 1/2\left(c^2h^2 \left.\frac{\partial^2 F}{\partial x^2}\right|_n \right. \\ &\left. + 2cbhK_0 \left.\frac{\partial^2 F}{\partial x\ \partial y}\right|_n + b^2\ K_0^2\ \frac{\partial^2 F}{\partial y^2}\right)\Bigg] + 0(h^3) \end{aligned} \tag{19}$$

where

$$F_n = F(x_n, y_n)$$

The term $0(h^3)$ means that, under sufficient conditions on the smoothness of F(x,y), the truncation error tends to zero like h^3. Substituting Eq. 18 and 19 into Eq. 17 yields

$$y_{n+1} = y_n + (a_0 + a_1)\ F_nh + a_1\left(c \left.\frac{\partial F}{\partial x}\right|_n + bF_n \left.\frac{\partial F}{\partial y}\right|_n\right) h^2 + 0(h^3) \tag{20}$$

By the chain rule of differential calculus, the differential Eq. 14 yields

$$y'' = \frac{\partial F}{\partial x} + F \frac{\partial F}{\partial y} \tag{21}$$

By using the differential Eq. 14 and 21, the Taylor series Eq. 15 can be written

$$y_{n+1} = y_n + hF_n + \frac{h^2}{2!}\left(\frac{\partial F}{\partial x} + F \frac{\partial F}{\partial y}\right)_n + 0(h^3) \tag{22}$$

By equating the coefficients of the terms hF, h^2 $\partial F/\partial x$, and h^2 F $\partial F/\partial y$ in Eq. 20 and 22, the three following relations are obtained.

$$a_0 + a_1 = 1 \qquad ca_1 = \frac{1}{2} \qquad a_1 b = \frac{1}{2} \tag{23}$$

This is a system of three equations in four unknowns which can only be solved in terms of one arbitrary nonzero parameter; taking c as this parameter yields

$$a_1 = \frac{1}{2c} \qquad b = c \qquad a_0 = \frac{2c-1}{2c} \tag{24}$$

Therefore, the general form of the R-K method (Eq. 17 and 18) is

$$y_{n+1} = y_n + \frac{1}{2c}\left[(2c - 1) K_0 + K_1\right] \tag{25}$$

where

$$K_0 = h\ F(x_n, y_n)$$

$$K_1 = h\ F(x_n + ch,\ y_n + cK_0) \tag{26}$$

Since for any nonzero c the Taylor-series Eq. 22 and the R-K Eq. 25 agree through the coefficients of the h^2 terms, Eq. 25 and 26 are the general second-order R-K formulas.

By taking higher order terms in Eq. 20 and 22, the truncation error E_n in Eq. 25 can be estimated by

$$E_n = -\frac{ch^3}{12}\left[\left(\frac{3c-2}{c}\right) y_n''' - 3 \left.\frac{\partial F}{\partial y}\right|_n y_n''\right] + 0(h^4) \qquad (27)$$

(For a detailed derivation, see Ref. 6). The first term in Eq. 27 is the error expression used to develop the "optimum" R-K formulas of Ralston. Except in trivial cases, it is not even possible to select c so that E_n behaves like $0(h^4)$. The various classical second-order R-K formulas are obtained from the generic form of Eq. 27 by choosing the appropriate value for c.

It can be shown that, if F is smooth enough, the numerical solution obtained from Eq. 25 for any nonzero c converges to the exact solution of Eq. 14 as h approaches zero. In real-time applications, such as in a weapon delivery computer, it is desirable to take as few integration steps as possible. This means that in the bombing algorithm the step size h should be as large as possible without unduly degrading the results. Since the error induced by Eq. 25 depends not only on h, but also on c, it seems reasonable to conjecture that the error can be made small by an appropriate choice of c. In fact, the best value of c depends not only on h, but also on the differential Eq. 14 and its initial condition, y_0. If the differential equation, the step size h, and the number of steps are fixed, the optimal value of c depends only on the initial condition y_0. If the initial conditions are generated by a continuous function (e.g., an aircraft path), the optimal value of c for smooth enough Fs in the differential equation depends continuously on the initial value y_0. If the behavior of c with respect to y_0 can be predicted beforehand, the best possible numerical results for the step size chosen are obtained from Eq. 25 and 26.

Before discussing how the optimum values of c are obtained in a practical problem, some decisions must be made as to the amount of error that will be tolerated. In many applications the value of c can be adjusted so that the total truncation error is zero. For instance, in applying this to the weapon delivery problem it appears that values of c always exist which yield zero error in the numerical solution. However, a zero-error solution may not be needed or even desirable in practice. It will become clear in the following discussion that there is a trade-off between how accurate a solution is needed and the amount of computation required to generate c as a function of the initial condition y_0.

The analytical determination of c is equivalent to solving the differential Eq. 14 in closed form. Of course, this is not usually possible. The obvious scheme is to solve Eq. 14 numerically using a small enough integration step so that an accurate solution is generated. This can be done for selected values of the initial condition y_0 and used as reference values for determining c. Once this has been done, the step size h and the number of steps to be used by the optimal formula must be chosen. Then, for any one of the initial conditions, Eq. 14 can be solved for a sequence of c values. Then, the value of c for which the best results are obtained (usually the one that gives zero error) can be chosen. This procedure will generate a discrete sample of the behavior of c as a function of the initial condition. Then, the sample points can be approximated with an appropriate fitting function to give an expression for c as a function of the initial condition y_0.

The above description of the determination of c has purposely been discussed in terms of imprecise generalities. There are two reasons for this: (1) tying down all the loose mathematical ends is outside the scope of this discussion; and (2) even if the technique were put on a firm theoretical basis, many assumptions and compromises would be necessary to determine its applicability to the weapon delivery algorithm. However, it is clear that the process for determining c could be viewed as a problem in optimization theory where the objective function (the truncation error) depends on solution of a differential equation. In the next section the application of this technique to the weapons delivery algorithm will be discussed. This will clarify some of the ideas involved and will show what must be done when more than one differential equation is involved.

APPLICATION TO THE BOMBING ALGORITHM

Systems of differential equations are involved in the bombing algorithm. The previously discussed technique carries over to systems of differential equations with very little change. To illustrate this, the general R-K equations will be given for a system of two equations in two unknown functions. Then, the generalization to n such equations will be obvious. Let,

$$\frac{dx}{dt} = F(t,x,y) \tag{28}$$

$$\frac{dy}{dt} = G(t,x,y)$$

with initial conditions $x(t_0)$ and $y(t_0)$ specified. The general second-order R-K formulas for this problem are

$$x_{n+1} = x_n + \frac{1}{2c}[(2c - 1) K_0 + K_1] \tag{29}$$

$$y_{n+1} = y_n + \frac{1}{2c}[(2c - 1) M_0 + M_1]$$

where

$$\begin{aligned} K_0 &= h\ F(t_n, x_n, y_n) \\ M_0 &= h\ G(t_n, x_n, y_n) \\ K_1 &= h\ F(t_n + ch, x_n + cK_0, y_n + cM_0) \\ M_1 &= h\ G(t_n + ch, x_n + cK_0, y_n + cM_0) \end{aligned} \tag{30}$$

Equations of the type of Eq. 29 are used in the weapon delivery problem.

The exact solution of the system Eq. 28 is a vector with components x(t) and y(t). At some specific point, say $t = t_1$, the numerical solution vector has components $X(t_1)$ and $Y(t_1)$. The error in the numerical solution is measured by how close the two solutions are in the vector norm sense. For instance, for the Euclidean norm the error is given by

$$E(t_1) = \sqrt{(x(t_1) - X(t_1))^2 + (y(t_1) - Y(t_1))^2} \tag{31}$$

The parameter c can now be chosen to minimize $E(t_1)$. Clearly, the optimal value of c depends now on both initial conditions $x(t_0)$ and $y(t_0)$. It is usually not possible to choose c so that $E(t_1)$ is zero. However, it can normally be chosen so that the error in one component of the numerical solution is zero. In the weapon delivery problem, by far the most important component in the solution vector is the impact range. Therefore, c can be adjusted so that the range at impact has zero error. The altitude variable Y is automatically exact at impact since it is forced to target altitude. Since the velocity components V_x, V_y, and the time of fall, t_f, are closely coupled with the other two variables, they also tend to be very accurate at the impact point.

After making computer runs using typical release conditions for each type of weapon, c equal to 0.75 seemed to work best for those weapons using the Mk 84 reference drag curve and c equal to 0.7 for all other weapons with higher drag. Since the low-drag bombs are not as sensitive to a change in c as the higher drag weapons and the value 0.7 works almost as well for c, it was decided to use c equal to 0.7 for all weapons in order to save computer storage. It should be pointed out that picking the value of 0.7 for c was based on the algorithm using 10 integration steps. If more than 10 integration steps are used, 0.7 for c would be adequate even though some other value might work better. However, if fewer than 10 integration steps are used, it might turn out that more than one c should be used to handle all the weapons.

Appendix B

WEAPON CONSTANTS FOR DECODE

REFERENCE DRAG CURVES

The use of three reference drag curves for the compression of drag data was discussed earlier in this report. The coefficients for the Mk 84, Garve, and rocket-gun reference drag curves are given in Table 1. For further detail see the FORTRAN listing of the algorithm in Appendix C under statement labels 32, 33, and 34.

TABLE 1. Reference Drag Curve Coefficients and Cuts.

Coefficient	Reference curve		
	Mk 84	Garve	Rocket-gun
Region 1:			
b_0	1.572924×10^{-3}	3.53503924	0.104115
b_1	0	-3.34778216	-0.230347
b_2	0	2.87262413	0.167644
Region 2:			
b_0	$4.67840889 \times 10^{-2}$	11.2616503	-0.194037
b_1	-0.109711069	-27.4162512	0.401478
b_2	6.6548007×10^{-2}	21.7308359	-0.164612
Region 3:			
b_0	-0.116380157	-23.7915472	7.33246×10^{-2}
b_1	0.217643894	44.2607764	-2.03275×10^{-2}
b_2	$-9.76706845 \times 10^{-2}$	-14.4996046	2.44682×10^{-3}
Cut			
First cut, CT_1	0.834	0.622	1.032
Second cut, CT_2	0.977	0.885	1.3

WEAPON CONSTANTS

Table 2 lists the necessary weapon-dependent constants for the algorithm for 28 weapons implemented in the A-7E airborne computer. The definition of terms is given on pp. 11 and 12, except for IDNO and IREF. An identification number, IDNO, is assigned to each weapon for programming purposes in Appendix C. IREF 1, 2, and 3 refer to reference drag curves Mk 84, Garve, and rocket-gun, respectively. Some weapons have drag curves that can be approximated by a constant in the region of interest with only a negligible loss in bombing accuracy. If so, CFORM1 and CFORM2 will be zero and, therefore, any of the three reference curves can be used. Appendix C gives a FORTRAN listing of the algorithm and the weapon-dependent constants for the 28 weapons listed in Table 2.

DECODE

The weapon-dependent constants required for the algorithm are stored in DECODE. Also, DECODE contains the necessary logic to load the correct constants for the particular weapon the pilot has selected. As mentioned in a previous section, on first-pass the necessary weapon constants are loaded in DECODE. After first-pass there is no need to go through DECODE before entering the algorithm for a trajectory calculation.

As previously discussed, an IBM 4-Pi Model TC-2 airborne computer was used as a reference in determining the number of computer words and timing for the algorithm. Using the scheme that will be outlined below, a total of 174 words were needed to store the weapon-dependent constants for 28 weapons implemented in the A-7E, plus 129 words for instructions to locate requested weapon and load appropriate constants. The scheme used for DECODE in reality will depend on the particular aircraft and its fire control requirements and how the algorithm is interfaced in the airborne computer.

In order to save computer words in DECODE, the weapon constants must be stored compactly with no duplication, if possible. At the same time, the method of loading the requested weapon constants should be simple and yet general enough to handle additional fire control requirements that may not have been anticipated. In our hypothetical scheme, one word is used to identify each weapon and has the form shown in Fig. 7. Figures 8 through 11 show in detail the storage table used. Bits 0 through 11 of the code word are tested to see which weapon constants are needed from storage of Fig. 8. For example, if the weapon constant DKG1 is required, then a 1 will appear in bit 2, etc. Figure 9 shows what the storage table of Fig. 8 looks like for the first five weapons with IDNO equal to 1, 2, 3, 4, and 5.

Bits 10 and 11 of the identification word are used to load the correct reference drag curve coefficients CC and CT as shown in Fig. 12. There are 9 numbers to load from CC and 2 numbers from CT.

TABLE 2. Weapon Constants in DECODE for 28 Weapons Currently Implemented in the A-7E Aircraft.

IDNO	Weapon	CFORM1	DM1	DKG1	CFORM2	DM2	DKG2	IREF	DMAX	DS	DS2	NSTEP	NT	ITYPE
1	Mk 43 Unretarded	0	0	2.5506 E-3	0	0	0	1	7.0	0	0	9	0	-1
2	Mk 57 Unretarded	0	0	6.2994 E-3	0	0	0	1	7.0	0	0	9	0	-1
3	Mk 61 Unretarded	0	0	4.01 E-3	0	0	0	1	7.0	0	0	9	0	-1
4	Mk 116 Weteye	3.9235 E-3	0	2.754 E-3	0	0	0	2	6.0	0	0	9	0	-1
5	Mk 76 with lug	3.9077 E-3	0	6.3649 E-3	0	0	0	2	6.0	0	0	9	0	-1
6	Mk 77 Fire	0	0	0.021266	0	0	0	1	6.0	0	0	9	0	-1
7	Mk 81	2.5704	0	0	0	0	0	1	7.0	0	0	9	0	-1
8	Mk 81 Snakeye Unretarded	0	0	9.767 E-3	0	0	0	1	7.0	0	0	9	0	-1
9	Mk 82 Mech Fuze	2.064	0	0	0	0	0	1	7.0	0	0	9	0	-1
10	Mk 82 Elec Fuze	1.4932	0	0	0	0	0	1	7.0	0	0	9	0	-1
11	Mk 83 Mech Fuze	1.3431	0	0	0	0	0	1	7.0	0	0	9	0	-1
12	Mk 83 Elec Fuze	1.21	0	0	0	0	0	1	7.0	0	0	9	0	-1
13	Mk 84	1.0	0	0	0	0	0	1	7.0	0	0	9	0	-1
14	Mk 117 Al	3.12	0	-1.223 E-3	0	0	0	1	7.0	0	0	9	0	-1
15	Mk 86 Wet sand filled	3.4972	0	0	0	0	0	1	7.0	0	0	9	0	-1
16	Mk 88 Wet sand filled	1.605	0	0	0	0	0	1	7.0	0	0	9	0	-1
17	Mk 82 Snakeye Unretarded	0	0	7.329 E-3	0	0	0	1	7.0	0	0	9	0	-1
18	Mk 82 Snakeye Retarded	0	0	7.329 E-3	1.6895 E-2	0.38	0.17166	1, 2	6.0	0.6617	0.35	1	4	1
19	Sadeye T1 = 4.0	2.0754	0	0	0.2217	0	0	1, 2	3.5	4.0	0.4	1	3	1
20	Rockeye II T1 = 4.372	2.2973	0.32	8.175 E-3	1.1136 E-2	0.41	0.16885	1, 2	4.5	2.186	0.6	2	2	1
21	CBU T1 = 5.0	2.2404	0	0	0.1178	0	0	1, 2	4.0	5.0	0.6	1	2	1
22	Mk 81 Snakeye Retarded	0	0	9.767 E-3	2.30625 E-2	0.38	0.23287	1, 2	6.0	0.679	0.35	1	4	1
23	Gun	2.9964	0	-0.014992	0	0	0	3	1.5	0	0	9	0	-1
24	Rocket	0.82	0	0	1.0	0	0	3	4.0	0.385	0	4	0	2
25	Mk 43 Retarded 0.4-sec delay	0	0	0	0	0	1.48	1	3.0	0.98	0.3	1	9	0
26	Mk 57 Retarded 0.8-sec delay	0	0	0	0	0	2.0	1	2.5	0.89	0.25	1	9	0
27	Mk 61 Retarded 0.6-sec delay	0	0	0	0	0	2.7	1	2.0	0.89	0.2	1	9	0
28	Mk 106 Mod 2	0.1514	0	0	0.1514	0	0	2	5.0	0.5	0.6	1	2	2

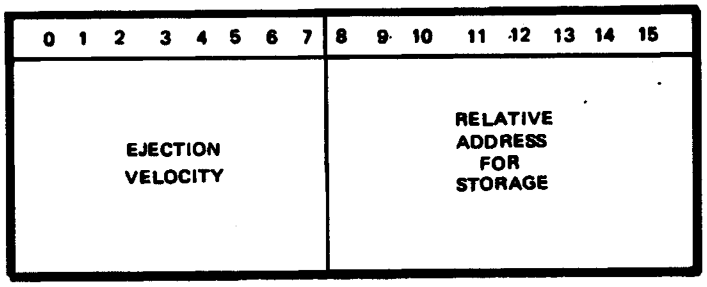

FIG. 7. Identification Word.

	CODE WORD	NEEDED FOR EVERY WEAPON
FIRST WEAPON	CFORM1 CFORM2	1 IN BIT 0 IF 1 IN BIT 1

	DKG1
	DKG2
		IF 1 IN BIT 2
		IF 1 IN BIT 3

	DM1
	DM2
		IF 1 IN BIT 4
		IF 1 IN BIT 5

	ITYPE, NSTEP, NT DS DMAX, DS2	IF 1 IN BIT 6

	FN
	SL
	VMUZ
		IF 1 IN BIT 7
		IF 1 IN BIT 8
		IF 1 IN BIT 9

SECOND WEAPON	CODE WORD	NEEDED FOR EVERY WEAPON

	CFORM1
	CFORM2
		IF 1 IN BIT 0
		IF 1 IN BIT 1

	DKG1 ⋮	IF 1 IN BIT 2 ⋮
	VMUZ	IF 1 IN BIT 9
⋮	⋮	⋮

FIG. 8. General Storage for DECODE.

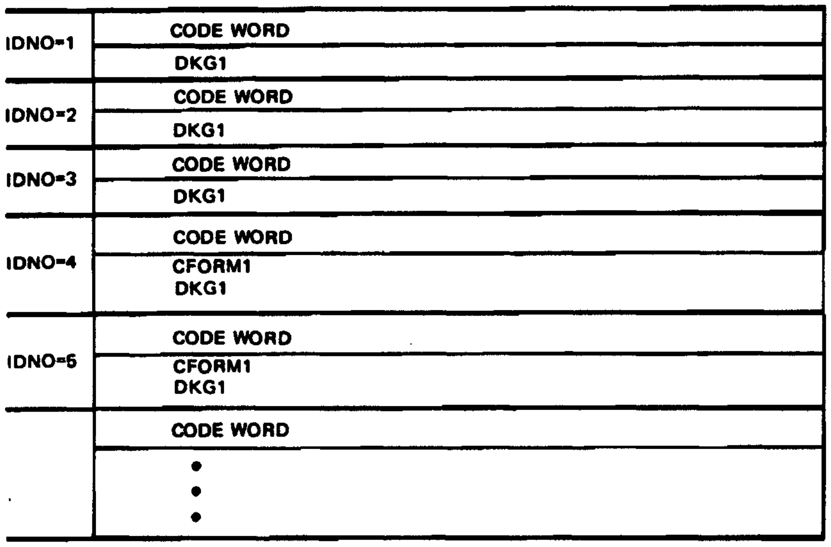

FIG. 9. Actual Storage Table for the First Five Weapons Listed in Table 1.

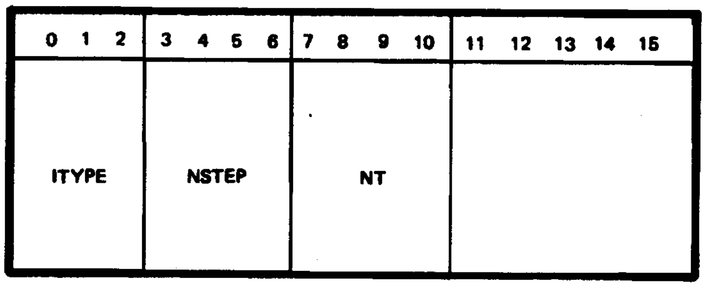

FIG. 10. Word Used in Fig. 8.

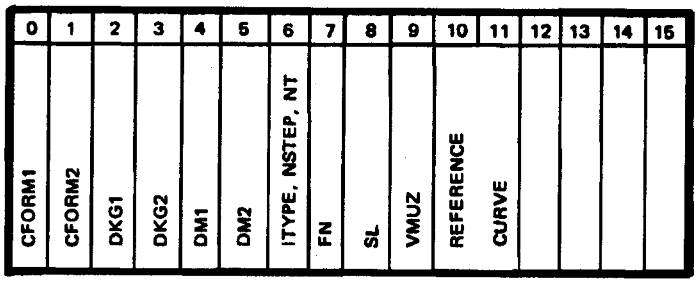

FIG. 11. Code Word Used in Fig. 8.

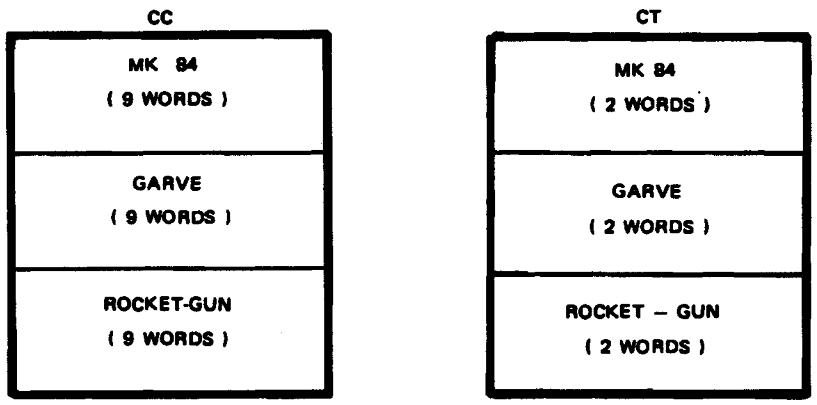

FIG. 12. Polynomial Coefficients CC and CT.

Appendix C
FORTRAN LISTING OF ALGORITHM

The FORTRAN listing given in this appendix follows the logic of the algorithm in Fig. 3 as closely as possible. That is, statement label 1001 corresponds to RK1 in Fig. 3, 1002 corresponds to RK2, etc. The first part of the FORTRAN listing is DECODE in a form somewhat like the DECODE scheme discussed in Appendix B. However, it is impossible to duplicate the machine language scheme in FORTRAN.

There is a brief computer printout after the FORTRAN listing of computed time-of-flight and impact range for one release condition per weapon. Each condition is "looped" four times to show rate of convergence to the impact range solution for the algorithm. Also, the computer printout can serve as a check if the listing is programmed by the reader on a different computer.

Tables 3-5, at the end of Appendix C, give a brief summary of impact range errors to be expected under some nominal release conditions for the Mk 82 unretarded Snakeye, Rockeye II, and Mk 82 retarded Snakeye weapons using the algorithm.

```
      DIMENSIØN  CT(2,2) , CC(3,3,2) , VKTS(20) , ALT(20) , DEG(20) ,
     +   NUMID(40)
      G = 32.174
      RAD = .01745329
      A = .7
      AA = 0.5/A
      YT = 0.0
      VYK = -5.0
      FRACT = .5
  898 FØRMAT (1H1,6X,'VKTS     ALT     DEG     TF        RANGE',4X,
     +   'IDNØ'/)
   99 CØNTINUE
      WRITE (108,898)
  100 CØNTINUE
  899 FØRMAT (16G5.0)
      READ (105,899) KIV,(VKTS(I),I=1,KIV)
      IF (KIV .EQ. 0)   GØ TØ 95
      READ (105,899) KY,(ALT(I),I=1,KY)
      READ (105,899) KDEG,(DEG(I),I=1,KDEG)
      READ (105,899) KIDNØ,(NUMID(I),I=1,KIDNØ),JBUG
      DØ 999 IIDNØ=1,KIDNØ
      IDNØ=NUMID(IIDNØ)
      DS2 = 0.0
      NT = 0
      NL = 10
      ICALC = 0
      CFØRM1 = 0.0
      CFØRM2 = 0.0
      DM1 = 0.0
      DM2 = 0.0
      DKG1 = 0.0
      DKG2 = 0.0
      VMUZ = 0.0
      VE = 0.0
      SL = 0.0
      DMAX = 7.0
      NSTEP = 9
      ITYPE = -1
      FN = 0.0
      KØN = 0
      IBØTH = 1
      NLM1 = NL-1
      GØ TØ (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
     +   21,22,23,24,25,26,27,28) , IDNØ
    1 IREF = 4
      DKG1 = 2.5506E-3
      GØ TØ 31
    2 IREF = 4
      DKG1 = 6.2994E-3
      GØ TØ 31
    3 IREF = 4
      DKG1 = 4.01E-3
```

```
        GØ TØ 31
 4      IREF = 2
        DMAX = 6.0
        CFØRM1 = 3.9235E-3
        DKG1 = 2.754E-3
        GØ TØ 31
 5      IREF = 2
        DMAX = 6.0
        CFØRM1 = 3.9077E-3
        DKG1 = 6.3649E-3
        GØ TØ 31
 6      IREF = 4
        DMAX = 6.0
        DKG1 = .021266
        GØ TØ 31
 7      IREF = 1
        CFØRM1 = 2.5704
        GØ TØ 31
 8      IREF = 4
        DKG1 = 9.767E-3
        GØ TØ 31
 9      IREF = 1
        CFØRM1 = 2.064
        GØ TØ 31
10      IREF = 1
        CFØRM1 = 1.4932
        GØ TØ 31
11      IREF = 1
        CFØRM1 = 1.3431
        GØ TØ 31
12      IREF = 1
        CFØRM1 = 1.21
        GØ TØ 31
13      IREF = 1
        CFØRM1 = 1.0
        GØ TØ 31
14      IREF = 1
        CFØRM1 = 3.12
        DKG1 = -1.223E-3
        GØ TØ 31
15      IREF = 1
        CFØRM1 = 3.4972
        GØ TØ 31
16      IREF = 1
        CFØRM1 = 1.605
        GØ TØ 31
17      IREF = 4
        DKG1 = 7.329E-3
        GØ TØ 31
18      IREF = 1
        DMAX = 6.0
        NSTEP = 1
```

```
      ITYPE = 1
      IBØTH = 2
      DKG1 = 7.329E-3
      CFØRM2 = 1.6895E-2
      DM2 = .38
      DKG2 = .17166
      DS2 = .35
      NT = 4
      DS = .5617
      SL = -.000269
      GØ TØ 31
10    IREF = 1
      DMAX = 3.5
      ITYPE = 1
      NSTEP = 1
      T1 = 4.0
      IBØTH = 2
      CFØRM1 = 2.0754
      CFØRM2 = .2217
      DS2 = .4
      NT = 3
      XSTEP = NSTEP
      DS = T1/XSTEP
      GØ TØ 31
20    IREF = 1
      DMAX = 4.5
      ITYPE = 1
      NSTEP = 2
      T1 = 4.372
      IBØTH = 2
      CFØRM1 = 2.2973
      DM1 = .32
      DKG1 = 8.175E-3
      CFØRM2 = 1.1134E-2
      DM2 = .41
      DKG2 = .16885
      DS2 = .6
      NT = 2
      XSTEP = NSTEP
      DS = T1/XSTEP
      GØ TØ 31
21    IREF = 1
      DMAX = 4.0
      ITYPE = 1
      NSTEP = 1
      T1 = 5.0
      IBØTH = 2
      CFØRM1 = 2.2404
      CFØRM2 = .1178
      DS2 = .6
      NT = 2
      XSTEP = NSTEP
```

```
          DS = T1/XSTEP
          GØ TØ 31
    22    IREF = 1
          DMAX = 6.0
          ITYPE = 1
          NSTEP = 1
          IBØTH = 2
          DKG1 = 9.767E-3
          CFØRM2 = 2.30625E-2
          DM2 = .38
          DKG2 = .23287
          DS2 = .35
          NT = 4
          DS = .679
          SL = -.000303
          GØ TØ 31
    23    IREF = 3
          DMAX = 1.5
          CFØRM1 = 2.9964
          DKG1 = -.014992
          VMUZ = 3300.0
          GØ TØ 31
    24    IREF = 3
          DMAX = 4.0
          ITYPE = 2
          NSTEP = 4
          CFØRM1 = .82
          CFØRM2 = 1.0
          T1 = 1.54
          FN = 1746.
          XSTEP = NSTEP
          DS = T1/XSTEP
          GØ TØ 31
    25    IREF = 4
          DMAX = 3.0
          NT = 9
          NSTEP = 1
          ITYPE = 0
          DS = .98
          DS2 = .3
          DKG2 = 1.48
          GØ TØ 31
    26    IREF = 4
          DMAX = 2.5
          NT = 9
          NSTEP = 1
          ITYPE = 0
          DS = .89
          DS2 = .25
          DKG2 = 2.
          GØ TØ 31
    27    IREF = 4
```

```
      DMAX = 2.0
      NT = 9
      NSTEP = 1
      ITYPE = 0
      DS = .89
      DS2 = .2
      DKG2 = 2.7
      GO TO 31
2     IREF = 2
      DMAX = 5.0
      ITYPE = 2
      NSTEP = 1
      CFORM1 = .1514
      CFORM2 = .1514
      NT = 2
      DS = .5
      DS2 = .6
31    GO TO (32,33,34,51) , IREF
32    CC(1,1,1) = 1.572924E-3
      CC(1,2,1) = 0.0
      CC(1,3,1) = 0.0
      CC(2,1,1) = 4.67840889E-2
      CC(2,2,1) = -.109711069
      CC(2,3,1) = 6.6548007E-2
      CC(3,1,1) = -.116380157
      CC(3,2,1) = .217643894
      CC(3,3,1) = -9.767068845E-2
      CT(1,1) = .834
      CT(2,1) = .977
      IF (IBOTH-1) 33,51,33
33    CC(1,1,IBOTH) = 3.53503924
      CC(1,2,IBOTH) = -3.34778216
      CC(1,3,IBOTH) = 2.87262413
      CC(2,1,IBOTH) = 11.2616503
      CC(2,2,IBOTH) = -27.4162512
      CC(2,3,IBOTH) = 21.7308359
      CC(3,1,IBOTH) = -23.7915472
      CC(3,2,IBOTH) = 44.2607764
      CC(3,3,IBOTH) = -14.4996046
      CT(1,IBOTH) = .622
      CT(2,IBOTH) = .885
      GO TO 51
34    CC(1,1,1) = .104115
      CC(1,2,1) = -.230347
      CC(1,3,1) = .167644
      CC(2,1,1) = -.194037
      CC(2,2,1) = .401478
      CC(2,3,1) = -.164612
      CC(3,1,1) = 7.33246E-2
      CC(3,2,1) = -2.032755E-2
      CC(3,3,1) = 2.44682E-3
      CT(1,1) = 1.032
      CT(2,1) = 1.3
51    DØ 999 IANG=1,KDEG
      THETA = DEG(IANG)*RAD
      DØ 999 IV=1,KIV
      DØ 999 IY=1,KY
      U = VKTS(IV)*1.6878
      DEL = ATAN(VE/U)
      V = SQRT(U*U+VE*VE)
      VXA = (V+VMUZ) * CØS(THETA-DEL)
      VYA = (V+VMUZ) * SIN(THETA-DEL)
      KØN = 0
      ICALC = 0
894   FØRMAT (1H )
      WRITE (108,894)
      DØ 999 JØN=1,4
1001  CF = CFØRM1
      DM = DM1
      DKG = DKG1
      MSTG = 1
      KFLAG = 0
      X = 0.0
      TS = T
      T = 0.0
      VX = VXA
      VY = VYA
      NUM = 0
      TH = FN
      Y = ALT(IY)
      YA = Y
1002  IF (ITYPE) 1003,1004,1004
1004  D = DS+SL*U
      GØ TØ 1008
1003  XSTEP = NSTEP
      XNSTEP = XSTEP+FRACT
      D = TS/XNSTEP
1005  IF (ICALC) 1006,1007,1006
1007  D = DMAX
      ICALC = 3
      KFLAG = 1
      GØ TØ 1008
1006  IF (D-DMAX) 1008,1008,1007
1008  AD = A*D
      I = -1
      YØ = Y
      VXØ = VX
      VYØ = VY
      RHØ = 2.37576E-3-Y*(6.87557E-8-Y*6.71618E-13)
1009  V = VX+ABS(VY)/2.0
      VSQ = VX*VX+VY*VY
      V = (V+VSQ/V)/2.0
      V = (V+VSQ/V)/2.0
      CM = V*(8.9544E-4+3.26E-9*Y)+DM
```

```
 1010 IF (CM-CT(1,MSTG)) 1012,1012,1011
 1012 IREG = 1
      GØ TØ 1015
 1011 IF (CM-CT(2,MSTG)) 1013,1013,1014
 1013 IREG = 2
      GØ TØ 1015
 1014 IREG = 3
 1015 CKDG = DKG+CF*(CC(IREG,1,MSTG)+(CC(IREG,2,MSTG)+CC(IREG,3,MSTG)*
     +   CM)*CM)
      HH = TH/V-RHØ*CKDG*V
      AN2 = HH*VX
      AP2 = HH*VY-G
      I = I+1
 1016 IF (I) 1018,1017,1018
 1017 Y = YO+AD*VY
      AP1 = AP2
      AN1 = AN2
      VX = VXO+AD*AN1
      VY = VYO+AD*AP1
      GØ TØ 1009
 1018 T = T+D
      X = X+D*(VXO+AA*(VX-VXO))
      Y = YO+D*(VYO+AA*(VY-VYO))
      VX = VXO+D*(AN1+AA*(AN2-AN1))
      VY = VYO+D*(AP1+AA*(AP2-AP1))
      NUM = NUM+1
 896  FØRMAT (5X,6(F10.2,3X),'KFLAG = ',I1,5X,'AT END ØF RUNKUT'/)
      IF (KBN+1-JBUG) 697,697,699
 697  WRITE (108,896) T,X,Y,VX,VY,D,KFLAG
 699  CØNTINUE
 1019 IF (NLM1-NUM) 1035,1030,1020
 1035 TL = (YT-Y)/VY
 1036 IF (ITYPE) 1037,1038,1037
 1038 KFLAG = 0
      GØ TØ 1040
 1037 IF (KFLAG) 1098,1039,1098
 1039 X = X+TL*VX
 1040 T = T+TL
 1041 GØ TØ 1098
 1030 IF (VY-VYK) 1031,1033,1033
 1033 KFLAG = 1
 1034 GØ TØ 1098
 1031 IF (YA-YT) 1033,1033,1032
 1032 XNLNM = NL-NUM
      D = (YT-Y)/(VY*XNLNM)
      GØ TØ 1006
 1020 IF (VY) 1021,1022,1022
 1022 YA = Y
      GØ TØ 1023
 1021 IF (Y-YT) 1030,1023,1023
 1023 IF (NUM-NSTEP) 1008,1024,1027
 1024 IF (ITYPE-2) 1025,1026,1025
```

```
1025 MSTG = 2
     DKG = DKG2
     DM = DM2
1026 D = 0.0
     TH = 0.0
     CF = CFORM2
1027 IF (NUM-NSTEP*NT) 1029,1028,1008
1029 D = D+DS2
     GO TO 1008
1028 XLNUM = NLM1-NUM
     XLNUM = XLNUM+FRACT
     D = (TS-T)/XLNUM
     GO TO 1005
895  FORMAT (5X,F6.0,2X,F7.0,2X,F4.0,2X,F7.2,2X,F9.1,2X,I4,
    +   3X,'KFLAG = ',I1)
1098 KON = KON+1
     WRITE (108,895) VKTS(IV),ALT(IY),DEG(IANG),T,X,IDNO,KFLAG
999  CONTINUE
     GO TO 100
95   END
```

```
VKTS      ALT      DEG      TF       RANGE      IDNO

450.     3000.    -10.    10.25    7499.9      1    KFLAG = 1
450.     3000.    -10.    10.28    7518.1      1    KFLAG = 0
450.     3000.    -10.    10.28    7518.1      1    KFLAG = 0
450.     3000.    -10.    10.28    7518.1      1    KFLAG = 0

450.     3000.    -10.    10.38    7345.6      2    KFLAG = 1
450.     3000.    -10.    10.46    7401.5      2    KFLAG = 0
450.     3000.    -10.    10.46    7401.5      2    KFLAG = 0
450.     3000.    -10.    10.46    7401.5      2    KFLAG = 0

450.     3000.    -10.    10.30    7439.6      3    KFLAG = 1
450.     3000.    -10.    10.35    7472.5      3    KFLAG = 0
450.     3000.    -10.    10.35    7472.5      3    KFLAG = 0
450.     3000.    -10.    10.35    7472.5      3    KFLAG = 0

450.     3000.    -10.    10.67    7115.2      4    KFLAG = 1
450.     3000.    -10.    10.77    7193.8      4    KFLAG = 0
450.     3000.    -10.    10.77    7193.7      4    KFLAG = 0
450.     3000.    -10.    10.77    7193.7      4    KFLAG = 0

450.     3000.    -10.    10.82    6989.4      5    KFLAG = 1
450.     3000.    -10.    10.93    7089.2      5    KFLAG = 0
450.     3000.    -10.    10.93    7089.2      5    KFLAG = 0
450.     3000.    -10.    10.93    7089.2      5    KFLAG = 0

450.     3000.    -10.    11.00    6832.6      6    KFLAG = 1
450.     3000.    -10.    11.13    6956.0      6    KFLAG = 0
450.     3000.    -10.    11.13    6955.9      6    KFLAG = 0
450.     3000.    -10.    11.13    6956.0      6    KFLAG = 0

450.     3000.    -10.    10.30    7438.2      7    KFLAG = 1
450.     3000.    -10.    10.35    7471.5      7    KFLAG = 0
450.     3000.    -10.    10.35    7471.5      7    KFLAG = 0
450.     3000.    -10.    10.35    7471.5      7    KFLAG = 0

450.     3000.    -10.    10.50    7204.9      8    KFLAG = 1
450.     3000.    -10.    10.62    7295.4      8    KFLAG = 0
450.     3000.    -10.    10.62    7295.4      8    KFLAG = 0
450.     3000.    -10.    10.62    7295.4      8    KFLAG = 0

450.     3000.    -10.    10.27    7471.1      9    KFLAG = 1
450.     3000.    -10.    10.31    7496.3      9    KFLAG = 0
450.     3000.    -10.    10.31    7496.3      9    KFLAG = 0
450.     3000.    -10.    10.31    7496.3      9    KFLAG = 0

450.     3000.    -10.    10.24    7508.3     10    KFLAG = 1
450.     3000.    -10.    10.27    7524.4     10    KFLAG = 0
450.     3000.    -10.    10.27    7524.4     10    KFLAG = 0
```

```
450.     3000.   -10.    10.27     7524.4     10    KFLAG = 0

450.     3000.   -10.    10.24     7518.1     11    KFLAG = 1
450.     3000.   -10.    10.26     7531.8     11    KFLAG = 0
450.     3000.   -10.    10.26     7531.8     11    KFLAG = 0
450.     3000.   -10.    10.26     7531.8     11    KFLAG = 0

450.     3000.   -10.    10.23     7526.8     12    KFLAG = 1
450.     3000.   -10.    10.25     7538.4     12    KFLAG = 0
450.     3000.   -10.    10.25     7538.4     12    KFLAG = 0
450.     3000.   -10.    10.25     7538.4     12    KFLAG = 0

450.     3000.   -10.    10.22     7540.6     13    KFLAG = 1
450.     3000.   -10.    10.23     7548.8     13    KFLAG = 0
450.     3000.   -10.    10.23     7548.8     13    KFLAG = 0
450.     3000.   -10.    10.23     7548.8     13    KFLAG = 0

450.     3000.   -10.    10.29     7453.0     14    KFLAG = 1
450.     3000.   -10.    10.33     7482.6     14    KFLAG = 0
450.     3000.   -10.    10.33     7482.6     14    KFLAG = 0
450.     3000.   -10.    10.33     7482.6     14    KFLAG = 0

450.     3000.   -10.    10.35     7378.3     15    KFLAG = 1
450.     3000.   -10.    10.42     7426.2     15    KFLAG = 0
450.     3000.   -10.    10.42     7426.2     15    KFLAG = 0
450.     3000.   -10.    10.42     7426.2     15    KFLAG = 0

450.     3000.   -10.    10.25     7501.0     16    KFLAG = 1
450.     3000.   -10.    10.28     7518.9     16    KFLAG = 0
450.     3000.   -10.    10.28     7518.9     16    KFLAG = 0
450.     3000.   -10.    10.28     7518.9     16    KFLAG = 0

450.     3000.   -10.    10.42     7303.6     17    KFLAG = 1
450.     3000.   -10.    10.50     7369.8     17    KFLAG = 0
450.     3000.   -10.    10.50     7369.8     17    KFLAG = 0
450.     3000.   -10.    10.50     7369.8     17    KFLAG = 0

450.     3000.   -10.    16.97     3488.5     18    KFLAG = 1
450.     3000.   -10.    16.43     3852.1     18    KFLAG = 0
450.     3000.   -10.    16.44     3854.9     18    KFLAG = 0
450.     3000.   -10.    16.44     3854.9     18    KFLAG = 0

450.     3000.   -10.    21.27     4489.1     19    KFLAG = 1
450.     3000.   -10.    20.61     4620.6     19    KFLAG = 0
450.     3000.   -10.    20.59     4630.3     19    KFLAG = 0
450.     3000.   -10.    20.59     4630.5     19    KFLAG = 0

450.     3000.   -10.    15.05     5668.6     20    KFLAG = 1
450.     3000.   -10.    14.73     5927.9     20    KFLAG = 0
450.     3000.   -10.    14.73     5928.9     20    KFLAG = 0
450.     3000.   -10.    14.73     5928.9     20    KFLAG = 0
```

```
450.    3000.    -10.    16.71     5737.0    21    KFLAG = 1
450.    3000.    -10.    15.82     6048.7    21    KFLAG = 0
450.    3000.    -10.    15.82     6050.6    21    KFLAG = 0
450.    3000.    -10.    15.82     6050.6    21    KFLAG = 0

450.    3000.    -10.    18.65     2942.4    22    KFLAG = 1
450.    3000.    -10.    17.78     3295.4    22    KFLAG = 0
450.    3000.    -10.    17.79     3302.7    22    KFLAG = 0
450.    3000.    -10.    17.79     3302.6    22    KFLAG = 0

450.    3000.    -10.    10.58    11152.4    23    KFLAG = 1
450.    3000.    -10.    10.59    10939.5    23    KFLAG = 0
450.    3000.    -10.    10.59    10939.9    23    KFLAG = 0
450.    3000.    -10.    10.59    10939.9    23    KFLAG = 0

450.    3000.    -10.     8.48    11968.8    24    KFLAG = 1
450.    3000.    -10.     7.36    12812.9    24    KFLAG = 0
450.    3000.    -10.     7.34    12826.1    24    KFLAG = 0
450.    3000.    -10.     7.34    12826.3    24    KFLAG = 0

450.    3000.    -10.    31.40     1661.3    25    KFLAG = 0
450.    3000.    -10.    31.40     1661.3    25    KFLAG = 0
450.    3000.    -10.    31.40     1661.3    25    KFLAG = 0
450.    3000.    -10.    31.40     1661.3    25    KFLAG = 0

450.    3000.    -10.    35.71     1381.6    26    KFLAG = 0
450.    3000.    -10.    35.71     1381.6    26    KFLAG = 0
450.    3000.    -10.    35.71     1381.6    26    KFLAG = 0
450.    3000.    -10.    35.71     1381.6    26    KFLAG = 0

450.    3000.    -10.    40.87     1214.0    27    KFLAG = 0
450.    3000.    -10.    40.87     1214.0    27    KFLAG = 0
450.    3000.    -10.    40.87     1214.0    27    KFLAG = 0
450.    3000.    -10.    40.87     1214.0    27    KFLAG = 0

450.    3000.    -10.    21.51     2061.9    28    KFLAG = 1
450.    3000.    -10.    20.49     2441.9    28    KFLAG = 0
450.    3000.    -10.    20.50     2449.7    28    KFLAG = 0
450.    3000.    -10.    20.50     2449.6    28    KFLAG = 0
```

TABLE 3. Sample Errors in Impact Range Calculations for the General-Purpose Bomb Mk 82/Snakeye/Unretarded.

Release conditions			Impact range, ft		Impact range error, ft
Velocity, knots	Altitude, ft	Angle, deg	Ballistic tables NAVAIR 01-1C-1T[a]	Algorithm	
400	2,500	0	8,039	8,040	1
	5,000	-10	9,158	9,159	1
	12,000	-30	10,536	10,538	2
	12,000	-45	7,530	7,531	1
450	2,500	0	8,998	8,996	-2
	5,000	-10	10,025	10,024	-1
	12,000	-30	11,340	11,341	1
	12,000	-45	7,998	7,999	1
500	2,500	0	9,946	9,942	-4
	5,000	-10	10,843	10,840	-3
	12,000	-30	12,067	12,068	1
	12,000	-45	8,406	8,406	0

[a] Reference 7.

TABLE 4. Sample Errors in Impact Range Calculations for the Cluster Bomb Mk 20 Mod 2 (Rockeye II) With a Fuze Setting of 4.0 sec.

Release Conditions			Impact range, ft		Impact range error, ft
Velocity, knots	Altitude, ft	Angle, deg	NWL ballistic table 183[a]	Algorithm	
400	2,500	-10	5,197	5,198	1
	4,000	-20	5,241	5,241	0
	5,000	-30	4,842	4,842	0
	8,000	-45	4,268	4,269	1
450	2,500	-10	5,609	5,599	-10
	4,000	-20	5,612	5,611	-1
	5,000	-30	5,158	5,154	-4
	8,000	-45	4,548	4,561	13
500	2,500	-10	5,966	5,962	-4
	4,000	-20	5,948	5,950	2
	5,000	-30	5,456	5,459	3
	8,000	-45	4,828	4,838	10

[a] Reference 8.

TABLE 5. Sample Errors in Impact Range Calculations for the General-Purpose Bomb Mk 82/Snakeye/Retarded.

Release conditions			Impact range, ft		Impact range error, ft
Velocity, knots	Altitude, ft	Angle, deg	Ballistic tables NAVAIR 01-1C-1T	Algorithm	
400	1,000	0	2,060	2,054	-6
	1,000	-10	2,534	2,525	-9
	1,500	-10	2,972	2,967	-5
	2,500	-20	3,072	3,069	-3
450	1,000	0	3,242	3,244	2
	1,000	-10	2,669	2,663	-6
	1,500	-10	3,128	3,130	2
	2,500	-20	3,219	3,223	4
500	1,000	0	3,401	3,412	11
	1,000	-10	2,786	2,784	-2
	1,500	-10	3,263	3,272	9
	2,500	-20	3,346	3,357	11

REFERENCES

1. The Boeing Company. *Algorithm for Fire Control*, by R. M. Toms, S. Onyshko, R. F. Etter, and F. Hamilton. Report for Naval Weapons Center Contract N00123-69-C-1344, D162-10026-1, Seattle, Wash., BC, 1969.

2. ------. *Fire-Control Algorithm Applications - Final Report*, by R. M. Toms, S. Onyshko, F. Hamilton, and L. A. Hanvey. Report for Naval Weapons Center Contract N00123-70-C-0829, Seattle, Wash., BC, 1970.

3. Naval Ordnance Test Station. *Ballistic Handbook*, by R. B. Seeley, and R. D. Cole. China Lake, Calif., NOTS, August 1965. (NOTS TP 3902.)

4. Ceschino, F., and J. Kuntzmann. *Numerical Solution of Initial Value Problems*, Englewood Cliffs, N. J., Prentice-Hall, 1966. 318 pp.

5. Ralston, Anthony. "Runge-Kutta Methods with Minimum Error Bounds," MATH COMPUT, Vol. 16 (1962), p. 431-437.

6. Hildebrand, F. B. *Introduction to Numerical Analysis*, New York, McGraw-Hill, 1956. 490 pp.

7. Office of the Chief of Naval Operations. *Tactical Manual Ballistics Tables*, prepared by the Naval Weapons Laboratory. Washington, D.C., CNO, August 1970. (NAVAIR 01-1C-1T.)

8. Naval Weapons Laboratory. *NWL Ballistic Table Number 183 (Interim) for Gunsight Delivery of the Cluster Bomb Mk 20 Mod 2 (Rockeye II), Zero Ejection Velocity*. Dahlgren, Va., NWL, June 1968.

INITIAL DISTRIBUTION

1 Director of Navy Laboratories (DNL)

48 Naval Air Systems Command

AIR-03 (1)
AIR-03B (1)
AIR-04 (1)
AIR-05 (1)
AIR-05A (1)
AIR-3021 (1)
AIR-3023 (1)
AIR-3031 (1)
AIR-3033 (1)
AIR-350 (1)
AIR-360 (1)
AIR-360C (1)
AIR-503 (1)
AIR-503A (1)
AIR-503B (1)
AIR-503E (1)
AIR-5031 (1)
AIR-5034 (1)
AIR-506 (1)
AIR-510 (1)
AIR-510C (1)
AIR-5102 (1)
AIR-5102G (1)
AIR-5103 (1)
AIR-5103F (1)
AIR-5103H (1)
AIR-5202 (1)
AIR-5205 (1)
AIR-532 (1)
AIR-532A (1)
AIR-532B (1)
AIR-5321 (1)
AIR-533 (1)
AIR-533B (1)
AIR-533F3 (1)
AIR-533F3B (1)
AIR-533F3F (1)
AIR-5333 (1)
AIR-5333A (1)
AIR-533C (1)
AIR-5333C3 (1)
AIR-6022 (1)
AIR-604 (2)
PMA-235 (1)
PMA-235A (1)
PMA-246 (1)
PMA-257 (1)

18 Chief of Naval Operations

OP-03 (1)
OP-03EG (1)
OP-05 (1)
OP-05W (1)
OP-07 (1)
OP-07TB (1)
OP-07TH (1)
OP-342 (1)
OP-342F (1)
OP-506 (1)
OP-506C2 (1)
OP-506C5 (1)
OP-522 (1)
OP-72 (1)
OP-72C (1)
OP-722 (1)
OP-722E (1)
OP-96 (1)

3 Chief of Naval Material

MAT-03 (1)
MAT-3L1 (1)
MAT-03L2 (1)

3 Naval Ordnance Systems Command
  ORD-0632 (2)
  ORD-932 (1)
1 Naval Ship Systems Command
2 Chief of Naval Research, Arlington
  ONR-104 (1)
3 Commandant of the Marine Corps
  Code AAJ-1C (1)
  Code AAW (1)
  Code AX (1)
2 Marine Corps Development and Education Command, Quantico
  Combat Services Support Division, Marine Corps Landing Force Development Center (1)
  War Games Division, Marine Corps Landing Force Development Center (1)
1 Commander-in-Chief, Pacific Fleet (3rd Division)
1 Air Test and Evaluation Squadron 5 (Conventional Weapons Project Officer)
14 Attack Carrier Air Wings (one each)
  1, 2, 3, 5, 6, 7, 9, 11, 14, 15, 16, 17, 19, 21
3 Attack Squadrons (one each)
  42, 122, 174
8 Marine Aircraft Groups (one each)
  12, 13, 14, 15, 24, 31, 32, 33
1 1st Marine Aircraft Wing
1 3rd Marine Aircraft Wing
1 Marine Attack Squadron 331
1 Marine Corps Air Station, Beaufort
1 Marine Corps Section, Army Command and General Staff College, Fort Leavenworth
2 Marine Training Squadrons (one each)
  103, 203
2 Naval Air Development Center, Johnsville
  Code ED-81A (1)
  Code WR-2 (1)
1 Naval Air Engineering Facility (SI), Naval Air Engineering Center, Philadelphia
2 Naval Air Force Atlantic Fleet
  Code 335 (1)
  Code 524 (1)
1 Naval Air Force, Pacific Fleet
2 Naval Air Station (one each)
  Jacksonville, North Island
1 Naval Air Test Center, Patuxent River
1 Naval Aviation Engineering Service Unit, Philadelphia (Technical Library)
1 Naval Avionics Facility, Indianapolis (Code MAL, Technical Library)
3 Naval Missile Center, Point Mugu
  Code 5240 (1)
  Code 5301-4 (1)
  Code 5320 (1)

List continued on inside back cover.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Weapons Center China Lake, Calif. 93555	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE
A BALLISTIC TRAJECTORY ALGORITHM FOR DIGITAL AIRBORNE FIRE CONTROL

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*
Arthur A. Duke, Thomas H. Brown, Kenneth W. Burke, and Richard B. Seeley

6. REPORT DATE September 1972	7a. TOTAL NO. OF PAGES 54	7b. NO. OF REFS 6

8a. CONTRACT OR GRANT NO. N00123-69-C-1344 N00123-70-C-0829	9a. ORIGINATOR'S REPORT NUMBER(S) NWC TP 5416
b. PROJECT NO. AirTask A365-33348/216-1/S-171-00-00	
c. AirTask A510-5103-216-2/1235-000-143	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT
Distribution limited to U. S. Government agencies only; test and evaluation; 17 August 1972. Other requests for this document must be referred to the Naval Weapons Center.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Chief of Naval Operations Washington, D. C. 20360

13. ABSTRACT

A method of numerical integration of the equations of motion of unguided air-to-surface weapons is developed. This algorithm, suitable for real-time solution by airborne digital computers, yields accurate trajectory parameters, provides great flexibility in release condition and weapon type, and minimizes computer memory requirements.

UNCLASSIFIED
Security Classification

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Airborne computer						
Ballistics						
Bombing						
Fire control						
Trajectory integration						
Weapon release						

DD FORM 1 NOV 65 1473 (BACK)
(PAGE 2)

UNCLASSIFIED
Security Classification

3 Naval Ordnance Laboratory, White Oak
  Code 0431 (1)
  Code 412 (1)
  Code 433 (1)
1 Naval Ordnance Station, Indian Head
1 Naval Ordnance Station, Louisville (Code R)
1 Naval Training Device Center, Orlando (Technical Library)
2 Naval War College, Newport
  Director of Libraries (1)
5 Naval Weapons Laboratory, Dahlgren
  Code KE (1)
  Code KRT (1)
  Code T (1)
  Code WWB (1)
  Code WX (1)
1 Operational Test and Evaluation Force (Air Warfare Division, Light Attack Assistant)
1 Army Combat Developments Command, Aviation Agency, Fort Rucker
1 Aberdeen Proving Ground (Technical Library)
1 Army Aviation Test Board, Fort Rucker
1 Army Materiel Systems Analysis Agency, Aberdeen Proving Ground
6 Headquarters, U. S. Air Force
  AFCSAMI (1)
  AFGOA (1)
  AFRDC (3)
  AFSPD (1)
1 Air Force Logistics Command, Wright-Patterson Air Force Base (MCMTM)
1 Air Force Systems Command, Andrews Air Force Base
2 Tactical Air Command, Langley Air Force Base
  DMA (1)
  DORQ-M (1)
2 Aeronautical Systems Division, Wright-Patterson Air Force Base
  ASJ (1)
  ASNNS (1)
1 Air Force Avionics Laboratory, Wright-Patterson Air Force Base
1 Air University Library, Maxwell Air Force Base (Document Library)
2 Armament Development & Test Center, Eglin Air Force Base
1 Tactical Fighter Weapons Center, Nellis Air Force Base (COA)
1 Wright-Patterson Air Force Base (Director, F-111-SPO, Armament Division)
1 Advanced Research Projects Agency, R&D Field Unit
2 Defense Documentation Center
1 Weapons Systems Evaluation Group
1 AC Electronics Division, General Motors Corporation, Milwaukee (Military Avionics System)
1 Autonetics, A Division of North American Rockwell Corporation, Anaheim, Calif.

2 Center for Naval Analyses, University of Röchester, Arlington
   Dr. Bothwell (1)
   Mr. Dibona (1)
1 Convair Division of General Dynamics, San Diego
1 General Electric Company, Binghampton, N. Y.
1 Grumman Aerospace Corporation, Bethpage, N. Y.
1 Hughes Aircraft Company, Culver City, Calif.
1 International Business Machines Corporation, Oswego, N. Y. (Electronics System Center)
1 Lear Incorporated, Grand Rapids, Mich. (Instrument Division)
1 Ling-Temco-Vought, Inc., Vought Aeronautics Division, Dallas
1 Litton Systems Inc., Woodland Hills, Calif. (Guidance and Control Division)
1 Lockheed-California Company, Burbank
1 McDonnell Douglas Corporation, St. Louis
1 North American Rockwell Corporation, Columbus, Ohio (Autonetics Division)
2 Northrop Corporation, Norair Division, Hawthorne, Calif.
   Electronics Division (1)
1 Nortronics, Anaheim (Electro-Mechanics Division)
1 Sperry Gyroscope Company, Great Neck, N. Y. (ILAAS Group)
1 The Rand Corporation, Santa Monica, Calif.
1 Westinghouse Defense & Space Center, Baltimore, Md.