AN EFFICIENT COVARIANCE MATRIX IMPLEMENTATION FOR LARGE-SCALE SYSTEMS

By

VIJAYENDRA MOHAN GUPTA

Bachelor of Engineering (Honours)

Birla Institute of Technology & Science

Pilani, Rajasthan, India

1971

Submitted to the Faculty of the Graduate College
of the Oklahoma State University
in partial fulfillment of the requirements
for the Degree of
MASTER OF SCIENCE
May, 1973

AN EFFICIENT COVARIANCE MATRIX IMPLEMENTATION

FOR LARGE-SCALE SYSTEMS

Thesis Approved:

James R. Rowland
Thesis Adviser

Paul A. McCollum

Edward L. Shreve

D. D. Durham
Dean of the Graduate College

ACKNOWLEDGMENT

I wish to express my sincere appreciation to Dr. James R. Rowland for spending many hours in valuable guidance on this thesis research. I would like to thank him for being my Thesis Advisor and Chairman of my Committee. I wish to acknowledge greatly the other members of my committee, Professor Paul A. McCollum and Dr. Edward L. Shreve, for their suggestions and teaching excellence.

I also wish to acknowledge support as Research Assistant derived from Contract No. DAAH01-72-C-0672 with the U. S. Army Missile Command, Redstone Arsenal, Alabama.

I would like to thank Mrs. Mary Griggs for her excellent typing of this thesis.

Finally, I would like to thank my parents, brother, and friends for their encouragement.

TABLE OF CONTENTS

Chapter | Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Computer software packages have proven to be very useful for the application of sophisticated analysis and design algorithms for industrial problems. Their usefulness in providing powerful results in an easily applied form for the user has led to the development of efficient software packages for large-scale systems. One problem area in which software packages are becoming more popular involves those systems having inherent noise problems resulting from random variations in disturbance inputs and/or system parameters. These random variations result in errors being propagated throughout the large-scale systems. A thorough knowledge of the large-scale system dynamics, statistical properties of dynamical systems, and some simulation experience is necessary for the development of computer software packages for these applications. In this work a direct algorithm is implemented to yield a computer software package for determining the propagation of errors due to noise in large-scale missile systems.

Background

Previous work on noise propagation problems has focused on the use of the Monte Carlo technique in which large numbers of runs are ensemble-averaged to obtain statistical results. Primary considerations in the use of this traditional approach are the generation of prespecified statistical inputs and the simulation of dynamical systems. A more modern approach based on computing the state covariance matrix directly has become popular in recent years. This new approach, referred to as the direct covariance algorithm, has been applied for an approximate analysis of large-scale nonlinear systems. The development of a computer software package using the direct covariance algorithm would greatly enhance large-scale system analysis capabilities.

The Monte Carlo method uses repeated sample functions as inputs to the model of a mathematical or physical process. Earlier noise propagation studies by the Monte Carlo method were based on the use of analog noise generators. Due to the fact that these generators were not repetitive, the analog approach became unpopular after the recent development of digital pseudo-random number generators. These generators could be used to generate the same numbers as many times as desired and, thus, ease the work of debugging the simulated program. Large amounts of simulated random data are required for acceptable results. For the digital implementation of the Monte Carlo technique, pseudo-random numbers are either drawn from tables (1) or generated from simple relationships within the computer. For the former case the random numbers must be stored and used whenever required. However, for the latter case Chambers (2), Hull and Dobell (3), MacLaren and Marsaglia (4), and Gelder (5) have developed mixed congruential and multiplicative recurrence formulas for generating pseudo-random numbers. The numbers generated are uniformly distributed on the interval (0,1). The uniformly distributed numbers may be converted into zero-mean, unity-variance, Gaussianly distributed random numbers by an exact closed-form expression developed by Box and Muller (6). An alternate, but approximate, method of converting the uniform sequence to a Gaussian sequence utilizes the Central Limit theorem which states that as the number of statistically independent variables is increased without limit, a Gaussian probability distribution is approached for the sum, regardless of the probability distributions of the various variables.

A direct technique (7-12) has resulted from the error covariance matrix propagation in the Kalman filtering equation (13,14). Though exact for linear time-varying systems, the direct covariance algorithm has also been applied for mildly non-linear systems. For example, this technique has been used by Kuhnel and Sage (15) for sensitivity equations about a nominal flight path due to trajectory initial condition dispersions and random system variations. They used a thirty-third order, six degree-of-freedom homing missile model to illustrate the application to a realistic situation. Kuhnel and Sage used only the adjoint method whereas Irwin and Hung (16) applied both direct and adjoint methods for evaluating the state covariance algorithm for large-scale, nonlinear, dynamical systems. An interval-by-interval linearization procedure has also been proposed (17,18). For nonlinear feedback systems, the direct covariance approach has been used by Brown (19-21) for solving trajectory optimization problems. Using a more accurate algorithm about a nominal trajectory, Clark (22, 23) has developed related results.

Rowland and Holmes (24) have shown that the direct covariance technique is more accurate and faster than the Monte Carlo approach. They demonstrated that the direct covariance algorithm can be applied to mildly nonlinear systems with acceptable results by using linearized incremental equations about the noise-free solution. The objective of this research is to develop a computer software package for the efficient implementation of the direct covariance algorithm.

Derivation of the Direct Covariance Algorithm

Consider the linear, time-varying, dynamical system represented by the vector differential equation

$$\dot{\underline{x}}(t) = A(t)\underline{x}(t) + B(t)\underline{w}(t) \tag{1.1}$$

where $\underline{x}(t)$ is an n-dimensional state vector, A(t) is an n by n matrix, B(t) is an n by m matrix, and $\underline{w}(t)$ is an m-dimensional input noise vector.

The covariance matrix of the state vector is defined as

$$P(t) \triangleq E\{\underline{x}(t)\underline{x}^T(t)\} \tag{1.2}$$

The elements of the input noise vector are zero-mean white noise processes, and their covariance matrix is represented by

$$E\{\underline{w}(t)\underline{w}^T(\tau)\} = Q_{\underline{w}}(t)\ \delta(t-\tau) \tag{1.3}$$

where $\delta(\cdot)$ is the impulse function. The m by m covariance matrix $Q_{\underline{w}}(t)$ may be time-varying in general.

The covariance matrix P(t) may be determined directly in terms of A(t), B(t), and $Q_{\underline{w}}(t)$ by using $\underline{x}(t)$ in (1.2). The solution of the time-varying, linear differential equation given by (1.1) is

$$\underline{x}(t) = \Phi(t,t_0)\ \underline{x}(t_0) + \int_{t_0}^{t} \Phi(t,\tau)\ B(\tau)\ \underline{w}(\tau)d\tau \tag{1.4}$$

Therefore, the covariance matrix of the state vector $\underline{x}(t)$ may be calculated as

$$P(t) = E\{\underline{x}(t)\underline{x}^T(t)\}$$

$$= E[\{\Phi(t,t_o)\ \underline{x}(t_o) + \int_{t_o}^{t} \Phi(t,\tau)\ B(\tau)\ \underline{w}(\tau)d\tau\}$$

$$\cdot \{\Phi(t,t_o)\underline{x}(t_o) + \int_{t_o}^{t} \Phi(t,\tau)\ B(\tau)\ \underline{w}(\tau)d\tau\}^T] \qquad (1.5)$$

Since $\underline{x}(t_o)$ and $\underline{w}(t)$ are uncorrelated for all $t>t_o$,

$$P(t) = E[\Phi(t,t_o)\ \underline{x}(t_o)\ \{\Phi(t,t_o)\ x(t_o)\}^T +$$

$$\int_{t_o}^{t} \int_{t_o}^{t} \Phi(t,\tau_1)\ B(\tau_1)\ \underline{w}(t_1)\{\Phi(t,\tau_2)B(\tau_2)\underline{w}(\tau_2)\}^T d\tau_1 d\tau_2]$$

$$= \Phi(t,t_o)\ E\{\underline{x}(t_o)\underline{x}^T(t_o)\}\ \Phi^T(t,t_o)$$

$$\int_{t_o}^{t} \int_{t_o}^{t} \Phi(t,\tau_1)\ B(\tau_1)\ E\{\underline{w}(\tau_1)\underline{w}^T(\tau_2)\}\ B^T(\tau_2)\ \Phi^T(t,\tau_2)d\tau_1 d\tau_2 \quad (1.6)$$

Using (1.3) and the sifting property of the delta function, (1.6) reduces to

$$P(t) = \Phi(t,t_o)\ P(t_o)\ \Phi^T(t,t_o) +$$

$$\int_{t_o}^{t} \Phi(t,\tau_1)\ B(\tau_1)\ Q_{\underline{w}}(\tau_1)\ B^T(\tau_1)\ \Phi^T(t,\tau_1)d\tau_1 \qquad (1.7)$$

The integral equation in (1.7) may be expressed more conveniently as a matrix differential equation for P(t). In establishing this form, the state transition matrix $\Phi(t,t_o)$ is identified as the solution of the homogeneous linear differential equation

$$\dot{\Phi}(t,t_o) = \frac{d}{dt}\Phi(t,t_o) = A\Phi(t,t_o) \qquad (1.8)$$

with the boundary condition $\Phi(t_o,t_o) = I$. Using the relationship in (1.8) to simplify (1.7) gives

$$\dot{P}(t) = \dot{\Phi}(t,t_0)\, P(t_0)\, \Phi^T(t,t_0) + \Phi(t,t_0)\, P(t_0)\, \dot{\Phi}^T(t,t_0)$$

$$+ \int_{t_0}^{t} \frac{\partial \Phi(t,\tau)}{\partial t} B(\tau)\, Q_{\underline{w}}(\tau_1)\, B^T(\tau_1)\, \Phi^T(t,\tau_1) d\tau_1$$

$$+ \int_{t_0}^{t} \Phi(t,\tau_1)\, B(\tau_1)\, Q_{\underline{w}}(\tau_1)\, B^T(\tau_1)\, \frac{\partial \Phi^T(t,\tau)}{\partial t} d\tau_1$$

$$+ \Phi(t,t)\, B(t)\, Q_{\underline{w}}(t)\, B^T(t)\, \Phi^T(t,t)$$

$$\dot{P}(t) = A(t)\, [\Phi(t,t_0)\, P(t_0)\, \Phi^T(t,t_0)$$

$$+ \int_{t_0}^{t} \Phi(t,\tau_1)\, B(\tau_1)\, Q_{\underline{w}}(\tau_1)\, B^T(\tau_1)\, \Phi^T(t,\tau_1) d\tau_1]$$

$$+ [\Phi(t,t_0)\, P(t_0)\, \Phi^T(t,t_0)$$

$$+ \int_{t_0}^{t} \Phi(t,\tau_1) B(\tau_1) Q_{\underline{w}}(\tau_1) B^T(\tau_1) \Phi^T(t,\tau_1) d\tau_1]^T A^T(t)$$

$$+ B(t)\, Q_{\underline{w}}(t)\, B^T(t) \qquad (1.9)$$

where $\Phi(t,t)$ has been replaced by the identity matrix I. Therefore,

$$\dot{P}(t) = A(t)\, P(t) + P(t)\, A^T(t) + B(t)\, Q_{\underline{w}}(t)\, B^T(t) \qquad (1.10)$$

The desired result in (1.10) yields P(t) by solving a set of linear differential equations.

Criteria for Comparison

Since the most efficient technique is sought for the study of noise propagation in large-scale systems, the criteria for comparison between the Monte Carlo technique and the direct covariance algorithm play an important role in selecting the most suitable technique. Some of these criteria are discussed in the following paragraphs.

Information Provided

The primary consideration for choosing a simulation technique is greatly influenced by the information provided by that technique. The Monte Carlo technique provides the complete probability density function associated with random phenomena, whereas the direct covariance technique only gives the variance about the nominal trajectory, which serves as the mean value. In many applications of interest, the mean and variance of selected states is all the information that is required for an acceptable analysis of system behavior.

Accuracy

The next criterion for comparison is the accuracy level provided, which varies with different techniques. The direct covariance algorithm gives exact results for linear systems and may be applied to yield acceptable results for mildly nonlinear systems. On the other hand, the results of 25 to 50 Monte Carlo runs may not provide acceptable accuracy, although a high accuracy may be expected with 1000 Monte Carlo runs (24). The step size chosen for integration may be used as a control for the tradeoff between accuracy and computational time.

Computer Storage

The computer software package efficiency may also be judged by the computer storage needed for the application of various techniques. The direct covariance algorithm requires somewhat more storage as compared to the Monte Carlo technique. The amount of additional storage depends upon the order of the system being considered as shown in later chapters.

Computational Time

Another objective of an efficient computer software package is to obtain a computationally fast algorithm. The speed and accuracy may be examined with respect to tradeoff possibilities. For extremely accurate results, the computational time needed may be quite large. By the use of large integration step sizes, the computational speed may be increased. There are many approximate techniques which may be used to reduce the computation time. For example, slowly time-varying coefficients may be replaced by constant coefficients and very small variables and coefficients may be replaced by zero. Moreover, if the order of the system can be reduced, a considerable savings in computer time might be realized.

Program Complexity

The computer software package should be simple so that anyone with only limited simulation experience is able to understand it. Due to the inverse relation of the complexity and computation time, the tradeoff between them is possible. With maximum complexity the computer time may be reduced by as much as a factor of ten in certain applications.

Possibilities of Extension

The general computer software package for the direct covariance algorithm is a fundamental step in the subsequent development of an efficient software package for Kalman filtering as a practical estimation algorithm. Furthermore, many approximate nonlinear filtering algorithms are based on similar considerations.

Outline

Following this introductory chapter, the direct covariance algorithm is extended in Chapter II for application to nonlinear systems. In addition, several Monte Carlo tests are performed to determine a suitable discretization procedure for subsequent use in validating the results of the digital computer software package. The software package development and its application to a large-scale missile system are described in Chapter III. Engineering tradeoff studies for the direct covariance algorithm between accuracy, computational speed, computer storage, and program complexity are performed in Chapter IV. Conclusions and recommendations are presented in Chapter V.

CHAPTER II

DIRECT COVARIANCE ALGORITHM EXTENSIONS AND MONTE CARLO TESTING

This chapter defines the general mathematical system under consideration and extends the direct covariance algorithm for this nonlinear case. Numerical results are presented for a second-order nonlinear system to demonstrate the applicability of the algorithm. Thereafter, the problem of modeling continuous white noise inputs on the digital computer is investigated from a more general viewpoint than considered previously. Three modeling representations are presented and then compared on a second-order system. The best of these discretization procedures is used in subsequent chapters to compare the Monte Carlo technique with the direct covariance algorithm on a thirty-first order math model of a six degree-of-freedom air defense missile system.

Mathematical Formulation

Consider the nonlinear, time-varying, dynamical system represented by the vector differential equation

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{w}, t) \qquad (2.1)$$

where $\underline{x}$ is the n-dimensional vector of system variables, $\underline{w}$ is an m-dimensional input noise vector, and t is the independent variable representing time.

The input noise vector $\underline{w}(t)$ has a mean value specified by the m-dimensional vector $\eta_{\underline{w}}(t)$ and a covariance matrix $Q_{\underline{w}}(t)$, which is m by m in dimension. These quantities may be defined mathematically as

$$E\{\underline{w}(t)\} \triangleq \eta_{\underline{w}}(t)$$

$$E\{[\underline{w}(t) - \eta_{\underline{w}}(t)] [\underline{w}(\tau) - \eta_{\underline{w}}(\tau)]^T\} \triangleq Q_{\underline{w}}(t)\ \delta(t-\tau) \tag{2.2}$$

where $\delta(\cdot)$ is the impulse function.

The covariance matrix of the state $\underline{x}(t)$ is defined as

$$P(t) \triangleq E\{[\underline{x}(t) - \eta_{\underline{x}}(t)] [x(\tau) - \eta_{\underline{x}}(\tau)]^T\} \tag{2.3}$$

where $\eta_{\underline{x}}(t)$ is the mean of $\underline{x}(t)$. The problem is to determine P(t) in terms of the mathematical description of the nonlinear system in (2.1) and the properties of the input noise vector given in (2.2).

An Approximate Covariance Analysis of Nonlinear Systems

The application of the direct covariance algorithm developed in Chapter I to the nonlinear system in (2.1) can be achieved as an approximate analysis. Let $\underline{x}_N(t)$ denote the noise-free nominal trajectory obtained by replacing $\underline{w}(t)$ by $\eta_{\underline{w}}(t)$ in (2.1). It is assumed that the input noise disturbances cause sufficiently small deviations about this nominal solution such that $\eta_{\underline{x}}(t) = \underline{x}_N(t)$. Let these small deviations $\underline{\delta x}(t)$ be defined by

$$\underline{\delta x}(t) \triangleq \underline{x}(t) - \underline{x}_N(t) \tag{2.4}$$

Expanding (2.1) in a Taylor's series about $\underline{x}_N(t)$ yields

$$\delta\dot{x}(t) = A(t)\ \underline{\delta x}(t) + B(t)\underline{w}(t) \tag{2.5}$$

where

$$A(t) \triangleq \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_{\substack{\underline{x}(t) = \underline{x}_N(t) \\ \underline{w}(t) = \underline{\eta}_w(t)}}$$

$$B(t) \triangleq \left.\frac{\partial \underline{f}}{\partial \underline{w}}\right|_{\substack{\underline{x}(t) = \underline{x}_N(t) \\ \underline{w}(t) = \underline{\eta}_w(t)}} \tag{2.6}$$

The approximation made in (2.5) is that the second and all higher-order terms in $\delta \underline{x}$ are negligible when compared to the linear terms. This approximation is valid if the $\delta \underline{x}$ variations are sufficiently small.

To demonstrate the importance of this approximation, consider the second-order nonlinear system investigated in (24). The system is described by

$$\dot{x}_1 = -2x_1 + x_2 + \gamma x_2^2 \text{ sign } (x_2)$$
$$\dot{x}_2 = -x_2 + w(t) \tag{2.7}$$

where w(t) is a zero-mean Gaussian white noise process applied for all $t \geq 0$. Figure 1 shows the results obtained in (24) by applying the direct covariance algorithm as the input covariance Q_w was increased from 0.01 to 5. As Q_w was increased, the higher-order δx variations in (2.5) became significant and larger errors were obtained. Therefore, the arbitrary application of the direct covariance algorithm to nonlinear systems with severe nonlinearities and/or extremely high input noise levels must be approached with some caution.

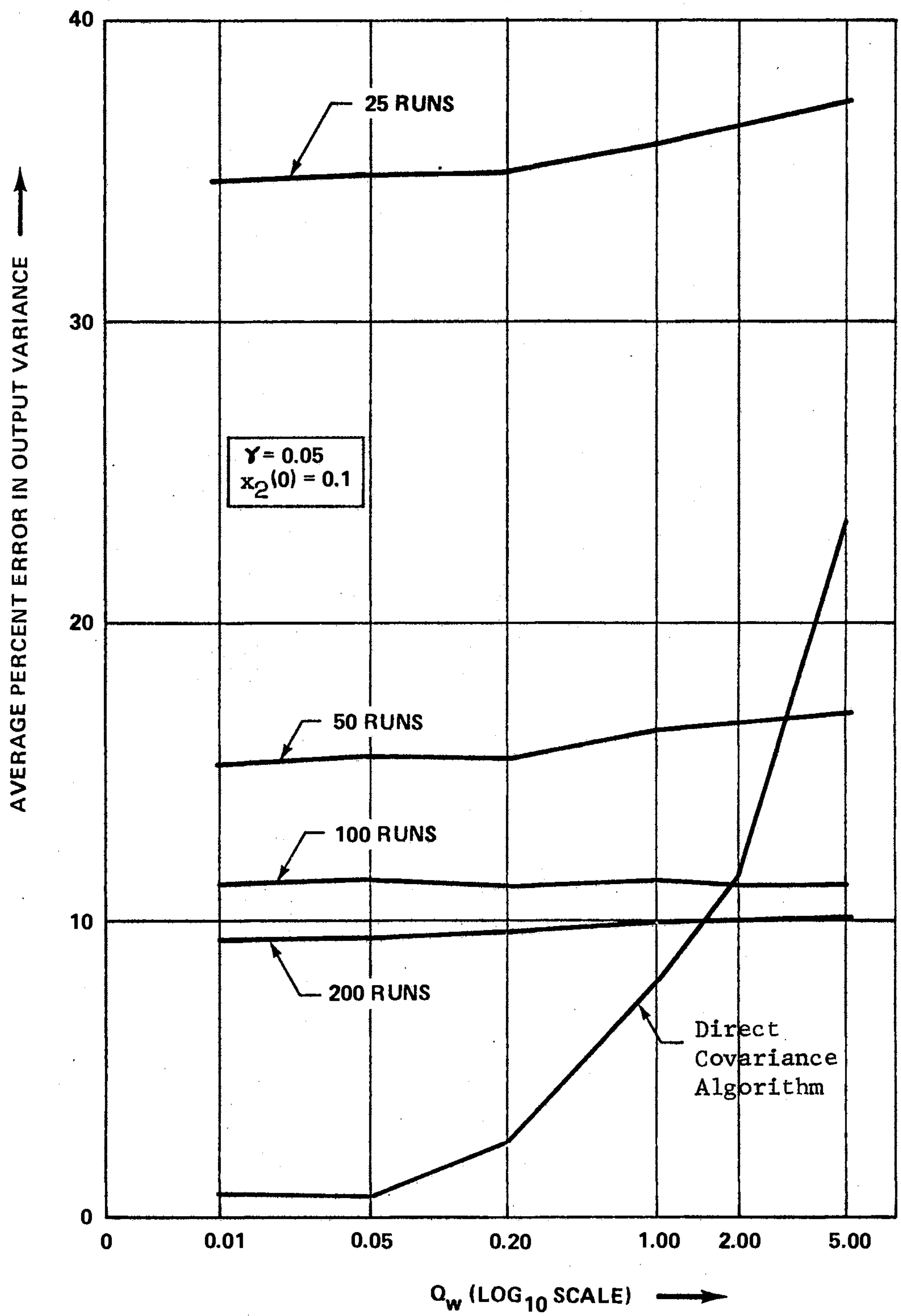

Figure 1. Comparisons Between the Direct Covariance Algorithm and Monte Carlo Simulations for (2.7)

Monte Carlo Testing

To validate the accuracy of the computer software package for the direct covariance algorithm, comparisons were made with the Monte Carlo technique. As a preliminary step, the discretization procedures for white noise inputs were investigated to determine whether improved Monte Carlo results could be obtained. Previous methods were based on the generation of pseudo-random numbers which were then held constant over the discretization interval. The relationships between the covariance matrix $Q_{\underline{w}_d}$ of discrete random sequences and $Q_{\underline{w}}$ defined in (2.2) is given by

$$Q_{\underline{w}_d} = Q_{\underline{w}} / T \qquad (2.8)$$

where T is the discretization interval. An extensive study was performed by Rowland and Holmes (24) on the above method, and some of those results are used here to evaluate new methods for the discrete representation of continuous white noise processes.

A new functional approach to the discretization problem has been developed in this work, and results are compared with the previous method in the next section. Suppose several zero-mean random numbers β_k are combined on each discretization interval to form a power series function of time as

$$w_d(\beta_0, \beta_1, \beta_2, \ldots, \beta_K, t) = \sum_{k=0}^{K} \beta_k t^k \quad \text{for } 0 < t < T \qquad (2.9)$$

The autocorrelation function of such a train of pulses is given in (25, 26) by

$$R_{w_d w_d}(t, t+\tau) = \begin{cases} \sum_{k=0}^{K} Q_{\beta_k} t^{2k} \left(1 - \frac{|\tau|}{T}\right) & \text{for } |\tau| < T \\ 0 & \text{Otherwise} \end{cases} \qquad (2.10)$$

where Q_{β_k} is the variance of β_k. The associated power spectral density is

$$S_{w_d w_d}(\omega) = \int_{-\infty}^{\infty} e^{-j\omega\tau} \left[\lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} R_{w_d w_d}(t,t+\tau)dt\right]d\tau$$

$$= \frac{2(1 - \cos \omega T)}{\omega^2} \sum_{k=0}^{K} Q_{\beta_k} \left(\frac{T^{2k-1}}{2k+1}\right) \qquad (2.11)$$

Note that the expression in (2.11) takes advantage of the periodicity of (2.10) and is valid even though the discrete representation of the given continuous random process is nonstationary.

For the continuous white noise case, the autocorrelation function in given by the impulse function

$$R_{ww}(\tau) = Q_w \delta(\tau) \qquad (2.12)$$

and the power spectral density is determined as

$$S_{ww}(\omega) = \int_{-\infty}^{\infty} Q_w \delta(\tau) e^{-j\omega\tau} d\tau = Q_w \qquad (2.13)$$

Equating (2.11) and (2.13) yields

$$Q_w = 2 \sum_{k=0}^{K} Q_{\beta_k} \left(\frac{T^{2k-1}}{2k+1}\right) \left[\frac{T^2}{2} - \frac{T^4 \omega^2}{24} + \frac{T^6 \omega^4}{720} - \cdots\right] \qquad (2.14)$$

from which, by setting $\omega = 0$, one may form the approximate relationship

$$Q_w = \sum_{k=0}^{K} Q_{\beta_k} \left(\frac{T^{2k+1}}{2k+1}\right) \qquad (2.15)$$

This is one of the new relationships developed to possibly yield a more accurate discrete representation of continuous white noise processes. Figure 2 shows the representation of the continuous and discrete white noise processes, including sample functions, autocorrelation functions, and the power spectral densities.

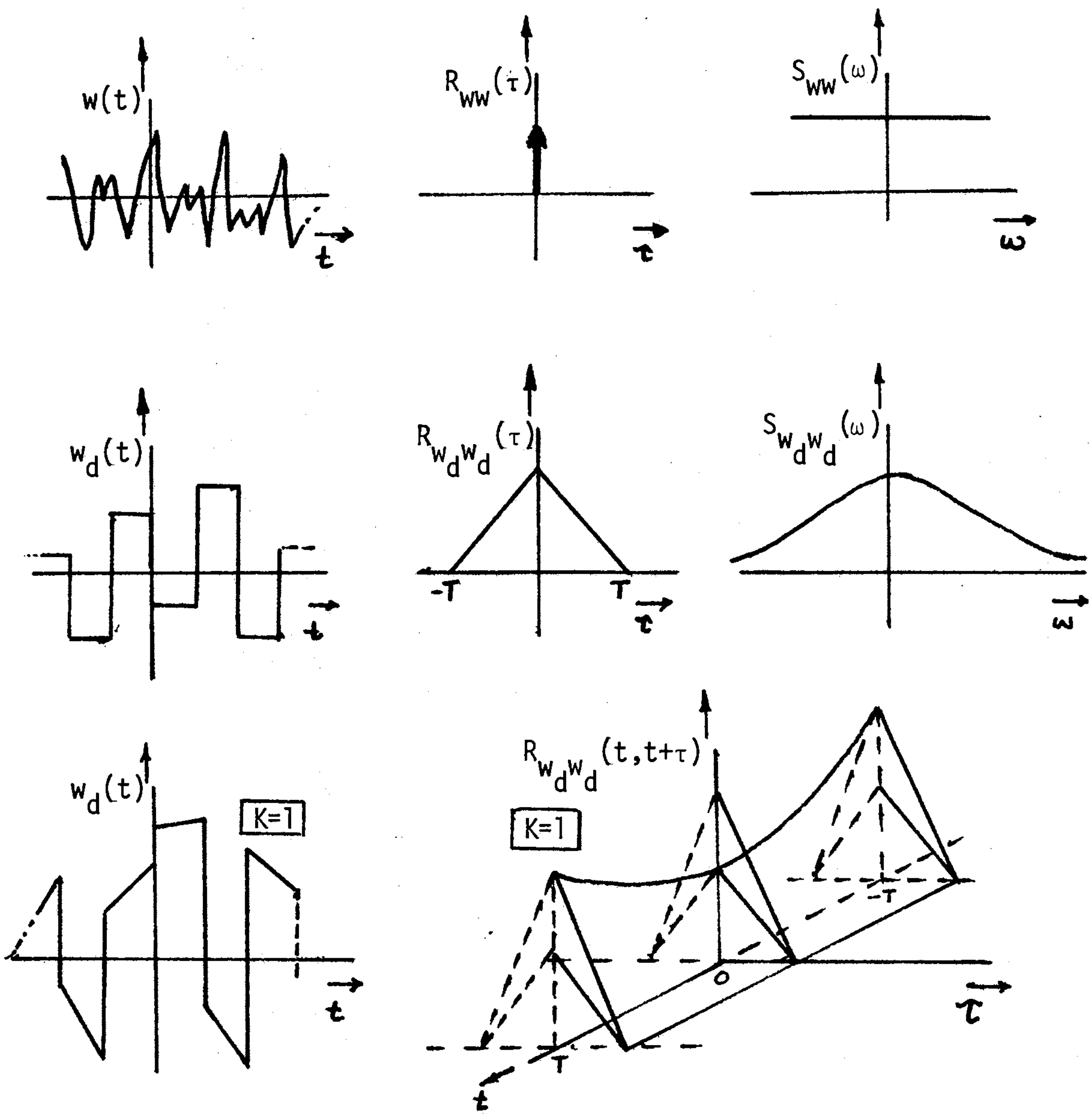

Figure 2, Continuous and Discrete White Noise Representations

Another method was developed towards the improvement of the discrete representation of continuous white noise processes. Consider the random process y(t) given by

$$y(t) = A\cos(\alpha t + \theta) \tag{2.16}$$

where A is a Gaussian random variable with variance σ_A^2 and a mean of zero, α is a constant, and θ is uniformly distributed on the range $(0, 2\pi)$. A and θ are assumed to be independent. It can easily be shown that

$$R_{yy}(\tau) = \begin{cases} \frac{\sigma_A^2}{2}\left(1 - \frac{|\tau|}{T}\right)\cos(\alpha\tau) & \text{for } |\tau|<T \\ 0 & \text{Otherwise} \end{cases} \tag{2.17}$$

Suppose a discrete random sequence $w_d(t)$ is generated by applying (2.16) on an interval-by-interval basis. This sequence may be used to approximate a given continuous white noise process as before by setting

$$Q_w = 2\int_0^T \frac{\sigma_A^2}{2}\cos(\alpha\tau)\cdot\left[1 - \frac{|\tau|}{T}\right]d\tau$$

$$= \sigma_A^2\left[\frac{1-\cos(\alpha T)}{T\alpha^2}\right] \tag{2.18}$$

This is the relationship developed for determining the variance of the discrete model. The simulation results of this method and the method developed earlier in the section are compared with the numerical results obtained earlier in (24). The method in (2.8) is referred to as the standard method, and the method developed in (2.9)-(2.15) is called the slope method. Furthermore, the alternate method in (2.16)-(2.18) is referred to as the cosine method.

Numerical Results

Consider the second-order, linear, time-invariant system described by

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -2x_1 - 3x_2 + w(t) \quad (2.19)$$

Recursive relationships used to generate the random input sequence w_d for the above second-order system have the form

$$Y_{i+1} = GY_i \qquad \text{(Modulo M)} \quad (2.20)$$

Brown and Rowland (27) obtained satisfactory statistical properties from the pseudo-random number generator with G = 19971, $M = 2^{20}$, and $Y_o = 31571$. The generated numbers are uniformly distributed on (0,1). These numbers may be converted into a zero-mean, unity-variance Gaussian distribution by the exact closed-form relation developed by Box and Muller (6)

$$Z_1 = (-2 \log_e Y_1)^{1/2} \cos 2\pi Y_2$$
$$Z_2 = (-2 \log_e Y_1)^{1/2} \sin 2\pi Y_2 \quad (2.21)$$

where Y_1 and Y_2 are uniformly distributed, and Z_1 and Z_2 are Gaussianly distributed random variables.

Numerical results for this example are shown in Figure 3 with the average per cent error on the output variance ($\sigma_{x_1}^2$) versus the number of Monte Carlo runs for the three methods being compared. Using a step size T of 0.05, the standard method utilized pseudo-random numbers with a variance Q_{w_d} of Q_w/T equal to 20. The case of K = 1 was used for the slope method with the random variables

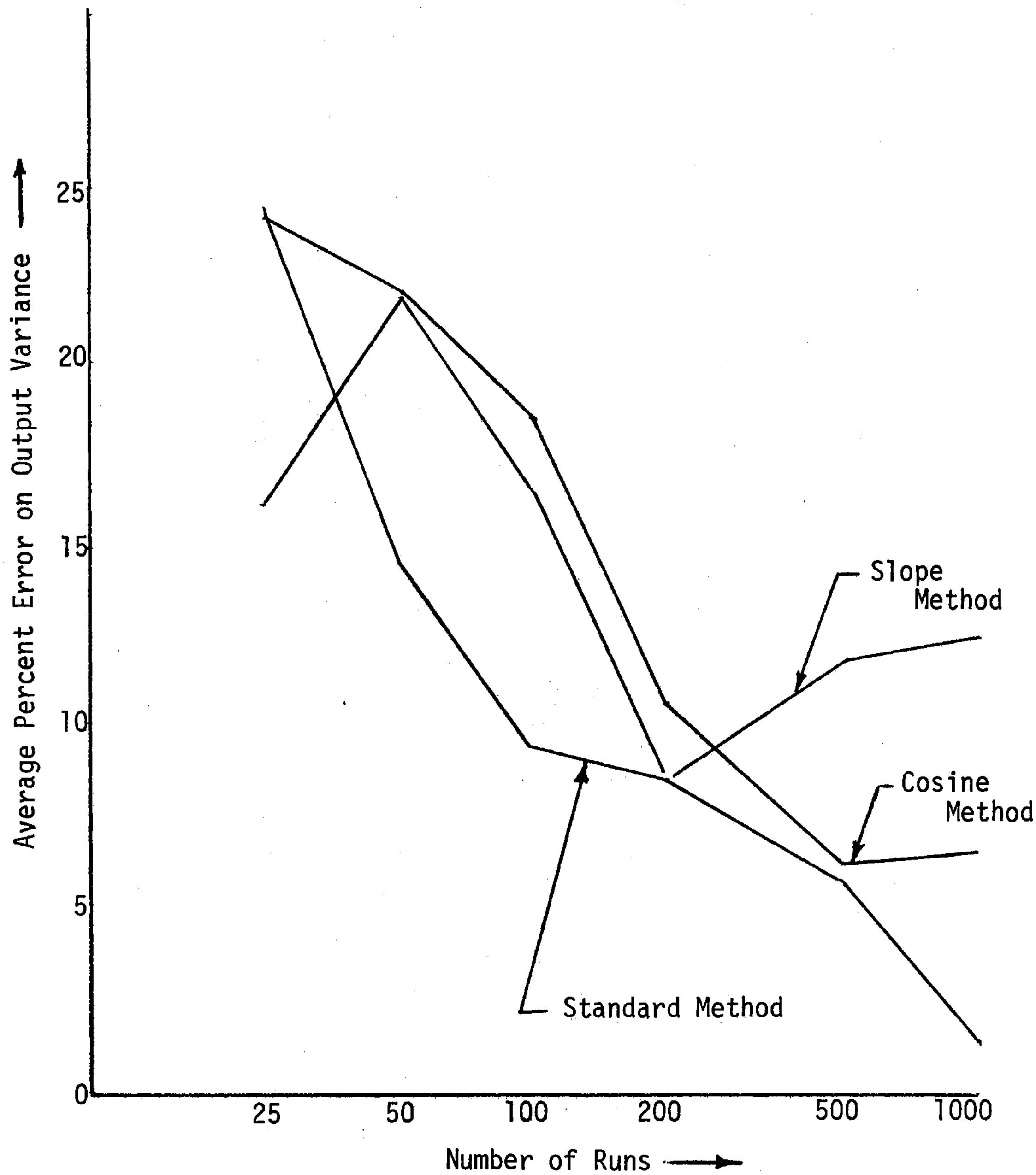

Figure 3. Average Percent Error on the Output Variance by the Monte Carlo Technique

β_0 and β_1 being given equal weight. Several other cases (K = 2,3, and 4) with several alternate weighting methods for the β's were also simulated, but no significant improvement was obtained. The results of the cosine method shown in Figure 3 used $\sigma_A^2 = 6.44$, $\alpha = 4\pi$, and T = 0.05. Different combinations of α and σ_A^2 were also used in other runs without improvement. Moreover, the use of Z_1 and Z_2 from (2.21) in consecutive intervals as opposed to using only Z_1, as shown in Figure 3, failed to yield any improvement. Finally, using alternate values of Z_1 and/or Z_2 did not improve the results shown. Therefore, the standard method was the best of those tested in terms of accuracy. In addition, the standard method requires only a single pseudo-random number per interval, which results in a particularly simple implementation as shown in Appendix A.

Summary

The direct covariance algorithm was extended in this chapter for application to linearized variational equations about the noise-free solution for nonlinear systems. Numerical results showed that the algorithm is applicable to those nonlinear systems with low input noise levels and mild nonlinearities. A generalization (28) was proposed for improving the discretization procedure for simulating continuous white noise processes on the digital computer. Extensive Monte Carlo testing on a second-order system indicated that the standard method developed earlier was both superior in accuracy and the most efficient for implementation purposes. This efficient discretization procedure forms the basis for the subsequent Monte Carlo validation of the computer software package developed in Chapter III.

CHAPTER III

IMPLEMENTATION OF THE DIRECT COVARIANCE ALGORITHM FOR LARGE-SCALE SYSTEMS

This chapter deals with the large-scale implementation of the direct covariance algorithm derived in the Chapter I and extended in Chapter II. A method for obtaining the exact solution for large-scale linear systems is presented, and the problems in implementing this solution for large-scale nonlinear systems are identified. The basic computer software package is developed with a particular emphasis on its application to large-scale missile systems. Initial numerical results are shown for applying the basic software package to a thirty-first order math model of a six degree-of-freedom air defense missile system.

Exact Solutions for Large-Scale Linear Systems

The direct covariance algorithm derived in Chapter I is repeated here for convenience as

$$\dot{P}(t) = A(t)P(t) + P(t)A^T(t) + B(t)Q_{\underline{w}}(t)B^T(t) \qquad (1.10)$$

In component form, (1.10) becomes

$$\begin{pmatrix} \dot{p}_{11} \cdots \dot{p}_{1n} \\ \vdots \quad\quad \vdots \\ \dot{p}_{n1} \cdots \dot{p}_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} \cdots a_{1n} \\ \vdots \quad\quad \vdots \\ a_{n1} \cdots a_{nn} \end{pmatrix} \begin{pmatrix} p_{11} \cdots p_{1n} \\ \vdots \quad\quad \vdots \\ p_{1n} \cdots p_{nn} \end{pmatrix} + \begin{pmatrix} p_{11} \cdots p_{1n} \\ \vdots \quad\quad \vdots \\ p_{1n} \cdots p_{nn} \end{pmatrix} \begin{pmatrix} a_{11} \cdots a_{n1} \\ \vdots \quad\quad \vdots \\ a_{1n} \cdots a_{nn} \end{pmatrix}$$

$$+ \begin{pmatrix} b_{11} \cdots b_{1m} \\ \vdots \quad\quad \vdots \\ b_{n1} \cdots b_{nm} \end{pmatrix} \begin{pmatrix} q_{11} \cdots q_{1m} \\ \vdots \quad\quad \vdots \\ q_{m1} \cdots q_{mm} \end{pmatrix} \begin{pmatrix} b_{11} \cdots b_{n1} \\ \vdots \quad\quad \vdots \\ b_{1m} \cdots b_{nm} \end{pmatrix} \qquad (3.1)$$

Since P(t) is a symmetric matrix, i.e. $p_{ij} = p_{ji}$, the number of component differential equations in (3.1) is n(n+1)/2, where n is the system order.

Equation (3.1) can be solved exactly for constant A and B matrices. Rewriting (3.1) in the vector form yields

$$\dot{\underline{p}}(t) = A' \, \underline{p}(t) + \underline{r} \qquad (3.2)$$

where

$$\underline{p}(t) = \begin{pmatrix} p_{11}(t) \\ p_{12}(t) \\ \vdots \\ p_{nn}(t) \end{pmatrix}$$

and A' and $\underline{r}$ are functions of the components of A, B, and $\underline{Q}_w$ in (3.1). The solution of the linear vector differential equation in (3.2) may be written as

$$\underline{p}(t) = e^{A'(t-t_o)}\underline{p}(t_o) + \int_{t_o}^{t} e^{A'(t-\tau)}\underline{r} \, d\tau \qquad (3.3)$$

where $e^{A'(t-t_o)}$ is the state transition matrix associated with $\underline{p}(t)$ in (3.2). This matrix exponential, sometimes denoted by $\Phi(t-t_o)$, may be evaluated as

$$e^{A'(t-t_o)} = I + A'(t-t_o) + \tfrac{1}{2}A'^2(t-t_o)^2 + \ldots \qquad (3.4)$$

Example

Equation (2.19) may be expressed in vector-matrix form by identifying

$$A = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \quad ; \quad B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad ; \quad \underline{Q}_w = (1)$$

Therefore, (3.1) becomes

$$\begin{pmatrix} \dot{p}_{11} & \dot{p}_{12} \\ \dot{p}_{12} & \dot{p}_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2 & -3 \end{pmatrix} \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix} + \begin{pmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{pmatrix} \begin{pmatrix} 0 & -2 \\ 1 & -3 \end{pmatrix}$$
$$+ \begin{pmatrix} 0 \\ 1 \end{pmatrix} (1) \quad (0 \quad 1) \tag{3.5}$$

Corresponding to (3.2), (3.5) may be written as

$$\begin{pmatrix} \dot{p}_{11} \\ \dot{p}_{12} \\ \dot{p}_{22} \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 \\ -2 & -3 & 1 \\ 0 & -4 & -6 \end{pmatrix} \begin{pmatrix} p_{11} \\ p_{12} \\ p_{22} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{3.6}$$

Using (3.3), the solution to (3.6) for P(0) = 0 is

$$\underline{p}(t) = \begin{pmatrix} \frac{1}{12} - \frac{1}{2} e^{-2t} + \frac{2}{3} e^{-3t} - \frac{1}{4} e^{-4t} \\ \frac{1}{2} e^{-2t} - e^{-3t} + \frac{1}{2} e^{-4t} \\ \frac{1}{6} - \frac{1}{2} e^{-2t} + \frac{4}{3} e^{-3t} - e^{-4t} \end{pmatrix} \tag{3.7}$$

Note that $e^{A'(t-t_o)}$ has $n^2(n+1)^2/4$ elements for an nth order system, which expands the computer storage requirements considerably beyond that required by using the matrix equation in (1.10) to solve for P(t) by numerical integration. For example, if n = 31, then P(t) may be obtained from (1.10) by solving 496 equations, whereas

$e^{A'(t-t_o)}$ would require nearly one-quarter of a million state transition matrix element evaluations. Moreover, if A and B are not constant in time, then the determination of the exact solution of P(t) in (3.2) is generally not possible. Since some components of A(t) and B(t) are always functions of time for nonlinear systems, the use of a suitable numerical integration formula, such as the fourth-order Runge-Kutta algorithm, is recommended for determining P(t) from (1.10) in general nonlinear cases.

The Basic Software Package

The considerations that were made during the development of the software package included obtaining accurate results while using a minimum amount of computer time, satisfying equipment requirements, such as computer storage, and determining the range of applicability for the direct algorithm on nonlinear systems.

The covariance matrix equation (1.10) was integrated along the nominal trajectory by using an integration step size for the covariance equations initially as half that of the system equations. The coefficient matrix A(t) for the system equations is a sparse matrix in many applications. For any large-scale system the coefficient matrix elements may be categorized as either zero, non-zero constants, nonlinear functions of the nominal states, or implicitly related to the nominal states. For example, the thirty-first order missile system considered here had 792 zero coefficient matrix elements, which were neglected during program computations. In addition, constant elements were defined in the beginning of the program and left unchanged thereafter. The coefficient matrix was computed at each integration interval along with the nominal solution to yield a considerable savings in computer storage over the method of storing the A(t) matrix for all time t. Thus, each nonlinear element of A(t) was updated during each interval. Finally, those coefficient matrix elements which are related to certain state variables only implicitly, i.e. the functional relationship is available only via complicated computer programmed statements, were computed numerically at each interval. Additional details will be provided following the description of the large-scale application in the next section.

The application of the direct covariance algorithm to the thirty-first order nonlinear missile system yielded only approximate results because the accuracy of the direct covariance algorithm for nonlinear systems depends entirely upon the relative accuracy of the linearizing approximation for incremental variations about the noise-free solution. The error in the direct covariance results increases as the nonlinear terms in the exact incremental equation become more significant. The time-varying coefficient matrix prohibits the use of the state transition matrix equations. Thus, an accurate numerical integration technique was needed to integrate the n(n+1)/2 equations for the symmetrical covariance matrix.

The basic approach in the development of the software package is shown in Figure 4 in the form of a flow chart. The Fortran listing of this computer software package applied to a thirty-first order math model of a six degree-of-freedom air defense missile system is given in Appendix B.

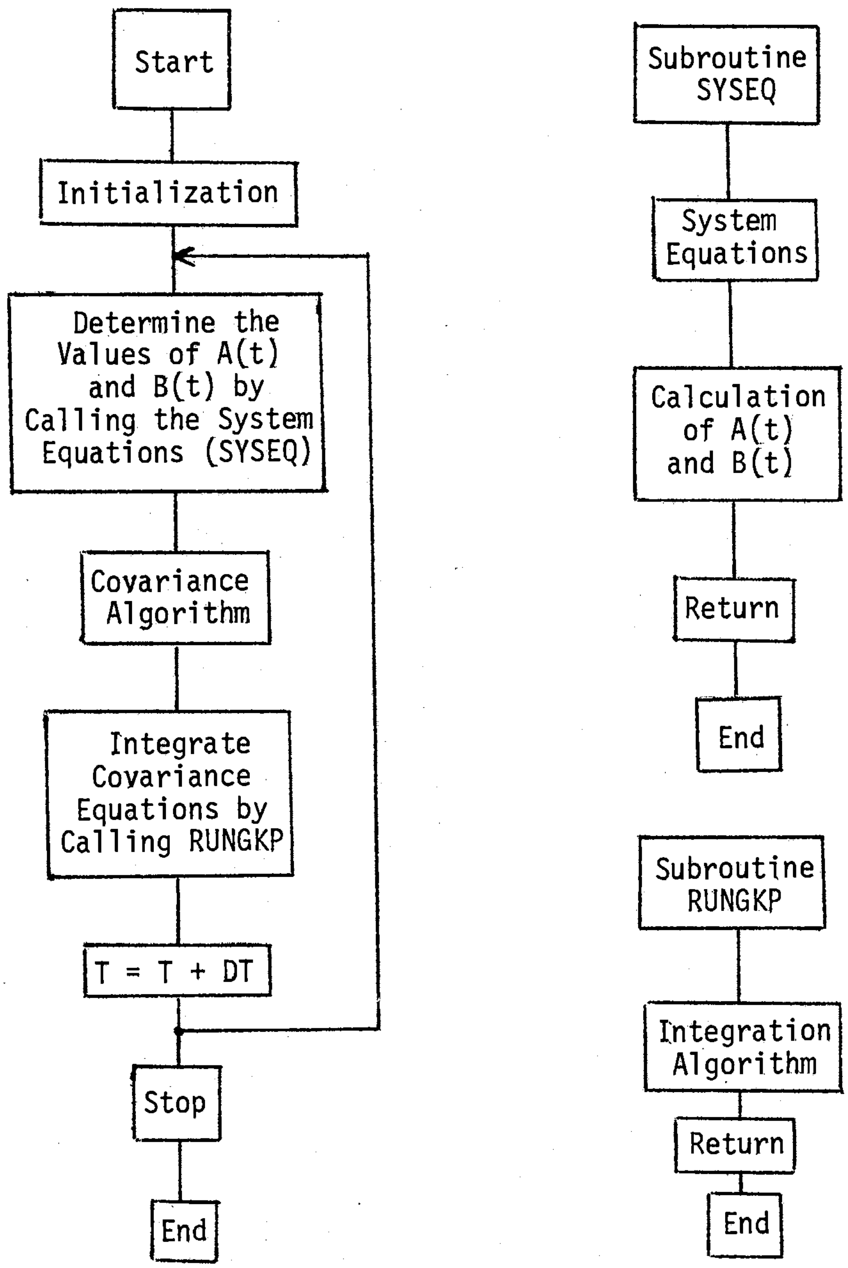

Figure 4. Flow Chart for the Development of the Computer Software Package

Description of the Missile System Application

The large-scale system investigated here is a thirty-first order math model of a six degree-of-freedom air defense missile system. The autopilot subprogram in fifteenth-order, the airframe subprogram which includes the missile rotational variables, the translational equations of motion, and launcher dynamics is twelfth-order, and the actuator subprogram is fourth-order. The block diagram for the thirty-first order missile system in shown in Figure 5 with details of the autopilot and actuator in Figure 6. The target routine shown in the figure calculates the target-to-missile relative position and speed and generates line of sight signals.

Table I identifies all states of the missile system and assigns a specific number to each state. For example, the missile altitude z is defined as the twenty-first state and occurs in the airframe subprogram. Table II provides the complete categorization of all elements of A(t) as either zero, indicated by blank entries, constant values (C), nonlinear functions of the nominal trajectory (NL), or numerically computed (NC). The number and per cent contained in each category are summarized in Table III.

Numerical Results

This section deals with the description of the method used for numerically calculating the A(t) coefficient matrix elements. Later in the section a detailed description of the input noise to the large-scale system is given. Finally, the numerical results obtained by applying the direct covariance algorithm to the thirty-first order

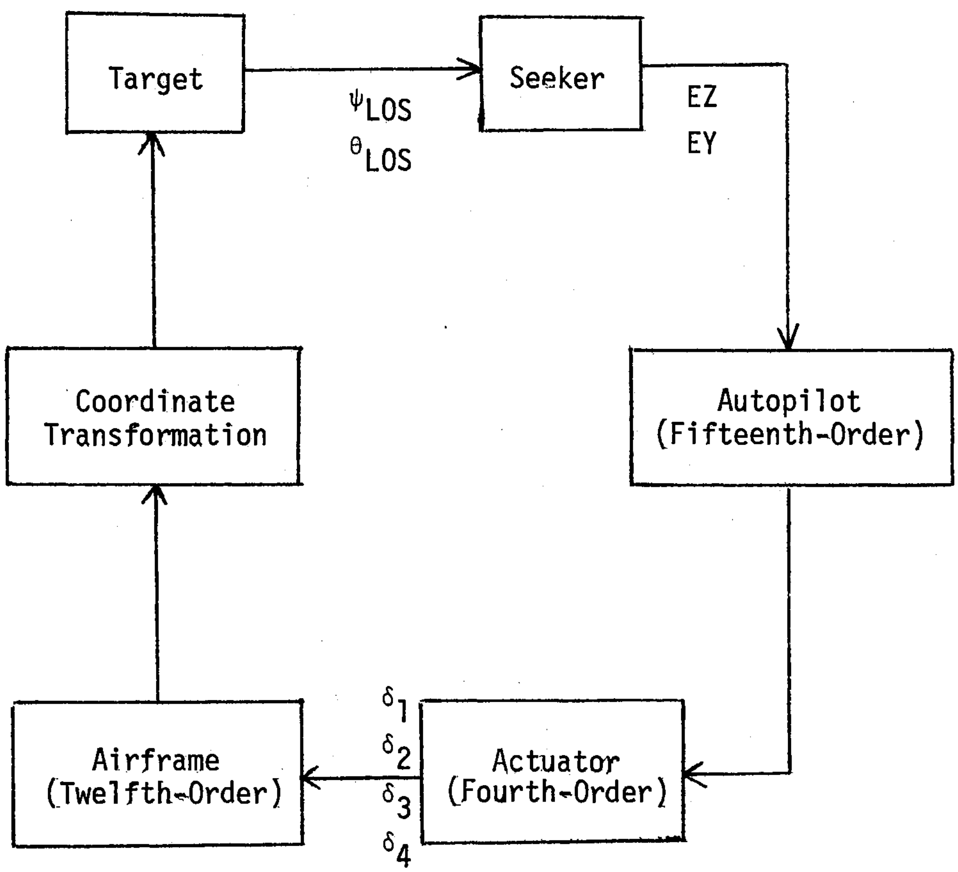

Figure 5. Block Diagram for the Thirty-First Order Missile System

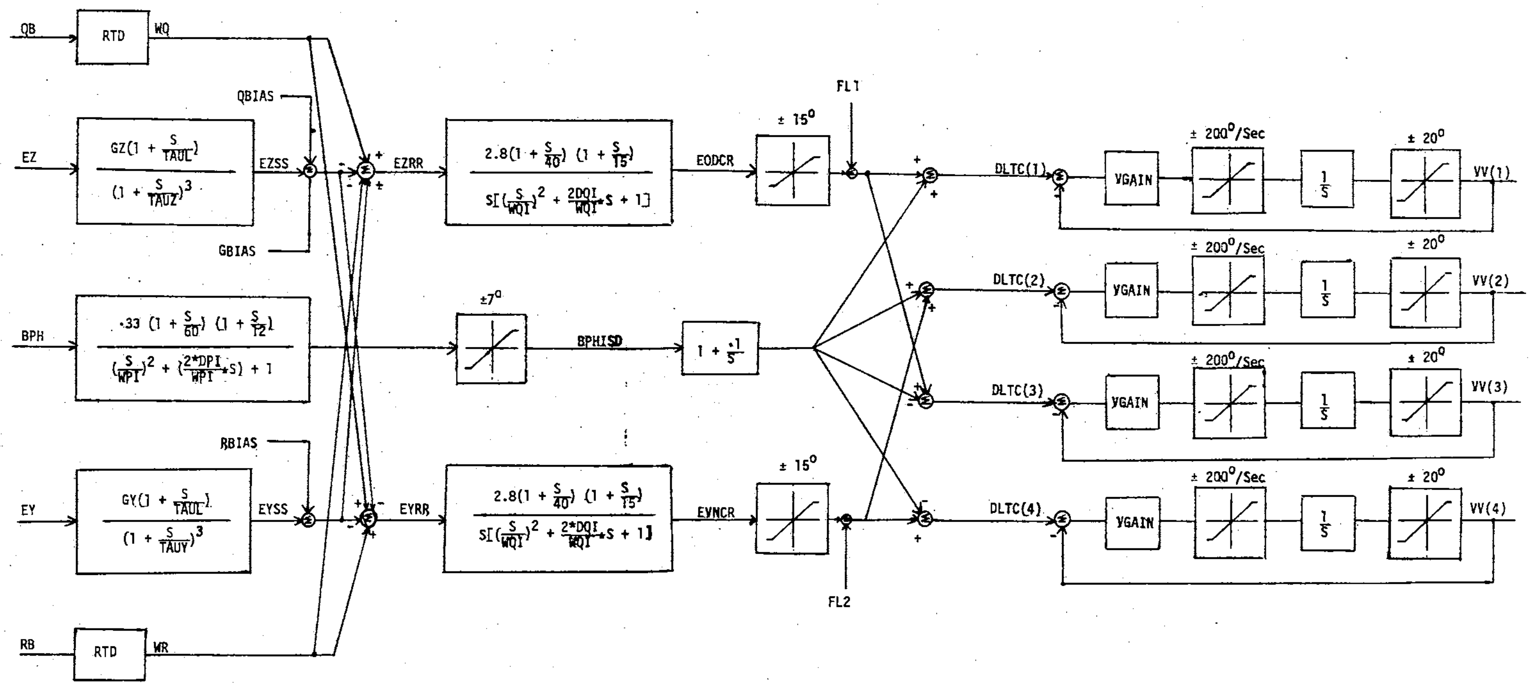

Figure 6. Block Diagram for the Autopilot and Actuators

TABLE I

DEFINITION OF THE MISSILE SYSTEM STATE VARIABLES

Subprogram	Description of State Variables	State Identification Name	State Identification
I. Autopilot	Guidance Pitch Filter	ZP1	1
		ZP2	2
		ZP3	3
	Guidance Yaw Filter	ZY1	4
		ZY2	5
		ZY3	6
	Roll Compensation	ZR1	7
		ZR2	8
		BPHIS	9
	Pitch Integrator	ZPI1	10
		ZPI2	11
		EODCR	12
	Yaw Integrator	ZYI1	13
		ZYI2	14
		EVNCR	15
II. Airframe	State Variables for Evaluating the Translational Equations of Missile Motion.	UE	16
		VE	17
		WE	18
		X	19
		Y	20
		Z	21
	Missile Rotational Variables	PB	22
		QB	23
		RB	24
	Euler Angles	THETA	25
		PHI	26
		PSI	27
III. Actuator	Vane Module Variables	VV(1)	28
		VV(2)	29
		VV(3)	30
		VV(4)	31

TABLE II

COEFFICIENT MATRIX FOR THE MISSILE SYSTEM

	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	28	30	31
1	C	C	C																												
2	C																														
3		C																													
4				C	C	C																									
5				C																											
6					C																										
7							C	C																		C					
8							C																								
9							NL	NL																		NL					
10		C	C		C	C				C	C												C	C							
11										C																					
12		C	C		C	C				C	C												C	C							
13		C	C		C	C							C	C									C	C							
14													C																		
15		C	C		C	C							C	C									C	C							
16																NC	NC	NC			NC				NC	NC	NC	NC	NC	NC	NC
17																NC	NC	NC			NC	NC*	NC*	NC*	NC	NC	NC	NC	NC	NC	NC
18																NC	NC	NC			NC	NC*	NC*	NC*	NC	NC	NC	NC	NC	NC	NC
19																C															
20																	C														
21																		C													
22																NC	NC	NC			NC	NC			NC	NC	NC	NC	NC	NC	NC
23																NC	NC	NC			NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
24																NC	NC	NC			NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
25																							NL	NL		NL					
26																						C	NL	NL	NL	NL					
27																							NL	NL	NL	NL					
28							NL	NL	NL			NL														NL		NL			
29							NL	NL	NL						NL											NL			NL		
30							NL	NL	NL			NL														NL				NL	
31							NL	NL	NL						NL											NL					NL

TABLE III

CATEGORIZATION OF COEFFICIENT MATRIX ELEMENTS

Categorization	Number	Percentage
Zero Elements	792	82.4%
Constant Elements	52	5.4%
Nonlinear Elements	38	4.0%
Implicitly Related Elements	79	8.2%
Total	961	100.0%

system are compared with 25 Monte Carlo runs.

Only those elements of the A(t) coefficient matrix which are implicitly related to certain variables are computed numerically. For the thirty-first order math model of the six degree-of-freedom air defense missile system, the numerically computed elements are denoted in Table III by NC. The state identification of these state variables is given in Table I. The elements labelled NC* in Table III are computed to modify the derivatives when launcher dynamics of the missile system are in effect and are equated to zero after the second lug leaves the launcher. Numerically, the partial derivatives for A(t) in (2.6) are given by

$$A(t) = \frac{\underline{f}(\underline{x}_N + \underline{\Delta x}, \underline{\eta_w}, t) - \underline{f}(\underline{x}_N, \underline{\eta_w}, t)}{\underline{\Delta x}} \qquad (3.8)$$

where the notation $\underline{\Delta x}$ represents small perturbations about the nominal flight path $\underline{x}_N(t)$. These perturbations have small lower limits when P(t) is very near zero, but $\underline{\Delta x}$ is increased by adding one-tenth of the standard deviation of the particular state under consideration when P(t) is set near zero. Therefore, the numerically computed elements of A(t) result in an adaptive feature for the direct covariance algorithm.

The large number of sequential calculations for the noise propagation equations results in numerical problems which can be handled most effectively by using double-precision throughout. To avoid these time consuming operations, the elements in a particular column of P(t) were set to zero whenever the corresponding diagonal element was below 10^{-10}. Since this limiting value was chosen arbitrarily, additional work is needed to remove this arbitrariness.

For the noise propagation studies, the noise was introduced at four places in the missile system. The first two places are shown in Figure 6, and the other two white noise inputs were added to the seeker subprogram of the missile system. These latter two noise inputs involved perturbing the line-of-sight signals ψ_{LOS} (BEPSZ) and θ_{LOS}(BEPSY) generated by the target subporgram as shown in Figure 5. These noise signals were passed through the dead-zone as shown in Figure 7. Two subprograms which were developed to obtain the variance of noise after passing it through the dead-zone are included in Appendix B as Subroutines SNOISE AND DETARA. These subprograms utilize the three cases depicted in Figure 8 in which the nominal values of BEPSZ or BEPSY lie below -TMP1, between -TMP1 and -TMP1, or above +TMP1. The density functions of EZ and EY are each composed of three impulses at SKSP or SKSY, zero, and -SKSP or -SKSY. The weighting on each of these impulses is determined by the area of the Gaussian input signals lying within the different ranges of the dead-zone nonlinearity as shown in Figures 7 and 8. The calculation of this area is performed in Subroutine DETARA. It should be emphasized that the dead-zone is a very harsh nonlinearity, which can result in a severe test in applying the direct covariance algorithm. However, the seeker noise was injected at this point in the system because such noise disturbances do occur in the actual missile system.

Figure 9 shows a comparison between the results obtained from the computer software package using the direct covariance algorithm and twenty-five Monte Carlo ensemble-averaged runs for that portion of the missile flight between one and two seconds. This part of the flight was selected for comparison purposes to avoid both the extremely harsh

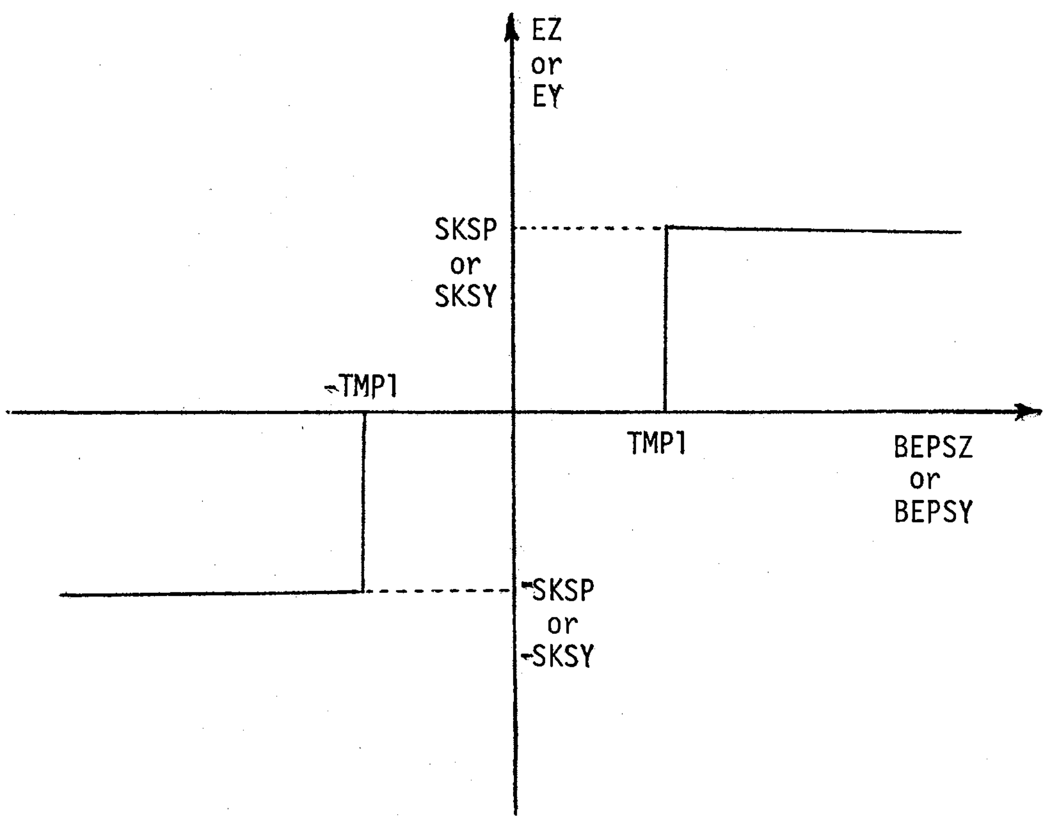

Figure 7. Dead-Zone Details Used in the SEEKER Subprogram

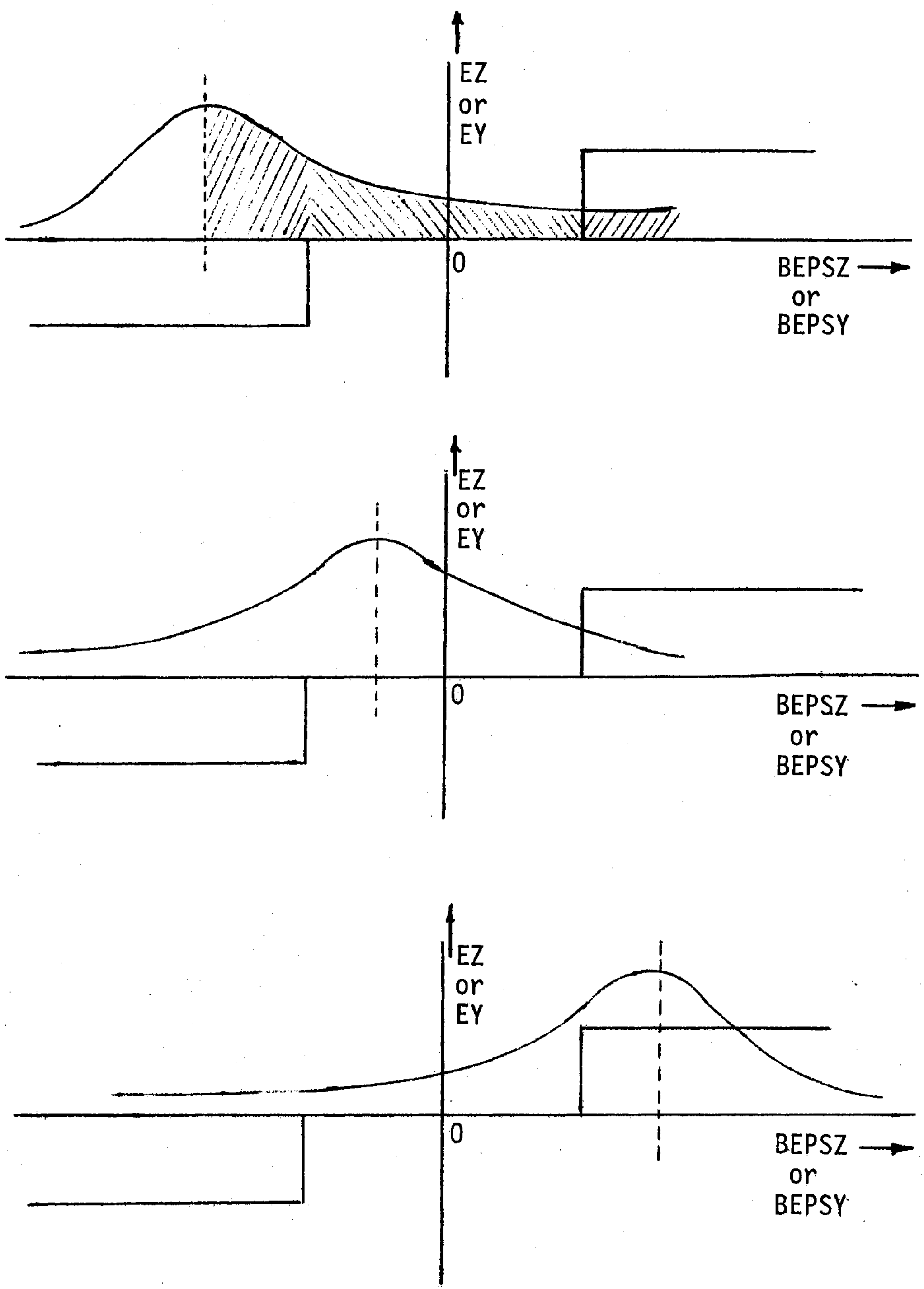

Figure 8. The Effects of the Dead-Zone Nonlinearity on Seeker Noise Inputs

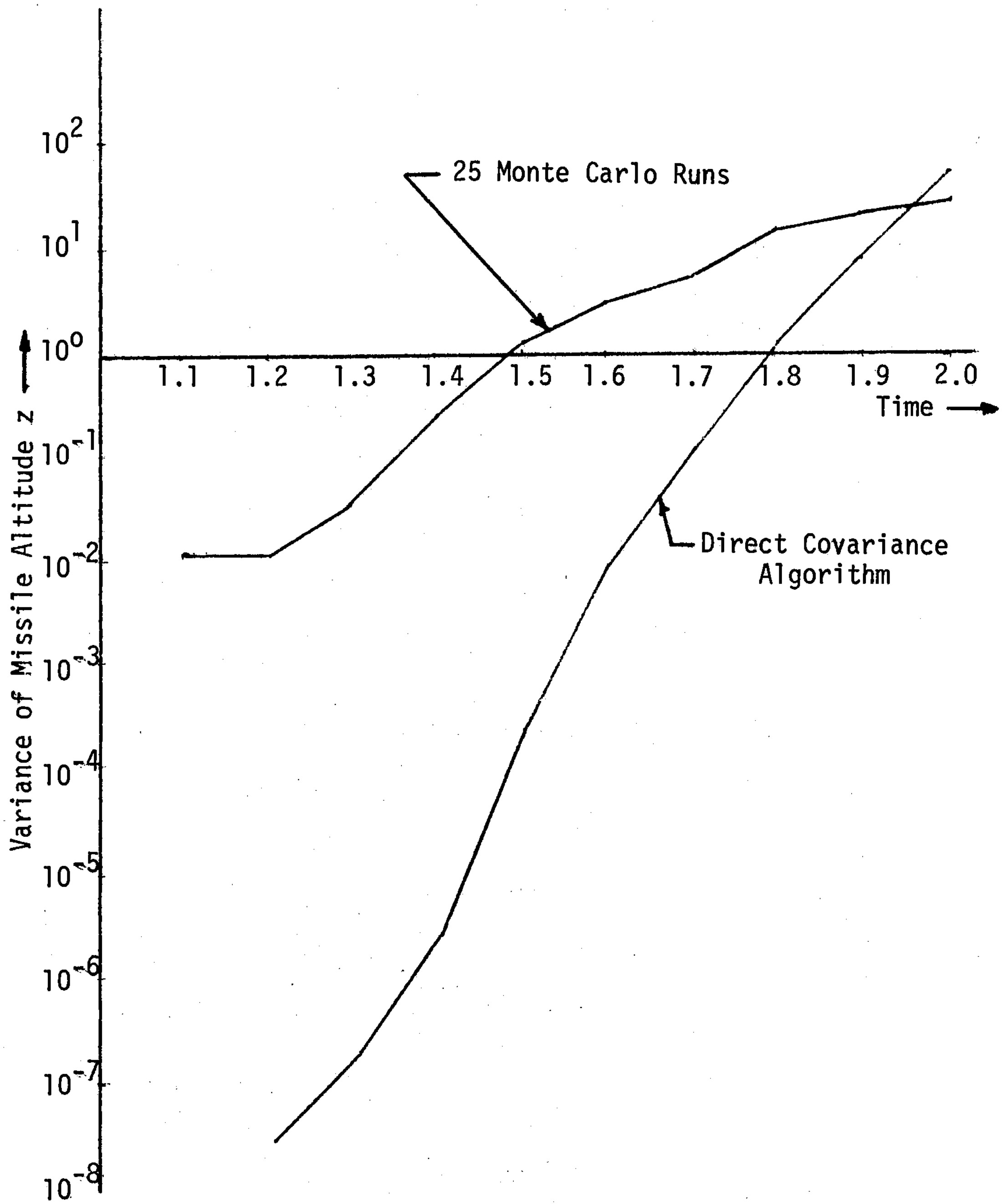

Figure 9. Comparisons of the Results Obtained from the Direct Covariance Algorithm and Monte Carlo Simulation

nonlinear characteristics of the launcher and the equally harsh nonlinear conditions as the missile approaches the target. The input noise variances for FL1 and FL2 were both 0.25 degrees^2 with seeker noise variances of $(0.15 \text{ degrees})^2$. These seeker noise characteristics were selected to conform with those used earlier in a terminal homing simulation on the hybrid computer at the U. S. Army Missile Command. All noise disturbances were first injected at one second into the missile flight, which meant that all states had a zero variance at that initial time of noise injection. Figure 9 shows that the Monte Carlo results rose very rapidly within one-tenth of a second after the noise was first injected into the missile program. On the other hand, the software package using the direct covariance algorithm yielded a steady logarithmic rate of increase. The differences in these two curves indicates that the missile system under consideration is operating in a highly nonlinear region for which the direct covariance algorithm gives unacceptable results. Further work is needed to pinpoint those regions of operation for which the software package can be applied directly and those regions in which the Monte Carlo technique and the covariance software package may be combined to yield satisfactory results.

Summary

The development of the basic computer software package for the direct covariance algorithm was described in this chapter. Its application to a thirty-first order six degree-of-freedom air defense missile system demonstrated that there are highly nonlinear regions in which the software package results are not in close agreement with

Monte Carlo results. Nevertheless, there is a need to consider trade-off possibilities to obtain greater efficiency for large-scale nonlinear systems operating in mildly nonlinear regions.

CHAPTER IV

ENGINEERING TRADEOFF STUDIES FOR THE DIRECT COVARIANCE ALGORITHM

Engineering tradeoffs are investigated in this chapter to improve the computational efficiency of the digital computer software package developed in Chapter III. Following a general discussion of the tradeoff philosophy, numerical comparisons on the large-scale missile system are made between accuracy and computational speed. The problems of computer storage and program complexity are then considered with regard to the use of an automatic sensitivity program for computing A(t) at each integration interval. Therefore, the final form of the computer software package is obtained by utilizing these indicated engineering tradeoffs to yield a computationally efficient program for the direct covariance algorithm.

Tradeoff Considerations

The considerations that must be made during tradeoff studies are closely related to the criteria for comparison purposes presented in Chapter I. Since the information provided and the extension possibilities are fixed by selecting the direct covariance approach, only the remaining criteria of accuracy, computational speed, computer storage, and program complexity may be used for tradeoff possibilities.

Accuracy

Accuracy plays a major role in achieving computational efficiency, since it has an inverse relationship with the computational speed. For example, trading accuracy for computational speed by changing the integration method from the fourth-order Runge-Kutta formula (RK4) to the second-order Runge-Kutta formula (RK2) may reduce the computation time considerably for large-scale systems. In any simulation problem the minimum acceptable accuracy level limits the maximum integration step size that may be chosen. Tradeoffs for the large-scale system are also influenced by the fact that direct covariance technique gives exact results for linear systems while the errors in the results of nonlinear systems depend on the amount of nonlinearity and the input noise level. In addition to the choice of integration method and the selection of the step size, the frequency at which the coefficient matrix is updated affects the algorithm accuracy.

Computational Speed

Tradeoffs may be used to minimize the computer time needed for the large-scale simulation and the application of the direct covariance algorithm. For the developed software package, the integration time needed for the covariance matrix equations may be reduced by nearly one-half by changing the integration method from RK4 to RK2, as mentioned earlier. A savings in computer time is also obtained by categorizing the coefficient matrix elements as zero, constants, nonlinear, and implicitly related to the state variables. Since the A(t) matrix is usually a sparse matrix, many coefficient elements are zero and thus neglecting them entirely during the calculations reduces the computer time considerably. Table III summarizes this categorization for the thirty-first order missile system described in Chapter III. Finally, further reductions in computational time may be achieved by calculating the A(t) coefficient matrix elements after every few integration intervals instead of every integration interval.

Computer Storage

The computer storage needed for applying the software package to the large-scale system can also be reduced by tradeoff. The general implementation of the direct covariance algorithm for large-scale systems requires a much higher computer storage as compared to a particular implementation. For an nth-order system, storing the large A(t) and B(t) matrices requires a large amount of computer storage. This may be reduced by deleting the zero elements and either converting these matrices into smaller matrices or to vector form. However, this procedure would tend to increase the complexity of the computer software package.

Program Complexity

The program complexity is another measure of an efficient computer software package. The general implementation of the direct covariance algorithm may reduce the program complexity to a minimum, whereas a particular implementation makes it quite complex. The complexity also increases, as noted above, by converting A(t) and B(t) in smaller matrices or vector form. Thus, a balance must be reached by trading accuracy, computational time, computer storage, and program complexity to provide a computationally efficient final software package.

Accuracy Versus Computational Speed

In the last section on tradeoff considerations it was mentioned that accuracy and computational speed are inversely related. Tradeoff studies were made between accuracy and computational time for the thirty-first order missile system, and the results are given in Table IV. The accuracy level was varied by using different integration methods (RK4 and RK2) and by changing the integration step size for obtaining the covariance matrix elements. The nominal solution was run at an integration step of 0.0025 seconds. The accuracy data provided in Table IV refers to the flight segment between one and two seconds into the missile flight, but the computational time is for the entire 10,000 ft. flight of approximately 12.8 seconds duration. The table entry denoted as RK2(a) refers to the application of the second-order Runge-Kutta formula on the basic system as given in Appendix B. However, RK2(b) utilized an added program feature in which the randomness of TMP1 in the seeker program is considered. The conclusion from Table IV is that RK2(b) represents an acceptable tradeoff between accuracy and computational time.

Computer Storage and Program Complexity

The computer storage utilized for RK2(b) was approximately 28K words, including the nominal solution program. The direct covariance algorithm had been programmed almost as efficiently as possible to require minimum core storage. The computer storage could be reduced further only by deleting zero elements of the A(t) matrix to convert it to a smaller matrix or to vector form. The program complexity would increase dramatically if such a program change were initiated.

TABLE IV

TRADEOFF STUDIES BETWEEN ACCURACY AND COMPUTATIONAL TIME

Integration Method	Step Size (sec)	Per Cent Difference from RK4 at t=2 sec.	Computational Time (Minutes)
RK4	.00125	---	75
RK2 (a)	.00125	68%	41
RK2 (b)	.00125	27%	41
RK2	.00250	92%	25

Therefore, the program given in Appendix B represents a suitable tradeoff between computer storage and complexity as well as between accuracy and computational time. Finally, the use of an automatic sensitivity program for evaluating all elements of A(t) at each integration interval would increase the computational time well beyond an hour for the missile system under consideration. While such a program innovation would provide the user with a rough estimate of the software package accuracy in the presence of power-law nonlinearities, its utilization is ineffective for the given missile system with dead-zone nonlinearities. In addition, both program complexity and computer storage requirements would be unacceptable for this type of large-scale application.

Summary

Tradeoffs between accuracy, computational speed, computer storage, and program complexity were investigated to yield a more computationally efficient software package. The result was a somewhat less accurate, but faster, application of the direct covariance algorithm for large-scale systems.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

In this research the direct covariance algorithm has been used to develop and implement a computer software package for noise propagation studies in large-scale systems. The Monte Carlo technique has been used to yield results for comparison with the results obtained by the covariance technique. Two methods were proposed to improve the discretization of continuous white noise used in the Monte Carlo simulation. It has been shown that the standard method is superior to the other methods considered both in accuracy and efficiency.

For large-scale systems it was shown that the state transition method was unreasonable to use because of the large number of calculations needed. Therefore, the resulting development of a computer software package for the direct covariance algorithm was based on using a Runge-Kutta integration formula for the propagation equations. This software package was applied to a thirty-first order math model of a six degree-of-freedom air defense missile system. Comparisons made with 25 Monte Carlo simulation runs indicated that the missile system was operating in a highly nonlinear region due primarily to dead-zone nonlinearities in the seeker subprogram.

Engineering tradeoffs were performed on the software package between accuracy, computational speed, computer storage, and program complexity. It was shown that the RK2 method with a step size equal to half that used in evaluating the nominal trajectory yielded an acceptable accuracy while reducing considerably the associated computational time. The end result is a computationally efficient computer software package for handling noise propagation problems in large-scale missile systems operating in mildly nonlinear regions.

Recommendations for Further Work

The areas recommended for the future research include the expansion of the basic software package for higher-order systems and further investigations of simplifying approximations and computer storage requirements.

The computer software package developed in this research may be expanded to handle higher-order large-scale missile systems. Further work is needed to handle the program in double-precision and for determining the best way to handle the regions of harsh nonlinear operations.

Simplifying approximations for large-scale systems should be examined in more detail. Further work is also needed to examine computer storage requirements for large-scale systems.

SELECTED BIBLIOGRAPHY

(1) RAND Corporation. A Million Random Digits with 100,000 Normal Deviates. Free Press, New York, 1955.

(2) Chambers, R. P. "Random Number Generation." IEEE Spectrum, Vol. 4, No. 2, February 1967.

(3) Hull, T. E. and A. R. Dobell. "Random Number Generators." SIAM Review, Vol. 4, 1962, pp. 230-254.

(4) MacLaren, M. D. and G. Marsaglia. "Uniform Random Number Generators." Journal of the Association of Computing Machinery, Vol. 12, 1965, pp. 83-89.

(5) Gelder, A. V. "Some New Results in Pseudorandom Number Generation." Journal of the Association of Computing Machinery, Vol. 14, 1960, pp. 785-792.

(6) Box, G. E. and M. E. Muller. "A Note on the Generation of Normal Deviates." Annals of Mathematical Statistics, Vol. 28, 1958, pp. 610-611.

(7) Sage, Andrew P. Optimum Systems Control. Prentice Hall, 1968, Section 9.2, pp. 220-226.

(8) Sage, Andrew P. and James M. Melsa. Estimation Theory: With Applications to Communications and Control. New York: McGraw-Hill, 1970.

(9) Meditch, James S. Stochastic Optimal Linear Estimation and Control. New York: McGraw-Hill, 1969, Sections 4.3 and 4.4, pp. 125-152.

(10) Liebelt, Paul B. An Introduction to Optimal Estimation, Sections 4.10 and 4.16, Addison-Wesley, 1967, pp. 112-134.

(11) Bryson, Arthur E., Jr. and Yu-Chi Ho. Applied Optimal Control, Sections 11.4 and 11.5, Blaisdell Publishing Company, 1969, pp. 328-344.

(12) Jazwinski, Andrew H. Stochastic Processes and Filtering Theory, Academic Press, 1970.

(13) Kalman, R. E. and R. S. Bucy. "New Results in Linear Filtering and Prediction Theory." ASME Transactions: Journal of Basic Engineering, Vol. 83D, March 1961, pp. 95-108.

(14) Kalman, R, E. "A New Approach to Linear Filtering and Prediction Problems." ASME Transactions; Journal of Basic Engineering, Vol. 82D, March 1960, pp. 35-45.

(15) Kuhnel, Walter C. and Andrew P. Sage. "Terminal State Error Analysis Using Adjoint-Generated Sensitivities." IEEE Transactions on Aerospace and Electronic Systems, Vol. 5, No. 2, March 1969, pp. 185-194.

(16) Irwin, John D. and James C. Hung. "Methods for Injection-Error Analysis and Their Comparison." IEEE Transactions on Automatic Control, Vol. 12, No. 3, June 1967, pp. 276-281.

(17) Calfee, R. V. LTV Missiles and Space Division, "Oral Presentation to Systems Analysis Branch." U. S. Army Missile Command, Redstone Arsenal, Alabama, July, 1970.

(18) Higdon, Donald T. An Approximate Method for Determining Response of Nonlinear Dynamic Systems to Random Disturbances, Department of Aeronautical Engineering, Wichita State University, February 1967, Aeronautical Report No. 67-2, NASA Grant NGR 17-003-005.

(19) Brown, Robert J., Jr. "Trajectory Optimization for the Combined Estimation and Control of Nonlinear Stochastic Systems." School of Electrical Engg., Georgia Institute of Technology, Atlanta, Georgia, May 1971, Ph.D. Thesis.

(20) Brown, Robert J., Jr. and James R. Rowland. "Trajectory Optimization for Open-loop Nonlinear Stochastic Systems." Proceedings of the Third Annual Southeastern Symposium on System Theory, Vol. I, Georgia Institute of Technology, Atlanta, Georgia, 5-6 April 1971, Paper F4.

(21) Brown, Robert J., Jr. and James R. Rowland. "Trajectory Optimization for Closed-Loop Nonlinear Stochastic Systems." Automatic Control Conference, Washington University, St. Louis, Missouri, 11-13 August 1971, Paper No. 7-D4.

(22) Clark, George M., Jr. "Improved Estimation and Control for Nongaussian Stochastic Systems." School of Electrical Engg., Georgia Institute of Tech., Atlanta, Georgia, May 1971, Ph.D. Thesis.

(23) Clark, George M., Jr. "Specific Controller Synthesis for Linear Stochastic Systems." Proceedings of the Third Annual Symposium on System Theory, Volume II, Georgia Institute of Technology, Atlanta, Paper K6, 5-6 April 1971.

(24) Rowland, James R. and Willard M. Holmes. "Statistical Analysis Techniques for Error Propagation in Large Scale Missile Systems." Report No. RG-TR-71-19, Guidance and Control Directorate, U. S. Army Missile Command, Redstone Arsenal, Alabama, August, 1971.

(25) Papoulis, Athanasios. Probability, Random Variables, and Stochastic Processes, McGraw-Hill Book Company, New York, 1965.

(26) Laning, H. H. and R. H. Battin. Random Processes in Automatic Control, McGraw-Hill, 1956.

(27) Brown, Robert J., Jr., and James R. Rowland. Autocorrelation Significance in Digital Pseudo-Random Number Generation. School of Electrical Engineering, Georgia Institute of Technology, Atlanta, Georgia, March 1970, pp. 1-20, Internal Technical Report.

(28) Rowland, James R. and Vijayendra M. Gupta. "Digital Simulations for Monte Carlo Analysis." Proceedings of the Fifteenth Midwest Symposium on Circuit Theory, University of Missouri-Rolla, Vol. I, Paper V.3, May 4-5, 1972.

APPENDIX A

COMPUTER PROGRAM FOR THE STANDARD METHOD OF MONTE CARLO SIMULATION

The program for the Monte Carlo technique using the standard method has been included in this Appendix. A second-order system was used to obtain Monte Carlo results for 25, 50, 100, 200, 500 and 1000 runs for comparing the results with other methods as discussed in Chapter II.

Statements 35 through 46 were used to generate zero-mean, unity-variance, Gaussianly distributed random numbers. Subsequent instructions were used for the calculation of the output variance and the percentage error on the output variance. The Runge-Kutta second-order formula (RK2) was used for integrating the second-order system.

```
      DIMENSION XE(2),XS(2),XMO(2),XM1(2),S(10),SOL(10),DIF(10),XEM(10)
      T=0.
      H = 0.05
      MS=2
      II = 0
      NTOTAL=100
      MTOT=NTOTAL/10
      DO 31 N=1,MTOT
      S(N) = 0.
      XEM(N)= 0.
   31 CONTINUE
      XMEAN=0.
      IX=31571
      DUM=0.1
      SIG = SQRT(1./H)
      DO 82 I=1,40
      IF(I.EQ.1) GO TO 81
      IF(I.EQ.2) GO TO 81
      IF(I.EQ.4) GO TO 81
      IF(I.EQ.8) GO TO 81
      IF(I.EQ.20)GO TO 81
      IF(I.EQ.40)GO TO 81
      GO TO 82
   81 NUM = 25*I
      XNUM = NUM
      XNUM1 = XNUM*XNUM
      XNUM2 = XNUM - 1.0
      XNUM3 = XNUM/XNUM2
      JJ=II+1
      DO 32 M=JJ,NUM
      XE(1)=0.
      XE(2)=0.
      DO 42 N=1,MTOT
      DO 52 L=1,10
      IY=19971*IX
      IYP=IY/1048576
      IX=IY-IYP*1048576
      AX=IX
      U=AX/1048576.
      IF(U)5,5,6
    5 U=-U
    6 CONTINUE
      IX=IY
      Z=SQRT(-2.0*ALOG(DUM))*SIG
      XNORM =Z*COS(6.28318*U)+XMEAN
      DUM=U
      CALL XEQN(XE,XMO,XNORM)
      DO 23 K=1,MS
   23 XS(K)=XE(K)+H*XMO(K)
      CALL XEQN(XS,XM1,XNORM)
      DO 24 K=1,MS
   24 XE(K)=XE(K)+0.5*H*(XMO(K)+XM1(K))
   52 CONTINUE
      S(N) = S(N) + XE(1)*XE(1)
```

```
      XEM(N) = XEM(N) + XE(1)
   42 CONTINUE
   32 CONTINUE
      WRITE(6,84)NUM
   84 FORMAT(1X,//' NO. OF RUNS = ',I5)
      WRITE(6,15)
      WRITE(6,83)
   83 FORMAT(T11,'TIME',T25,'S(NA)',T38,'SOL(NA)',T53,'DIF(NA)',T68,
     1'XEM(NA)')
      DO 62 NA=1,MTOT
      XNA=NA
      T=H*XNA*10.
      SOL(NA)   =0.08333333333-0.5*EXP(-2.0*T)+0.6666666667*EXP(-3.0*T)
     1-0.25*EXP(-4.0*T)
      XEM(NA) = XEM(NA)*XEM(NA)/(XNUM*XNUM)
      XEM(NA) = XEM(NA)*XNUM3
      S(NA) = S(NA)/XNUM2- XEM(NA)
      DIF(NA) = 100.0*(S(NA)-SOL(NA))/SOL(NA)
      WRITE(6,7)T,S(NA),SOL(NA),DIF(NA),XEM(NA)
    7 FORMAT(10X,F5.2,4F15.6)
   62 CONTINUE
      WRITE(6,15)
   15 FORMAT(//)
      SS1=0.
      DO 97 NA=1,MTOT
      SS1=SS1+ABS(DIF(NA))
      S(NA) = (S(NA)+XEM(NA))*XNUM2
      XEM(NA) = SQRT(XEM(NA)/XNUM3)*XNUM
   97 CONTINUE
      S1=SS1*0.1
      WRITE(6,94)S1
   94 FORMAT(20X,'PER CENT ERROR = ',F20.8)
      II=NUM
   82 CONTINUE
      STOP
      END
```

```
      SUBROUTINE XEQN(XD,XMD,RT)
       DIMENSION XD(2),XMD(2)
      XMD(1)=XD(2)
      XMD(2)=-2.0*XD(1)-3.0*XD(2)+RT
      RETURN
      END
```

APPENDIX B

THE COMPUTER SOFTWARE PACKAGE APPLIED TO THE LARGE-SCALE MISSILE SYSTEM

This appendix includes the implemented computer software package on the thirty-first order math model of a six degree-of-freedom air defense missile system. In addition to the modification of the original program, the six subprograms which have been implemented are COEFF, COVAR, RUNGKP, MDERIV, SNOISE, and DETARA.

The main program initializes all the coefficient matrix elements, covariance matrix elements, and other variables used in the program. The SYSINT subprogram updates the nonlinear terms of the coefficient matrix, enters Subprogram COEFF to evaluate the coefficients for the implicitly related variables, and calls the COVAR subprogram where the covariance differential equations are calculated. These equations are then integrated by entering RUNGKP from SYSINT. The Subprograms SNOISE and DETARA are used to calculate the variance of the noise introduced in the SEEKER program.

```
C ***
C      TERMINAL HOMING - ALL DIGITAL SIMULATION
C ***
C
C *** BLANK COMMON HOUSES AERODYNAMIC COEFFICIENTS AND DERIVATIVES IN
C *** TABULAR FORM FOR USE BY THE 1, 2, AND 3 VARIATE LOOK UP SCHEME.
C
       COMMON DXDYDZ(60),IADD(20),AERO(1360)
C
C *** COMMON BLOCK /TIMES CONTAINS CURRENT TIME, STEP LENGTH AND OTHER
C *** EVENT TIMES IN THE SIMULATION.
C
       COMMON /TIMES/T,DT,TBG,TSTOP,IPR,J,LAUNCH
       DOUBLE PRECISION T,DT
C
C *** COMMON BLOCK /CNTRL/ CONTAINS CARD INPUT DATA WHICH CONTROLS
C *** PROGRAM SELECTION (MODULE TEST OR SYSTEM RUN) AND MODULE TEST
C *** DATA(WHEN MODE=2)
C
       COMMON /CNTRL/MODE, MDLS(4),IV,DATAM(16,4)
C
C *** COMMON BLOCK /AUTOP/ CONTAINS INTEGRATION VARIABLES, DERIVATIVES
C *** AND INTERMEDIATE VARIABLES REQUIRED BY THE AUTOPILOT MODULE
C
       COMMON /AUTOP/NA,VA(15),DVA(15),OV(7)
C
C *** COMMON BLOCK /SEEKR/ CONTAINS INTEGRATION VARIABLES, DERIVATIVES
C *** AND INTERMEDIATE VARIABLES REQUIRED BY TNE AUTOPILOT MODULE
       COMMON / SEEKR/ NS,VS(2),DVS(2),OSV(8)
C
C *** COMMON BLOCK /VANES/ CONTAINS INTEGRATION VARIABLES AND DERIVATIVES
C *** REQUIRED IN THE VANE ANGLE CALCULATION MODULE
C
       COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
C
C *** COMMON BLOCK /ROTATE/ CONTAINS ROTATIONAL VARIABLES AND DERIVATIVES
C *** USED IN THE MISSILE MODULE
C
       COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
      1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
C
C *** COMMON BLOCK /STATEV/ CONTAINS TRANSLATIONAL VARIABLES AND
C *** DERIVATIVES
C
       COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
C
C *** COMMON BLOCK /ADDV/   CONTAINS ADDITIONAL VARIABLES DERIVED FROM
C *** THE STATE (INTEGRATION) VARIABLES
C
       COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VSS,RHO
C
C *** COMMON BLOCK /COEFS/ CONTAINS THE THRUST AND AERDYNAMIC
C *** COEFFICIENTS AND DERIVATIVES  OBTAINED BY TABLE INTERPOLATION
C
```

```
      COMMON /COEFS/THR,AERC(18)
C
C *** COMMON BLOCK CONTAINS AIRFRAME CONSTANTS GOVERNING AERODYNAMIC
C *** FORCES AND THRUST MISALIGNMENT
C
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
C
C *** COMMON BLOCK /MSINCG/ CONTAINS MASS, INERTIAS AND CG POSITION OF
C *** THE AIRFRAME PLUS THE CONSTANT VALUES FROM WHICH THEY ARE OBTAINED
C
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
C
C *** COMMON BLOCK /FCEMOM/ CONTAINS THE AERODYNAMIC FORCES, MOMENTS,
C *** AND THRUST MISALIGNMENT COMPONENTS
C
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
C
C *** COMMON BLOCK /INCEPT/ CONTAINS TARGET POSITION AND VELOCITY,
C *** TARGET-MISSILE INTERCEPT SPEED AND RANGE AND INPUTS TO THE SEEKER
C
      COMMON /INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
C
C *** COMMON BLOCK /TRANSF/ CONTAINS MATRICES FOR CONVERSION FROM
C *** VARIOUS COORDINATE SYSTEMS TO OTHERS
C
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
C
C *** COMMON BLOCK CONTAINS UTILITY VALUES SUCH AS GRAVITY ACC. AND
C *** RADIANS TO DEGREES CONSTANTS.
C
      COMMON /UTILTY/G,RTD
      COMMON /VMG/ H,MS
      COMMON /BLOCK1/P(31,31),DP(31,31)
      COMMON /BLOCK2/ A2(31,31),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON / BLOCK7/KK3,THRP,TIMP
      COMMON /BLOCK8/KK1,KK5,VP
      COMMON /BLOCK9/C2(84,31),KOK
      COMMON /BLIK2/ AVD(4),BVD(4)
      COMMON /SNSE/ AREA(31)
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      DIMENSION LBL(10)
C
C *** READ THRUST AND AERODYNAMIC TABLES FROM CARDS
C
      KOUNT = 0
      KICK = 20
      KIK = 1
      KOK = 0
      K400 = 0
      KK1 = 1
      KK3 = 0
```

```
      KK5 = 0
      VP = 1.0
      B2(2) = 1.0
      B2(1) = 1.0
      TMVEL = -0.10
      TMRNGE = 10000.1
      DO 88 I=1,4
      AVD(I) = 0.0
88    BVD(I) = 0.0
      DO 1 I=1,84
      DO 1 K=1,31
1     C2(I,K) = 0.0
      DO 29 I=1,MS
      DO 29 K=1,MS
      A2(I,K) = 0.
      DP(I,K) = 0.
29    P(I,K) =0.
      TMP1 = WQ1*WQ1
      TMP2 = 2.*DQ1*WQ1
      TMP3 = PYAK1*PYBK1
      TMP4 = PYAK1+PYBK1
      TMP5 = WQG*WQG
      TMP6 = 2.*DQG*WQG
      TMP7 = PYIK1*WQ1*WQ1/TMP3
C
C *** CONSTANT 'A' MATRIX ELEMENTS
C
      A2(1,1) = -3.*TAUZ
      A2(1,2) = TAUZ*A2(1,1)
      A2(1,3) =-TAUZ*TAUZ*TAUZ
      A2(2,1) = 1.
      A2(3,2) = 1.
      A2(4,4) = -3.*TAUY
      A2(4,5) = TAUY*A2(4,4)
      A2(4,6) = -TAUY*TAUY*TAUY
      A2(5,4) = 1.
      A2(6,5) = 1.
      A2(7,7) = -2.*DP1*WP1
      A2(7,8) = -WP1*WP1
      A2(7,26) = -A2(7,8)*RTD
      A2(8,7) = 1.
      A2(10,2) = -TMP7
      A2(10,3) = -TMP7*TAUL
      A2(10,5) = TMP7
      A2(10,6) = -A2(10,3)
      A2(10,10) =-TMP2
      A2(10,11) = -TMP1
      A2(10,23)=RTD*TMP7
      A2(10,24) =-A2(10,23)
      A2(11,10) = 1.
      A2(12, 2) = A2(10, 2)
      A2(12, 3) = A2(10, 3)
      A2(12, 5) = A2(10, 5)
      A2(12, 6) = A2(10, 6)
```

```
         A2(12,10) = TMP4+A2(10,10)
         A2(12,11) = TMP3+A2(10,11)
         A2(12,23) = A2(10,23)
         A2(12,24) = A2(10,24)
         A2(13,2) = -TMP7
         A2(13,3) = -TMP7*TAUL
         A2(13,5) = -TMP7
         A2(13,6) = A2(13,3)
         A2(13,13) = -TMP2
         A2(13,14) = -TMP1
         A2(13,23) = A2(12,23)
         A2(13,24) = A2(13,23)
         A2(14,13) = 1.
         A2(15, 2) = A2(13, 2)
         A2(15, 3) = A2(13, 3)
         A2(15, 5) = A2(13, 5)
         A2(15, 6) = A2(13, 6)
         A2(15,13) = TMP4+A2(13,13)
         A2(15,14) = TMP3+A2(13,14)
         A2(15,23) = A2(13,23)
         A2(15,24) = A2(13,24)
         A2(19,16) =1.0
         A2(20,17) =1.0
         A2(21,18) =1.0
         A2(26,22) = 1.0
         READ(5,62)(AREA(I),I=1,30)
         AREA(31) = 0.0
         WRITE (6  ,900)
         KNT1 = 1
         KNT2 = 3
         IL = 1
   30    READ (  5,910) I,J,K,(CXDYDZ(L),L=KNT1,KNT2),LBL
         IF (I.EQ.999)GO TO 40
         WRITE (6  ,920) LBL
         KNT1 = KNT2+1
         L = KNT2/3
         IADD(L) = IL
         KNT2 = KNT2+3
         IF (J.EQ.0)J=1
         IF (K.EQ.0)K=1
         IU = I*J*K+IL-1
         READ (  5,930) (AERO(L),L=IL,IU)
         IL = IU+1
         GO TO 30
   40    CONTINUE
         WP = PB*RTD
         WQ = QB*RTD
         WR = RB*RTD
C
C *** CALL INITIA TO INITIALIZE THE PROGRAM AND READ RUN DATA
C
         CALL INITIA
C
C *** CALL TOTAL SYSTEM RUN CONTROL ROUTINE
```

```
C
      CALL SYSRUN
      STOP
62    FORMAT(10F8.6)
 900  FORMAT (1H1, 50X,'T-H AERODYNAMIC TABLES')
 910  FORMAT (3I3, 1X,3F10.0, 10A4)
 920  FORMAT (/45X,10A4)
 930  FORMAT (8F10.0)
      END

      SUBROUTINE INITIA
C ***
C     THIS ROUTINE READS VARIOUS RUN DATA FROM CARDS AND INITIALIZES
C     THE REMAINDER OF THE PROGRAM
C ***
      COMMON /CNTRL/MODE,MDLS(4),IV,DATAM(16,4)
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J ,LAUNCH
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI
      COMMON /INCEPT/UT(3),XT(3)
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      DOUBLE PRECISION T,DT
      CALL INTHRC
      CALL INTRAN
      CALL INAUPT
      READ (  5,900) MODE,MDLS,IV,IT,ITCG,IRAIL,IWIND
      GO TO(20,30),MODE
20    READ(  5,930) (DATAM(J,1),J=1,16),(DATAM(J,2),J=1,4)
      READ (  5,940)DT,TSTOP,IPR
      IF(IV.NE.0)READ(  5,910)UE,VE,WE,X ,Y,Z,PB,QB,RB,THETA,PHI,PSI
      IF (IT.NE.0)READ(  5,910) UT,XT
      IF(ITCG.NE.0)READ(  5,910)XTCG,YTCG,ZTCG
      IF(IRAIL.NE.0)READ(  5,910)RL1,RL2
      IF(IWIND.NE.0)READ(  5,910) WUE,WVE,WWE
      RETURN
30    DO 40 I=1,4
      IF (MDLS(I).EQ.0)GO TO 40
      READ(  5,920)  DATAM(1,I)
      READ(  5,910) (DATAM(J,I),J=2,16)
40    CONTINUE
      RETURN
900   FORMAT(16I5)
910   FORMAT(8F10.0)
920   FORMAT(F20.0)
930   FORMAT(20A4)
940   FORMAT(2F10.0,I10)
      END
```

```
      SUBROUTINE SYSINT
C **
C     THIS ROUTINE INTEGRATES ALL EQUATIONS OVER 1 TIME STEP
C ***
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J ,LAUNCH
      COMMON /STATEV/NT,VT(6),DVT(6)
      COMMON /ROTATE/NR,VR(6),DVR(6),SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI
     1,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /SEEKR/ NS,VS(2),DVS(2),DSV(8)
      COMMON /AUTOP/NA,VA(15),DVA(15),DVAD(7)
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY
     1,RLCG,RDCG
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /VMG/ H,MS
      COMMON /BLOCK1/P(31,31),DP(31,31)
      COMMON /BLOCK2/ A2(31,31),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /BLOCK4/ VV5(4),DLTC(4)
      COMMON /BLOK1/DTH
      COMMON /BLIK1/BPHISM
      COMMON /BLIK2/ AVD(4),BVD(4)
      DOUBLE PRECISION T,DT, HALFDT
      DIMENSION QT(12),QR(12),QA(30),QV(8)
      DO 40 KUT = 1,4
      KAT = KUT
      GO TO (30,10,20,10),KUT
10    T = T+HALFDT
      GO TO (15,20),J
15    CALL THRCON
20    CALL AUTOPT
      CALL VANEMD
      CALL TRANSM
      CALL ROTATM
30    CALL RK4(NA,VA,QA,KUT)
      CALL RK4(NV,VV,QV,KUT)
      CALL RK4(NT,VT,QT,KUT)
      CALL RK4(NR,VR,QR,KUT)
   40 CONTINUE
      CALL AUTOPT
      CALL VANEMD
      CALL TRANSM
      CALL ROTATM
      IF(T.LE.1.0)GO TO 1001
      KOUNT = KOUNT+1
C
C *** NONLINEAR 'A' MATRIX ELEMENTS
C
      IF(ABS(BPHISM).GE.( RLIM-0.001)) GO TO 12
      A2(9,7) = RK1*(RA1+RB2+A2(7,7))/RA1/RB2
      A2(9,8) = RK1*(1.+A2(7,8))/RA1/RB2
      A2(9,26) = RK1*A2(7,26)/RA1/RB2
      GO TO 13
```

```
12    A2(9,7) =0.0
      A2(9,8) =0.0
      A2(9,26) = 0.0
13    IF(ABS(VA(12)).GE.(PYLIM-0.001)) GO TO 22
      A2(28,12) = VGAIN
      A2(30,12) = VGAIN
      GO TO 23
22    A2(28,12) =0.0
      A2(30,12) =0.0
23    IF(ABS(VA(15)).GE.(PYLIM-0.001)) GO TO 32
      A2(29,15) = VGAIN
      A2(31,15) = VGAIN
      GO TO 33
32    A2(29,15) =0.0
      A2(31,15) =0.0
33    IF(ABS(VV(1)) .GE.( VLIM-0.001)) GO TO 42
      A2(28,28) = -VGAIN
      GO TO 43
42    A2(28,28) =0.0
43    IF(ABS(VV(2)) .GE.( VLIM-0.001)) GO TO 52
      A2(29,29) = -VGAIN
      GO TO 53
52    A2(29,29) = 0.0
53    IF(ABS(VV(3)) .GE.( VLIM-0.001)) GO TO 62
      A2(30,30) = -VGAIN
      GO TO 63
62    A2(30,30) =0.0
63    IF(ABS(VV(4)) .GE.( VLIM-0.001)) GO TO 72
      A2(31,31) =-VGAIN
      GO TO 73
72    A2(31,31) =0.0
73    CONTINUE
      IF(ABS(AVD(1)).GE.(VRLIM-0.001)) GO TO 83
      A2(28,7) = A2(9,7)*VGAIN
      A2(28,8) = A2(9,8)*VGAIN
      A2(28,9) = 0.1*VGAIN
      A2(28,26) =A2(9,26)*VGAIN
      GO TO 84
83    A2(28,7) = 0.0
      A2(28,8) = 0.0
      A2(28,9) = 0.0
      A2(28,12) =0.0
      A2(28,26) =0.0
      A2(28,28) =0.0
84    IF(ABS(AVD(2)).GE.(VRLIM-0.001)) GO TO 93
      A2(29,7) = A2(28,7)
      A2(29,8) = A2(28,8)
      A2(29,9) = A2(28,9)
      A2(29,26) = A2(28,26)
      GO TO 94
93    A2(29,7) = 0.0
      A2(29,8) = 0.0
      A2(29,9) = 0.0
      A2(29,15) =0.0
```

```
      A2(29,26) =0.0
      A2(29,29) =0.0
94    IF(ABS(AVD(3)).GE.(VRLIM-0.001)) GO TO 103
      A2(30, 7) =-A2(28,7)
      A2(30, 8) =-A2(28,8)
      A2(30, 9) =-A2(28,9)
      A2(30,26) =-A2(28,26)
      GO TO 104
103   A2(30,7) = 0.0
      A2(30, 8) =0.0
      A2(30, 9) =0.0
      A2(30,12) =0.0
      A2(30,26) =0.0
      A2(30,30) =0.0
104   IF(ABS(AVD(4)).GE.(VRLIM-0.001)) GO TO 113
      A2(31, 7) =A2(30,7)
      A2(31, 8) =A2(30,8)
      A2(31, 9) =A2(30,9)
      A2(31,26) = A2(30,26)
      GO TO 114
113   A2(31,7) = 0.0
      A2(31, 8) =0.0
      A2(31, 9) =0.0
      A2(31,15) = 0.0
      A2(31,26) =0.0
      A2(31,31) =0.0
114   IF(BVD(1).LE.0.0)GO TO 133
      A2(28,7) = 0.0
      A2(28,8) = 0.0
      A2(28,9) = 0.0
      A2(28,12) =0.0
      A2(28,26) =0.0
      A2(28,28) =0.0
133   IF(BVD(2).LE.0.0)GO TO 143
      A2(29,7) = 0.0
      A2(29,8) = 0.0
      A2(29,9) = 0.0
      A2(29,15) =0.0
      A2(29,26) =0.0
      A2(29,29) =0.0
143   IF(BVD(3).LE.0.0)GO TO 153
      A2(30,7) = 0.0
      A2(30, 8) =0.0
      A2(30, 9) =0.0
      A2(30,12) =0.0
      A2(30,26) =0.0
      A2(30,30) =0.0
153   IF(BVD(4).LE.0.0)GO TO 163
      A2(31,7) = 0.0
      A2(31, 8) =0.0
      A2(31, 9) =0.0
      A2(31,15) = 0.0
      A2(31,26) =0.0
      A2(31,31) =0.0
```

```
163     CONTINUE
        A2(25,23) =CSPHI
        A2(25,24) =-SNPHI
        A2(27,23) = SNPHI/CSTHA
        A2(27,24) = CSPHI/CSTHA
        A2(26,23) = SNTHA*A2(27,23)
        A2(26,24) = SNTHA*A2(27,24)
        A2(25,26) = -VR(2)*SNPHI-VR(3)*CSPHI
        A2(26,25) = -A2(25,26)/(CSTHA*CSTHA)
        A2(27,26) = (-VR(3)*SNPHI+VR(2)*CSPHI)/CSTHA
        A2(26,26) =A2(27,26)*SNTHA
        A2(27,25) =-A2(26,25)*SNTHA
        IF(KOUNT.NE.10)GO TO 1
        KCUNT = 0
        CALL COEFF
1       DTH = SNGL(DT)
        DTH = DTH/2.0
        DO 2  IK=1,2
        KAT = 1
        DO 2 IJ=1,2
        CALL COVAR
        CALL RUNGKP
2       KAT = KAT + 1
        DO 29 II=1,31
        IF(P(II,II).GE.1.0E-10)GO TO 29
        DO 28 IJ=1,31
        P(II,IJ) = 0.0
 28     CONTINUE
29      CONTINUE
        IF(KICK.NE.20) GO TO 299
        WRITE(6,124)T
124     FORMAT(1X,'TIME = ',F6.4)
        DO 288 I=1,MS
288     WRITE(6,11)I,(P(I,K),K=1,I)
11      FORMAT(//1X,'P(',I2,', J) =',7E15.5/4(11X,7E15.5/))
        KICK = 0
299     CONTINUE
        KICK = KICK + 1
        RETURN
        ENTRY INSYST
        HALFDT = .5D+0*DT
1001    RETURN
        END
```

```
      SUBROUTINE SNOISE(TMP1,BEPS,EC,SGSQ)
      COMMON / SEEKK/SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /UTILTY/G,RTD
      COMMON /BLOCK1/P(31,31),DP(31,31)
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS
      SIGBEP = 0.15/RTD
      IF(EC.NE.-SKSP) GO TO 21
      DIST = -TMP1 - BEPS
      POS = DIST/SIGBEP
      CALL DETARA (POS,AL1)
      AL = AL1+ 0.5
      POS = POS + 2.0*TMP1
      CALL DETARA (POS,AO1)
      AO = AO1 - AL1
      AU = 1.0 - AL - AO
      GO TO 41
   21 IF(EC.NE.0.0) GO TO 22
      DIST = BEPS + TMP1
      POS = DIST/SIGBEP
      CALL DETARA (POS,AO1)
      AL = 0.5 - AO1
      POS = TMP1 - BEPS
      CALL DETARA (POS,AO2)
      AU = 0.5 - AO2
      AO = AO1 + AO2
      GO TO 41
   22 DIST = BEPS - TMP1
      POS = DIST/SIGBEP
      CALL DETARA (POS,AU1)
      AU = AU1+ 0.5
      POS = POS + 2.0*TMP1
      CALL DETARA(POS,AO1)
      AO = AO1 - AU1
      AL = 1.0 - AU - AO
   41 SIGEC =          AL*(-SKSP) + AU*SKSP
      SGSEC = (AU+AL)*SKSP*SKSP
      SGSQ = SGSEC - SIGEC*SIGEC
      RETURN
      END
```

```
      SUBROUTINE COVAR
      COMMON /VMG/ H,MS
      COMMON /BLOCK1/P(31,31),DP(31,31)
      COMMON /BLOCK2/ A2(31,31),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      DOUBLE PRECISION T,DT
      DIMENSION A3(15),P3(15)
      DO 25 I=1,MS
      DO 25 JJ=1,I
25    P(JJ,I) = P(I,JJ)
      DO 1 JK=1,3
      A3(JK) = A2(1,JK)
1     A3(JK+3)=A2(1,JK+18)
      DO 3 I = 1,1
      DO 2 JK=1,3
      P3(JK) = P(JK,I)
2     P3(JK+3)=P(JK+18,I)
      DP(1,I) = 0.
      DO 3 JK = 1,6
3     DP(1,I) = DP(1,I) + A3(JK)*P3(JK)
      DO 4 JI=1,2
      DP(2,JI) = A2(2,1)*P(1,JI)
4     DP(3,JI) = A2(3,2)*P(2,JI)
      DP(3,3) = A2(3,2)*P(2,3)
      DO 5 JK=1,3
      A3(JK) = A2(4,JK+3)
5     A3(JK+3)=A2(4,JK+18)
      DO 7 I=1,4
      DO 6 JK=1,3
      P3(JK) = P(JK+3,I)
6     P3(JK+3)=P(JK+18,I)
      DP(4,I) = 0.
      DO 7 JK=1,6
7     DP(4,I) = DP(4,I)+A3(JK)*P3(JK)
      DO 8 JI=1,5
8     DP(5,JI) = A2(5,4)*P(4,JI)
      DO 40 JI=1,6
40    DP(6,JI) = A2(6,5)*P(5,JI)
      DO 41 JI=1,7
41    DP(7,JI) = A2(7,7)*P(7,JI)+A2(7,8)*P(8,JI)+A2(7,26)*P(26,JI)
      DO 42 JI=1,8
42    DP(8,JI) = A2(8,7)*P(7,JI)
      DO 43 JI=1,9
43    DP(9,JI) = A2(9,7)*P(7,JI) + A2(9,8)*P(8,JI)+A2(9,26)*P(26,JI)
      DO 44 JI=1,11
44    DP(11,JI) = A2(11,10)*P(10,JI)
      DO 45 JI=1,14
45    DP(14,JI) = A2(14,13)*P(13,JI)
      DO 9 I = 10,12,2
      DO 9 JI = 1,I
9     DP(I,JI) = A2(I,2)*P(2,JI)+A2(I,3)*P(3,JI)+A2(I,5)*P(5,JI)+A2(I,6)
     1*P(6,JI)+A2(I,10)*P(10,JI)+A2(I,11)*P(11,JI)+A2(I,23)*P(23,JI)+
```

```
      2A2(I,24)*P(24,JI)
       DO 10 I=13,15,2
       DO 10 JI=1,I
   10  DP(I,JI) = A2(I,2)*P(2,JI)+A2(I,3)*P(3,JI)+A2(I,5)*P(5,JI)+A2(I,6)
      1*P(6,JI)+A2(I,13)*P(13,JI)+A2(I,14)*P(14,JI)+A2(I,23)*P(23,JI)
      2+A2(I,24)*P(24,JI)
       JL = 16
       JM = 18
       KIT = 0
   17  DO 11 I=JL,JM
       DO 12 JK=1,3
   12  A3(JK) = A2(I,JK+15)
       A3(4) = A2(I,21)
       DO 13 JK=5,11
   13  A3(JK) = A2(I,JK+20)
       DO 11 II=1,I
       DO 14 JK=1,3
   14  P3(JK) = P(JK+15,II)
       P3(4) = P(21,II)
       DO 15 JK=5,11
   15  P3(JK) = P(JK+20,II)
       DP(I,II) = 0.
       DO 11 JK=1,11
   11  DP(I,II) = DP(I,II) + A3(JK)*P3(JK)
       IF(KIT.EQ.1) GO TO 16
       KIT = 1
       JL = 22
       JM = 24
       GO TO 17
   16  DO 18 I=1,22
   18  DP(22,I) = DP(22,I)+A2(22,22)*P(22,I)
       DO 19 JK=23,24
       DO 19 I=1,JK
   19  DP(JK,I) = DP(JK,I)+A2(JK,22)*P(22,I)+A2(JK,23)*P(23,I)+A2(JK,24)
      1*P(24,I)
       II = 16
       DO 20 JK=19,21
       DO 26 I=1,JK
   26  DP(JK,I) = A2(JK,II)*P(II,I)
       II=II+1
   20  CONTINUE
       DO 21 JK=25,27
       DO 21 I=1,JK
   21  DP(JK,I) = A2(JK,23)*P(23,I)+A2(JK,24)*P(24,I)+A2(JK,26)*P(26,I)
       DO 22 I=1,26
   22  DP(26,I) = DP(26,I)+A2(26,22)*P(22,I)+A2(26,25)*P(25,I)
       DO 83 I=1,27
   83  DP(27,I) = DP(27,I) + A2(27,25)*P(25,I)
       JL = 28
       JI = 12
       DO 23 JK=28,31
       IF(JK.EQ.30)JI=12
       DO 27 I=1,JK
   27  DP(JK,I) = A2(JK,7)*P(7,I)+A2(JK,8)*P(8,I)+A2(JK,9)*P(9,I) +
```

```
      1A2(JK,26)*P(26,I)+A2(JK,JI)*P(JI,I)+A2(JK,JL)*P(JL,I)
       JL = JL+1
       JI = JI+3
 23    CONTINUE
       IF(LAUNCH.GT.2)GO TO 81
       DO 82 JK=17,18
       DO 82 I=1,JK
 82    DP(JK,I)=DP(JK,I)+A2(JK,22)*P(22,I) +A2(JK,23)*P(23,I) +A2(JK,24)*
      1P(24,I)
 81    DO 24 II=1,MS
       DO 24 JJ=1,II
 24    DP(II,JJ) =DP(II,JJ)+DP(II,JJ)
       DP(1,1) = DP(1,1) + EZNOIS*B2(1)*B2(1)
       DP(4,4) = DP(4,4) + EYNOIS*B2(2)*B2(2)
       DP(28,28) = DP(28,28) + VGAIN*VGAIN*0.25
       DP(29,29) = DP(29,29) + VGAIN*VGAIN*0.25
       DP(30,28) = DP(30,28) + VGAIN*VGAIN*0.25
       DP(30,30) = DP(30,30) + VGAIN*VGAIN*0.25
       DP(31,29) = DP(31,29) + VGAIN*VGAIN*0.25
       DP(31,31) = DP(31,31) + VGAIN*VGAIN*0.25
       RETURN
       END
```

```
       SUBROUTINE RUNGKP
       COMMON /VMG/ H,MS
       COMMON /BLOCK1/P(31,31),DP(31,31)
       COMMON /BLOCK2/ A2(31,31),KIK,KOUNT,KICK,KAT
       COMMON /BLOK1/DTH
       DIMENSION P1(31,31),DP1(31,31)
       GO TO(10,30),KAT
 10    DO 20 I=1,MS
       DO 20 J=1,I
       P1(I,J) = P(I,J)
       DP1(I,J) = DP(I,J)
 20    P(I,J) = P(I,J) + DTH*DP(I,J)
       RETURN
 30    VDT = DTH/2.0
       DO 40 I=1,MS
       DO 40 J=1,I
 40    P(I,J) = P1(I,J) + VDT*(DP1(I,J) + DP(I,J))
       RETURN
       END
```

```
      SUBROUTINE COEFF
C ***
C     THIS SUBROUTINE CALCULATES THE IMPLICIT 'A' MATRIX ELEMENTS
C ***
      COMMON / SEEKR/NS,BTHTG,BPSIG,BTHD,BPSD,EZ,EY,OSV(6)
      COMMON / INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON / AUTOP/NA,ZP1,ZP2,ZP3,ZY1,ZY2,ZY3,ZR1,ZR2,BPHIS,ZPI1,ZPI2,
     1EODCR,ZYI1,ZYI2,EVNCR,ZPD1,ZPD2,ZPD3,ZYD1,ZYD2,ZYD3,ZRD1,ZRD2,
     2BPHISD,ZPID1,ZPID2,EODCRD,ZYID1,ZYID2,EVNCRD,EZSS,EYSS,WQC,WRC,
     3EZRR,EYRR,BDELPC
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /STATEV/NT,UE,VE,WE,X(3),DUE,DVE,DWE,DX,DY,DZ
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WF,WQ,WR,BTHETA,BPH,BPS
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /VANEK/VGAIN,VLIM,VRLIM
      COMMON /COEFS/THR,CMQ,CNR,CNP,CY2,CL3,CXO,CMO,CDCM,CNF,CN2,
     1CLP,CL2,CXC,CNQ,CMDQP,CLDRP,CMR,CLD
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VA,RHO
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /UTILTY/G,RTD
      COMMON /VMG/ H,MS
      COMMON /BLOCK1/P(31,31),DP(31,31)
      COMMON /BLOCK2/ A2(31,31),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /BLOCK6/ BACS(3)
      COMMON / BLOCK7/KK3,THRP,TIMP
      COMMON /BLOCK8/KK1,KK5,VP
      COMMON /BLOCK9/C2(84,31),KOK
      COMMON /BLOCC1/DUE1,DVE1,DWE1,DPB1,DQB1,DRB1
      DOUBLE PRECISION T, DT
      DIMENSION X6(3),BCSEC1(3,3),ECSBC1(3,3),VV1(4),DEL1(3)
      KK1 = 0
      KK2 = 1
      KK3=7
      KK4 = 1
      KK6 = 1
      UE1 = UE
      PP1 = SQRT(ABS(P(16,16)))
      UE = UE + 0.1
      IF(PP1.GT.0.1 )  UE  =  UE1 + 0.1*PP1
      GO TO 143
643   DO 2 I = 1,3
2     DEL1(I) = DEL(I)
      IF(ABS(VV(II)) .LE.VLIM)GO TO 3
      VV(II) = XLIMIT(VV(II),VLIM)
3     TMP1 = VV(1)+VV(2)
      TMP2 = VV(3)+VV(4)
      DEL(1) = 0.25*(TMP1+TMP2)
```

```
      DEL(3) = 0.25*(TMP2-TMP1)
      DEL(2) = 0.25*(VV(2)+VV(4)-VV(1)-VV(3))
      KK1 = 0
      GO TO 343
543   SNTHA1 = SNTHA
      CSTHA1 = CSTHA
      SNPHI1 = SNPHI
      CSPHI1 = CSPHI
      SNPSI1 = SNPSI
      CSPSI1 = CSPSI
      DO 7 II=1,3
      DO 7 JJ=1,3
      BCSEC1(II,JJ) = BCSECS(II,JJ)
7     ECSBC1(II,JJ) = ECSBCS(II,JJ)
143   RHO1 = RHO
      VP1 = VP
      VA1 = VA
      QUE1 = QUE
      XMN1 = XMN
      ALFA1 = ALFA
      BETA1 = BETA
      ALFAP1 = ALFAP
      CSPHI1 = CSPHIP
      SNPHI1 = SNPHIP
      THRP1 = THRP
      TIMP1 = TIMP
      XM1 = XM
      XIX1 = XIX
      XIY1 = XIY
      RDCG1 = RDCG
      THR1 = THR
      CMQ1 = CMQ
      CNR1 = CNR
      CNP1 = CNP
      CY21 = CY2
      CL31 = CL3
      CXO1 =CXO
      CMO1 = CMO
      CDCM1 =CDCM
      CNF1 = CNF
      CN21 = CN2
      CLP1 = CLP
      CL21 = CL2
      CXC1 = CXC
      CNQ1 = CNQ
      CLDRP1 = CLDRP
      CMDQP1 = CMDQP
      CMR1 = CMR
      CLD1 = CLD
343   FXA1 = FXA
      FYA1 = FYA
      FZA1 = FZA
      XMXA1 = XMXA
      XMYA1 = XMYA
```

```
      XMZA1 = XMZA
      FTHX1 = FTHX
      FTHY1 = FTHY
      FTHZ1 = FTHZ
      CALL TRANSM
      IF(KK2.EQ.2)GO TO 22
      CALL THRCON
      GO TO 22
144   A2(I1,J1) = (DUE1-DUE)/ZZ1
      A2(I1+1,J1) = (DVE1-DVE)/ZZ1
      A2(I1+2,J1) = (DWE1-DWE)/ZZ1
      A2(I1+6,J1)= (DPB1-DPB)/ZZ1
      A2(I1+7,J1)= (DQB1-DQB)/ZZ1
      A2(I1+8,J1)= (DRB1-DRB)/ZZ1
      IF(KK3.EQ.7)GO TO 155
      IF(KK1.EQ.1)GO TO 555
      DO 4 I = 1,3
4     DEL(I) = DEL1(I)
      GO TO 355
555   SNTHA = SNTHA1
      CSTHA  = CSTHA1
      SNPHI  = SNPHI1
      CSPHI  = CSPHI1
      SNPSI  = SNPSI1
      CSPSI  = CSPSI1
      DO 171 II=1,3
      DO 171 JJ=1,3
      BCSECS(II,JJ) = BCSEC1(II,JJ)
171   ECSBCS(II,JJ) = ECSBC1(II,JJ)
155   RHO  = RHO1
      VP = VP1
      VA=VA1
      QUE = QUE1
      XMN  = XMN1
      ALFA  = ALFA1
      BETA  = BETA1
      ALFAP  = ALFAP1
      CSPHIP = CSPHI1
      SNPHIP = SNPHI1
      THRP  = THRP1
      TIMP  = TIMP1
      XM = XM1
      XIX = XIX1
      XIY = XIY1
      RDCG  = RDCG1
      THR = THR1
      CMQ = CMQ1
      CNR = CNR1
      CNP = CNP1
      CY2 = CY21
      CL3 = CL31
      CX0 = CX01
      CM0 = CM01
      CDCM = CDCM1
```

```
      CNF = CNF1
      CN2 = CN21
      CLP = CLP1
      CL2 = CL21
      CXC = CXC1
      CNQ= CNQ1
      CLDRP= CLDRP1
      CMDQP = CMDQP1
      CMR = CMR1
      CLD = CLD1
355   FXA = FXA1
      FYA = FYA1
      FZA = FZA1
      XMXA = XMXA1
      XMYA = XMYA1
      XMZA= XMZA1
      FTHX  = FTHX1
      FTHY  = FTHY1
      FTHZ  = FTHZ1
      GO TO(143,543,643,343,57),KK6
64    ZZ1 = UE -UE1
      KK4 = 2
      UE = UE1
      VE1 = VE
      PP1 = SQRT(ABS(P(17,17)))
      VE = VE + 0.001
      IF(PP1.GT..001)  VE  =  VE1 + 0.1*PP1
      I1 = 16
      J1 = 16
      GO TO 144
44    ZZ1=VE-VE1
      KK4 = 3
      WE1 = WE
      VE = VE1
      PP1 = SQRT(ABS(P(18,18)))
      WE = WE + 0.1
      IF(PP1.GT.0.1 )  WE  =  WE1 + 0.1*PP1
      J1 = 17
      GO TO 144
45    ZZ1 = WE-WE1
      KK4 = 4
      X6(3) = X(3)
      WE=WE1
      PP1 = SQRT(ABS(P(21,21)))
      X(3) = X(3) + 1.0
      IF(PP1.GT.1.0 ) X(3) =X6(3) + 0.1*PP1
      J1 = 18
      GO TO 144
46    ZZ1 = X(3)-X6(3)
      KK4 = 5
      THETA1 = THETA
      X(3) = X6(3)
      PP1 = SQRT(ABS(P(25,25)))
      THETA = THETA + 0.01
```

```
      IF(PP1.GT.0.01)THETA =THETA1+ 0.1*PP1
      KK1 = 1
      KK6 = 2
      J1 = 21
      GO TO 144
47    ZZ1 = THETA-THETA1
      KK4 = 6
      THETA = THETA1
      PSI1 = PSI
      PP1 = SQRT(ABS(P(27,27)))
      PSI = PSI + 0.01
      IF(PP1.GT.0.01) PSI  = PSI1 + 0.1*PP1
      KK3 = 6
      J1 = 25
      GO TO 144
48    ZZ1 = PSI-PSI1
      KK4 = 7
      PHI1 = PHI
      PSI = PSI1
      PP1 = SQRT(ABS(P(26,26)))
      PHI = PHI + 0.01
      IF(PP1.GT.0.01) PHI  = PHI1 + 0.1*PP1
      J1 = 27
      GO TO 144
49    ZZ1 = PHI-PHI1
      KK4 = 8
      PHI = PHI1
      VV1(1) = VV(1)
      PP1 = SQRT(ABS(P(28,28)))
      VV(1) = VV(1) + 0.1
      IF(PP1.GT.0.1 ) VV(1)=VV1(1)+ 0.1*PP1
      KK5 = 1
      KK2 = 2
      KK6 = 3
      II = 1
      J1 = 26
      GO TO 144
50    ZZ1 = VV(1)-VV1(1)
      KK4 = 9
      VV(1) = VV1(1)
      VV1(2) = VV(2)
      PP1 = SQRT(ABS(P(29,29)))
      VV(2) = VV(2) + 0.1
      IF(PP1.GT.0.1 ) VV(2)=VV1(2)+ 0.1*PP1
      II = 2
      J1 = 28
      GO TO 144
51    ZZ1 = VV(2)-VV1(2)
      KK4 = 10
      VV(2) = VV1(2)
      VV1(3) = VV(3)
      PP1 = SQRT(ABS(P(30,30)))
      VV(3) = VV(3) + 0.1
      IF(PP1.GT.0.1 ) VV(3)=VV1(3)+ 0.1*PP1
```

```
      II = 3
      J1 = 29
      GO TO 144
52    ZZ1 = VV(3)-VV1(3)
      KK4 = 11
      VV1(4) = VV(4)
      VV(3) = VV1(3)
      PP1 = SQRT(ABS(P(31,31)))
      VV(4) = VV(4) + 0.1
      IF(PP1.GT.0.1 ) VV(4)=VV1(4)+ 0.1*PP1
      II = 4
      J1 = 30
      GO TO 144
53    ZZ1 = VV(4)-VV1(4)
      KK4 = 12
      PB1 = PB
      VV(4) = VV1(4)
      PP1 = SQRT(ABS(P(22,22)))
      PB = PB + 0.01
      IF(PP1.GT.0.01)  PB   =   PB1 + 0.1*PP1
      KK6 = 4
      WF1 = WF
      WF = PB*RTD
      J1 = 31
      GO TO 144
54    ZZ1 = PB-PB1
      A2(22,22) = (DPB1-DPB)/ZZ1
      A2(23,22) = (DQB1-DQB)/ZZ1
      A2(24,22) = (DRB1-DRB)/ZZ1
      IF(LAUNCH.GT.2) GO TO 92
      A2(17,22) = (DVE1-DVE)/ZZ1
      A2(18,22) = (DWE1-DWE)/ZZ1
92    KK4 = 13
      QB1 = QB
      PB = PB1
      WF = WF1
      WQ1 = WQ
      PP1 = SQRT(ABS(P(23,23)))
      QB = QB + 0.01
      IF(PP1.GT.0.01)  QB   =   QB1 + 0.1*PP1
      WQ = QB*RTD
      GO TO 355
55    ZZ1 = QB - QB1
      A2(23,23) = (DQB1-DQB)/ZZ1
      A2(24,23) = (DRB1-DRB)/ZZ1
      IF(LAUNCH.GT.2) GO TO 93
      A2(17,23) = (DVE1-DVE)/ZZ1
      A2(18,23) = (DWE1-DWE)/ZZ1
93    KK4 = 14
      RB1 = RB
      QB = QB1
      WQ = WQ1
      WR1 = WR
      PP1 = SQRT(ABS(P(24,24)))
```

```
      RB = RB + 0.01
      IF(PP1.GT.0.01)  RB  =  RB1 + 0.1*PP1
      WR = RB*RTD
      GO TO 355
56    ZZ1 = RB - RB1
      A2(23,24) = (DQB1-DQB)/ZZ1
      A2(24,24) = (DRB1-DRB)/ZZ1
      IF(LAUNCH.GT.2) GO TO 94
      A2(17,24) = (DVE1-DVE)/ZZ1
      A2(18,24) = (DWE1-DWE)/ZZ1
94    RB = RB1
      WR = WR1
      KK1 = 1
      KK3=0
      KK5 = 0
      KK6 = 5
      GO TO 355
22    DUE1 = BCSECS(1,1)*BACS(1)+BCSECS(1,2)*BACS(2)+BCSECS(1,3)*BACS(3)
      DVE1 = BCSECS(2,1)*BACS(1)+BCSECS(2,2)*BACS(2)+BCSECS(2,3)*BACS(3)
      DWE1 = BCSECS(3,1)*BACS(1)+BCSECS(3,2)*BACS(2)+BCSECS(3,3)*BACS(3)
     1+G
      GO TO (10,40),J
10    XMXTH = FTHZ*YTCG-FTHY*ZTCG
      XMYTH = ZTCG*FTHX+XTCG*FTHZ
      XMZTH = -YTCG*FTHX-XTCG*FTHY
40    XMX = XMXA+XMXTH
      XMY = XMYA+FZA*RDCG+XMYTH
      XMZ = XMZA-FYA*RDCG+XMZTH
      TMP1 = (1.-XIX/XIY)*PB
      DPB1 = XMX/XIX
      DQB1 = XMY/XIY+TMP1*RB
      DRB1 = XMZ/XIY-TMP1*QB
      GO TO(90,90,91),LAUNCH
90    CALL MDERIV
91    GO TO (64,44,45,46,47,48,49,50,51,52,53,54,55,56),KK4
57    IF(LAUNCH.GT.2) GO TO 95
      GO TO 96
95    IF(KOK.EQ.1) GO TO 96
      KOK = 1
      DO 97 I=17,18
      DO 97 I1=22,24
97    A2(I,I1) = 0.0
96    RETURN
      END
```

```
      SUBROUTINE MDERIV
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCC1/DUE1,DVE1,DWE1,DPB1,DQB1,DRB1
      DOUBLE PRECISION T,DT
      DIMENSION BACC(3)
      EQUIVALENCE (DVB,BACC(2)),(DWB,BACC(3))
      GO TO(30,50),LAUNCH
   30 RLCG1 = RLCG
      RLCG1 = RLCGO + RDCG
      CALL TRANS(ECSBCS,DUE1,BACC)
      TMP1 = RLCG1/XIY
      TMP2 = XM*RLCG1
      TMP3 = TMP1*TMP2 + 1.0
      FLY = (DRB1*TMP2-DVB*XM)/TMP3
      FLZ = -(DQB1*TMP2 + DWB*XM)/TMP3
      DVB = DVB + FLY/XM
      DWB = DWB + FLZ/XM
      DPB1 = 0.0
      DQB1 = DQB1 + FLZ*TMP1
      DRB1 = DRB1-FLY*TMP1
      CALL TRANS(BCSECS,BACC,DUE1)
      RETURN
   50 CALL TRANS(ECSBCS,DUE1,BACC)
      DVB = 0.0
      DWB = 0.0
      DPB1 = 0.0
      DQB1 = 0.0
      DRB1 = 0.0
      CALL TRANS(BCSECS,BACC,DUE1)
      RETURN
      END

      SUBROUTINE DETARA (POS,AAA)
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS
      II1 = 0
      DO 23 I=1,30
      AA1 = (I-1)/10.0
      AA2 = 0.1 + AA1
      II1 = II1 + 1
      IF(POS.GT.AA1.AND.POS.LE.AA2)GO TO 24
   23 CONTINUE
      II1 = 31
      CORRCT = 0.0
      GO TO 25
   24 CORRCT = 10.0*(POS-AA1)*(AREA(II1)-AREA(II1+1))
   25 AAA = 0.5-AREA(II1) + CORRCT
      RETURN
      END
```

```
      SUBROUTINE AUTOPT
      COMMON / AUTOP/NA,ZP1,ZP2,ZP3,ZY1,ZY2,ZY3,ZR1,ZR2,BPHIS,ZPI1,ZPI2,
     1EODCR,ZYI1,ZYI2,EVNCR,ZPD1,ZPD2,ZPD3,ZYD1,ZYD2,ZYD3,ZRD1,ZRD2,
     2BPHISD,ZPID1,ZPID2,EODCRD,ZYID1,ZYID2,EVNCRD,EZSS,EYSS,WQC,WRC,
     3EZRR,EYRR,BDELPC
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /SEEKR/ NS,VS(2),DVS(2),OSV(8)
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /BLIK1/BPHISM
      EQUIVALENCE  (EZ,OSV(1)),   (EY,OSV(2))
C *** LIMITATION OF INTEGRATORS*
      EODCR = XLIMIT(EODCR,PYLIM)
      EVNCR = XLIMIT(EVNCR,PYLIM)
C *** GUIDANCE FILTER - PITCH
      ZPD1 = GYZ*EZ-TAUZ*((3.*(ZP1+TAUZ*ZP2))+TAUZ*TAUZ*ZP3)
      ZPD2 = ZP1
      ZPD3 = ZP2
      EZSS = TAUL*ZP3+ZP2
C *** GUIDANCE FILTER - YAW
      ZYD1 = GYZ*EY-TAUY*((3.*(ZY1+TAUY*ZY2))+TAUY*TAUY*ZY3)
      ZYD2 = ZY1
      ZYD3 = ZY2
      EYSS = TAUL*ZY3+ZY2
      WQC = EZSS+QBIAS+GBIAS
      WRC = EYSS + RBIAS
      WQDIF = WQ -WQC
      WRDIF = WR -WRC
      EZRR = WQDIF-WRDIF
      EYRR = WQDIF+WRDIF
C *** ROLL COMPENSATION
      ZRD1 = WP1*(WP1*(BPH-ZR2)-2.*DP1*ZR1)
      ZRD2 = ZR1
      BPHISM = RK1*(ZR2+((RA1+RB2)*ZR1+ZRD1)/RA1/RB2)
      BPHISD = XLIMIT(BPHISM,RLIM)
      BDELPC =0.1*(BPHIS + 10.0*BPHISD)
C *** PITCH INTEGRATOR
      ZPID1 = TMP7*EZRR - TMP2*ZPI1 - TMP1*ZPI2
      ZPID2 = ZPI1
      EODCRD = TMP3*ZPI2+TMP4*ZPI1+ZPID1
C *** YAW INTEGRATOR*
      ZYID1 = TMP7*EYRR - TMP2*ZYI1 - TMP1*ZYI2
      ZYID2 = ZYI1
      EVNCRD = TMP3*ZYI2+TMP4*ZYI1+ZYID1
      RETURN
      ENTRY INAUPT
      TMP1 = WQ1*WQ1
      TMP2 = 2.*DQ1*WQ1
      TMP3 = PYAK1*PYBK1
      TMP4 = PYAK1+PYBK1
      TMP5 = WQG*WQG
      TMP6 = 2.*DQG*WQG
      TMP7 = PYIK1*WQ1*WQ1/TMP3
```

```
      RETURN
      END

      SUBROUTINE TARGET
C
C *** THIS ROUTINE CALCULATES TARGET/MISSILE RELATIVE POSITION AND
C *** SPEED AND GENERATES LINE-OF-SIGHT SIGNALS IN SEEKER PLATFORM
C *** COORDINATES
C
      COMMON / SEEKR / NS,VS(2),DVS(2),DSV(8)
      COMMON /STATEV/NT,UE(3), X(3),DUE(3),DX(3)
      COMMON / INCEPT/ UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /UTILTY/G,RTD
      DIMENSION RMP(3),SMP(3),TMP(3)
      EQUIVALENCE (RXBA,RMP(1)),(RYBA,RMP(2)),(RZBA,RMP(3))
      EQUIVALENCE (RXG,SMP(1)),(RYG,SMP(2)),(RZG,SMP(3))
      A = 0.0
      B = 0.0
      C = 0.0
      DO 10 I=1,3
      SMP(I) = UT(I)-UE(I)
      TMP(I) = XT(I)-X(I)
      RMP(I) = TMP(I)-SMP(I)
      A = A+TMP(I)*TMP(I)
   10 B = B+SMP(I)*SMP(I)
      TMRNGE = SQRT(A)
      TMVEL = SQRT(B)
      COSA =0.
      DO 20 I=1,3
      A = TMP(I)/TMRNGE
      B =SMP(I)/TMVEL
   20 COSA = COSA+A*B
      TMVEL = COSA*TMVEL
      A = VS(1)/RTD
      CSTHG = COS(A)
      SNTHG = SIN(A)
      A = VS(2)/RTD
      CSPSG = COS(A)
      SNPSG = SIN(A)
      A = TMP(1)*CSTHG-TMP(3)*SNTHG
      RXG = A*CSPSG+TMP(2)*SNPSG
      RYG = TMP(2)*CSPSG - A*SNPSG
      RZG = TMP(3)*CSTHG + TMP(1)*SNTHG
      BEPSZ = ATAN(-RZG/RXG)
      BEPSY = ATAN(RYG/RXG)
      RETURN
      ENTRY INTGT
      VS(1) = ATAN((X(3)-XT(3))/XT(1))*RTD
      VS(2) = 0.
      RETURN
      END
```

```
      SUBROUTINE TRANSM
C ***
C     THIS ROUTINE CALCULATES DERIVATIVES FOR THE TRANSLATIONAL
C     EQUATIONS OF MISSILE MOTION, INCLUDING LAUNCHER DYNAMICS WHEN
C     APPROPRIATE.
C ***
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WF,WQ,WR,BTHETA,BPH,BPS
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCK6/ BACS(3)
      COMMON /COEFS/THR,AERC(18)
      COMMON /UTILTY/G,RTD
      COMMON / BLOCK7/KK3,THRP,TIMP
      COMMON /BLOCK8/KK1,KK5,VP
      DIMENSION ANGLS(6)
      EQUIVALENCE (ANGLS(1),PB)
C
C *** CALCULATE EULER TRIGONOMETRICAL TERMS
C
      IF(KK1.EQ.0)GO TO 20
      SNTHA = SIN(THETA)
      CSTHA = COS(THETA)
      SNPHI = SIN(PHI)
      CSPHI = COS(PHI)
      SNPSI = SIN(PSI)
      CSPSI = COS(PSI)
C
C *** CALCULATE BODY/EARTH AND EARTH/BODY TRANSFORMATION MATRICES
C
      TMP1 = SNPHI*SNTHA
      TMP2 = CSPHI*SNTHA
      BCSECS(1,1) = CSPSI*CSTHA
      BCSECS(2,1) = SNPSI*CSTHA
      BCSECS(3,1) =-SNTHA
      BCSECS(1,2)= CSPSI*TMP1-SNPSI*CSPHI
      BCSECS(2,2)= SNPSI*TMP1+CSPSI*CSPHI
      BCSECS(3,2)= CSTHA*SNPHI
      BCSECS(1,3)= CSPSI*TMP2+SNPSI*SNPHI
      BCSECS(2,3)= SNPSI*TMP2-CSPSI*SNPHI
      BCSECS(3,3)= CSTHA*CSPHI
      DO 15 I=1,3
      DO 15 K=1,3
15    ECSBCS(I,K)= BCSECS(K,I)
C
C *** CALCULATE AERODYNAMIC FORCES AND MOMENTS
C
20    CALL AERODY
C
C *** CALCULATE THRUST COMPONENTS
```

```
C
      FTHX = THR*COSAT
      FTHY = THR*SATPHI
      FTHZ = THR*SATCPH
C
C *** CALCULATE BODY ACCELERATIONS EXCLUDING GRAVITY
C
      BACS(1) = (FTHX-FXA)/XM
      BACS(2) = (FTHY+FYA)/XM
      BACS(3) = (FTHZ+FZA)/XM
      IF(KK3.NE.0)RETURN
C
C *** TRANSFORM BODY ACCELERATIONS TO ECS AND CALCULATE DERIVATIVES
C
      CALL TRANS(BCSECS,BACS,DUE)
      DWE = DWE+G
      DX = UE
      DY = VE
      DZ = WE
      RETURN
      ENTRY INTRAN
C
C *** CALCULATE THRUST ANGLES AS SINES AND COSINES
C
      TMP1 = SQRT(XTCG*XTCG+YTCG*YTCG+ZTCG*ZTCG)
      COSAT = XTCG/TMP1
      SATPHI = YTCG/TMP1
      SATCPH = ZTCG/TMP1
C
C *** CONVERT INITIAL VALUES TO RADIANS
C
      DO 10 I=1,6
10    ANGLS(I) = ANGLS(I)/RTD
      RETURN
      END
```

```
      SUBROUTINE ROTATM
C ***
C     THIS ROUINE CALCULATES DERIVATIVES FOR THE MISSILE ROTATIONAL
C     VARIABLES PB,QB,RB AND THE EULER ANGLES THETA, PHI, PSI.
C ***
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /TIMES/T,DT,TBC,TSTOP,IPR,J,LAUNCH
      DOUBLE PRECISION T,DT
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /UTILTY/G,RTD
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      DIMENSION BACC(3)
      EQUIVALENCE (DVB,BACC(2)),(DWB,BACC(3))
C
C *** MOMENTS DUE TO THRUST MISALIGNMENT
C
      GC TO (10,40),J
 10   XMXTH = FTHZ*YTCG-FTHY*ZTCG
      XMYTH = ZTCG*FTHX+XTCG*FTHZ
      XMZTH = -YTCG*FTHX-XTCG*FTHY
C
C *** TOTAL APPLIED MOMENTS
C
 40   XMX = XMXA+XMXTH
      XMY = XMYA+FZA*RDCG+XMYTH
      XMZ = XMZA-FYA*RDCG+XMZTH
C
C *** DERIVATIVES
C
      TMP1 = (1.-XIX/XIY)*PB
      DPB = XMX/XIX
      DQB = XMY/XIY+TMP1*RB
      DRB = XMZ/XIY-TMP1*QB
      DTHA = QB*CSPHI-RB*SNPHI
      DPSI = (RB*CSPHI+QB*SNPHI)/CSTHA
      DPHI = PB+DPSI*SNTHA
      WP = PB*RTD
      WQ = QB*RTD
      WR = RB*RTD
      BPH = PHI*RTD
C
C *** MODIFY DERIVATIVES WHEN LAUNCHER DYNAMICS ARE IN EFFECT
C
      GO TO (50,30,20),LAUNCH
 20   RETURN
 30   RLCG = RLCGO+RDCG
      CALL TRANS(ECSBCS,DUE,BACC)
      TMP1= RLCG/XIY
      TMP2 = XM*RLCG
```

```
      TMP3 = TMP1*TMP2+1.
      FLY = (DRB*TMP2-DVB*XM)/TMP3
      FLZ = -(DQB*TMP2+DWB*XM)/TMP3
      DVB= DVB+FLY/XM
      DWB = DWB+FLZ/XM
      DPB =0.
      DQB = DQB+FLZ*TMP1
      DRB = DRB-FLY*TMP1
      CALL TRANS(BCSECS,BACC,DUE)
      RETURN
50    CALL TRANS(ECSBCS,DUE,BACC)
      DVB = 0.
      DWB =0.
      DPB =0.
      DQB =0.
      DRB = 0.
      CALL TRANS(BCSECS,BACC,DUE)
      RETURN
      ENTRY ROTZER
      XMXTH =0.
      XMYTH = 0.
      XMZTH  =0.
      RETURN
      END

      SUBROUTINE SEEKER
      COMMON / SEEKR/NS,BTHTG,BPSIG,BTHD,BPSD,EZ,EY,OSVV(6)
      COMMON / SEEKK/SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON / INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /UTILTY/G,RTD
      COMMON /SNSE/ AREA(31),EZNOIS,EYNOIS
      DOUBLE PRECISION T,DT
      ENTRY INSEEK
      I = IDINT(T*1.D3+.5D0)
      I = MOD(I,50)
      IF(I.NE.0) RETURN
      TMP1 = TMRNGE/32810.
      TMP1 = .75*TMP1*TMP1
      EZ = DEAD(-TMP1,TMP1, BEPSZ)*SKSP
      CALL SNOISE(TMP1,BEPSZ,EZ,EZNOIS)
      EY = DEAD(-TMP1,TMP1,BEPSY)*SKSY
      CALL SNOISE(TMP1,BEPSY,EY,EYNOIS)
      BTHTG = BTHTG + DTSAMP*EZ
      BPSIG = BPSIG + DTSAMP*EY
      RETURN
      END
```

```
      SUBROUTINE VANEMD
C ***
C     THIS ROUTINE EVALUATES DERIVATIVES FOR INTEGRATION VARIABLES
C     USED IN THE VANES MODULE.
C ***
      COMMON / AUTOP/NA,ZP1,ZP2,ZP3,ZY1,ZY2,ZY3,ZR1,ZR2,BPHIS,ZPI1,ZPI2,
     1EODCR,ZYI1,ZYI2,EVNCR,ZPD1,ZPD2,ZPD3,ZYD1,ZYD2,ZYD3,ZRD1,ZRD2,
     2BPHISD,ZPID1,ZPID2,EODCRD,ZYID1,ZYID2,EVNCRD,EZSS,EYSS,WQC,WRC,
     3EZRR,EYRR,BDELPC
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /VANEK/VGAIN,VLIM,VRLIM
      COMMON /BLOCK4/ VV5(4),DLTC(4)
      COMMON /BLIK2/ AVD(4),BVD(4)
      DLTC(1) = EODCR+BDELPC
      DLTC(2) = EVNCR+BDELPC
      DLTC(3) = EODCR-BDELPC
      DLTC(4) = EVNCR-BDELPC
      DO 30 I=1,4
      VV5(I) = VV(I)
      IND = 1
      IF(ABS(VV(I)).LE.VLIM)GO TO 10
      IND = 2
      VV(I)= XLIMIT(VV(I),VLIM)
10    DVV(I) = XLIMIT(VGAIN*(DLTC(I)-VV(I)),VRLIM)
      GO TO(30,20),IND
      AVD(I) = DVV(I)
      BVD(I) = DVV(I)*VV(I)
20    IF(DVV(I)*VV(I).GT.0.)DVV(I)=0.
30    CONTINUE
      TMP1 = VV(1)+VV(2)
      TMP2 =VV(3)+VV(4)
      DEL(1) = 0.25*(TMP1+TMP2)
      DEL(3) = 0.25*(TMP2-TMP1)
      DEL(2) = 0.25*(VV(2)+VV(4)-VV(1)-VV(3))
      RETURN
      END
```

```
      SUBROUTINE AERODY
C ***
C     THIS ROUTINE EVALUATES AERODYNAMIC FORCES AND MOMENTS APPLIED TO
C     THE MISSILE, USING COEFFICENTS AND DERIVATIVES OBTAINED BY TABLE
C     INTERPOLATION. FORCES AND MOMENTS ARE RETURNED IN COMMON BLOCK
C     /FCEMOM/.
C ***
      COMMON /COEFS/THR,CMQ,CNR,CNP,CY2,CL3,CXO,CMO,CDCM,CNF,CN2,
     1CLP,CL2,CXC,CNQ,CMDQP,CLDRP,CMR,CLD
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VA,RHO
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /TIMES/T,DT,TBC,TSTOP,IPR,J,LAUNCH
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI,
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /VANES/NV,VVD(8),DELQ,DELR,DELP
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCK8/KK1,KK5,VP
      DOUBLE PRECISION T, DT
      DIMENSION BVEL(3),DUM(3)
      EQUIVALENCE  (UB,BVEL(1)),(VB,BVEL(2)),(WB,BVEL(3))
      IF(KK5.EQ.1)GO TO 30
      DUM(1) = UE-WUE
      DUM(2) = VE-WVE
      DUM(3) = WE-WWE
      CALL TRANS(ECSBCS,DUM,BVEL)
      RHO = 2.3738E-3+6.7844E-8*Z
      VA = 1116.08+3.6292E-3*Z
      TMP1 = VB*VB+WB*WB
      VP = UB*UB+TMP1
      TMP1 = SQRT(TMP1)
      QUE = 0.5*RHO*VP
      VP = SQRT(VP)
      XMN=VP/VA
      ALFA = ATAN(WB/UB)
      BETA = ATAN(VB/UB)
      ALFAP = ATAN(TMP1/UB)
      IF (TMP1.EQ.0.)GO TO 40
      CSPHIP = WB/TMP1
      SNPHIP = VB/TMP1
      GO TO 50
40    CSPHIP = 1.
      SNPHIP = 0.
50    CONTINUE
      GO TO (10,20),J
10    CALL DTLUX1
      GO TO 30
20    CALL DTLUX2
30    SN2PHI = 2.*SNPHIP*CSPHIP
      SN4PHI = 2.*SN2PHI*(CSPHIP-SNPHIP)*(CSPHIP+SNPHIP)
      SN2PHI = SN2PHI*SN2PHI
      TMP1 = DELR*CMR
      TMP2 = DELQ*CMDQP
```

```
      TMP3 = TMP1*CSPHIP+TMP2*SNPHIP
      TMP4 = TMP2*CSPHIP-TMP1*SNPHIP
      TMP1 = CNP*SN4PHI+TMP3
      TMP2 = CMO+CDCM*SN2PHI+TMP4
      CM = CSPHIP*TMP2+SNPHIP*TMP1
      CN = CSPHIP*TMP1-SNPHIP*TMP2
      CL = CL2*SN4PHI+CL3*SNPHIP+DELP*CLD
      CX=CXO+CXC
      TMP1 = DELR*CLDRP
      TMP2 = DELQ*CNQ
      TMP3 = TMP1*CSPHIP+TMP2*SNPHIP
      TMP4 = TMP2*CSPHIP-TMP1*SNPHIP
      TMP1 = CY2*SN4PHI+TMP3
      TMP2 = CNF+CN2*SN2PHI+TMP4
      CY = CSPHIP*TMP1-SNPHIP*TMP2
      CZ = -CSPHIP*TMP2-SNPHIP*TMP1
      TMP1 = QUE*S
      FXA = TMP1*CX
      FYA = TMP1*CY
      FZA = TMP1*CZ
      TMP1 = TMP1*D
      TMP2 = 0.5*D/VP
      XMXA = TMP1*(CL+WP*TMP2*CLP)
      XMYA= TMP1*(CM+WQ*TMP2*CMQ)
      XMZA = TMP1*(CN+WR*TMP2*CNR)
      RETURN
      END

      FUNCTION DEAD(P1,P2,X)
C
C     DEAD SPACE
C
      DEAD =0.0
      IF(X.GT.P1.AND.X.LT.P2)RETURN
      DEAD = SIGN(1.0,X)
      RETURN
      END
```

```
      SUBROUTINE SYSRUN
C ***
C     THIS ROUTINE CONTROLS THE CALCULATION OF THE MISSILE TRAJECTORY
C     AND TARGET-MISSILE INTERCEPT POINT. THE PRINT ROUTINE IS CALLED
C     AS REQUIRED TO PRINT RESULTS.
C ***
      COMMON / INCEPT/ UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON /STATEV/NT,UB,VB,WB,X(3),DUE(6)
      COMMON /COEFS/THR,AERC(18)
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON / SEEKR/ NS,BTHTG,BPSIG,OSV(10)
      COMMON / VANES/ NV,VV(4),DVV(4),DELQ,DELR,DELP
      COMMON /TRANSF/BCSECS(3,3),ECSBCS(3,3),BCSGCS(3,3),ECSGCS(3,3)
      COMMON /BLOCK1/P(31,31),DP(31,31)
      COMMON /BLOCK2/ A2(31,31),KIK,KOUNT,KICK,KAT,B2(2),K400
      COMMON /BLOCK9/C2(84,31),KOK
      DOUBLE PRECISION T,DT,SVDT
      DIMENSION XMOLD(3),TOLD(3),XST(3)
C
C *** PRINT DATA HEADING AND INITIALIZE LAUNCHER DYNAMICS INDEX
C
      CALL PRHEAD
      LAUNCH = 1
C
C *** INITIALIZE AERODYNAMICS ROUTINE, DERIVATIVES AND TARGET POSITION.
C
      DELQ = 0.0
      DELR = 0.0
      DELP =0.0
      THR = 0.0
      T = 0.D0
      BEPSZ =0.
      BEPSY = 0.
      CALL THRCON
      CALL TRANSM
      CALL ROTATM
      CALL INTGT
      BEPSZ = 0.
      BEPSY = 0.
      CALL INSEEK
      CALL AUTOPT
      CALL VANEMD
      J = 1
      K=1
      DO 5 I=1,3
    5 XST(I) = X(I)
      SVDT = DT
      N = IDINT(DT/.5D-3)
      IPR = N*IPR
      DT = .5D-3
      CALL INSYST
      CALL INRK4
C
```

```
C *** INTEGRATE MISSILE EQUATIONS AND CALCULATE TARGET-MISSILE POSITION.
C
10    KSTEP =0
      CALL PRDATA
20    DO 25 I=1,3
      XMOLD(I)= X(I)
25    TOLD(I) = XT(I)
      CALL SYSINT
      CALL TARGET
      CALL INSEEK
      GO TO (70,90),J
70    IF(THR)80,80,90
80    J=2
      CALL ROTZER
90    GO TO (75,85,95),LAUNCH
75    DO 76 I=1,3
76    XMOLD(I) = X(I)-XST(I)
      CALL TRANS(ECSBCS,XMOLD,TOLD)
      IF (TOLD(1).LT.RL1)GO TO 45
      LAUNCH = 2
      WRITE(  6,910)T
      GO TO 45
85    DO 86 I=1,3
86    XMOLD(I) = X(I)-XST(I)
      CALL TRANS(ECSBCS,XMOLD,TOLD)
      IF(TOLD(1).LT.RL2)GO TO 45
      LAUNCH = 3
      WRITE(  6,920)T
      IPR = IPR/N
      N = IDINT(T/SVDT)+1
      DT = DFLOAT(N)*SVDT-T
      CALL INSYST
      CALL INRK4
      CALL SYSINT
      DT = SVDT
      KSTEP = MOD(N,IPR)-1
      CALL INSYST
      CALL INRK4
95    GO TO (30,40),K
C
C *** IF MISSILE WITHIN 5 FT. OF TARGET DIVIDE STEP LENGTH BY 2(FIRST TIME
30    IF(TMRNGE.GT.5.)GO TO 40
      DT = .5D+0*DT
      IPR = IPR+IPR
      K = 2
C
C *** IF MISSILE-TARGET RELATIVE VELOCITY IS POSITIVE INTERCEPT HAS
C *** OCCURRED
C
40    IF(TMVEL.GE.0.0) GO TO 50
      IF(T.GT.TSTOP) RETURN
45    KSTEP = KSTEP+1
      IF(T.GT.2.0)RETURN
      IF (KSTEP-IPR)20,10,10
```

```
C
C *** CALCULATE MISS DISTANCE FROM CURRENT AND PREVIOUS POSITIONS
C
50      A = 0.
        B = 0.
        C = 0.
        DO 60 I=1,3
        TMP1 = XMOLD(I)-TOLD(I)
        A = A+TMP1*TMP1
        TMP2 = X(I)-XT(I)
        B = B+TMP2*TMP2
        TMP1 = X(I)-XMOLD(I)
60      C = C+TMP1*TMP1
        A=SQRT(A)
        B=SQRT(B)
        C = SQRT(C)
        Z = .5*(A+B+C)
        A = 2.*SQRT(Z*(Z-A)*(Z-B)*(Z-C))/C
        WRITE (  6,900) A
        WRITE(6,124)T
        DO 288 I=1,31
288     WRITE(6,11)I,(P(I,K),K=1,I)
11      FORMAT(//1X,'P(',I2,', J) =',7E15.5/ (11X,7E15.5/))
124     FORMAT(1X,'TIME = ',F6.4)
 900    FORMAT (//20X,'*****  MISS DISTANCE    *****',F10.2, ' FT.')
 910    FORMAT (10X,'FIRST LUG OFF LAUNCHER AT T = ', F8.4)
 920    FORMAT (10X,'SECOND LUG OFF LAUNCHER AT T = ',F8.4)
        END

        SUBROUTINE TRANS(TMTX,VECTOR,RESULT)
        DIMENSION TMTX(3,3),VECTOR(3),RESULT(3)
        RESULT(1) = TMTX(1,1)*VECTOR(1)+TMTX(1,2)*VECTOR(2)+TMTX(1,3)*
       1VECTOR(3)
        RESULT(2) = TMTX(2,1)*VECTOR(1)+TMTX(2,2)*VECTOR(2)+TMTX(2,3)*
       1VECTOR(3)
        RESULT(3) = TMTX(3,1)*VECTOR(1)+TMTX(3,2)*VECTOR(2)+TMTX(3,3)*
       1VECTOR(3)
        RETURN
        END
```

```
      SUBROUTINE THRCON
C ***
C     THIS ROUTINE CALCULATES MISSILE MASS, INERTIAS AND CG POSITION
C     AS A FUNCTION OF ENGINE THRUST CONDITIONS. THE INTEGRAL OF THE
C     THRUST IS CALCULATED BY THE TRAPEZOIDAL RULE TO OBTAIN ENGINE
C     IMPULSE.
C ***
      COMMON /COEFS/THR,AERC(18)
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /UTILTY/G,RTD
      COMMON / BLOCK7/KK3,THRP,TIMP
      DOUBLE PRECISION T,DT
      TIMP = TIMP+.5*(T-TPR)*(THR+THRP)
      THRP = THR
      TPR = T
      DELW = TIMP/SI
      XM = (WO-DELW)/G
      TMP1 = 1.-DELW/WO
      XIX = XIXO*TMP1
      XIY = XIYO*TMP1
      RDCG = RDCGO-DELW*CGSHWP
      RETURN
      ENTRYINTHRC
C
C *** ZERO STARTING VALUES OF THRUST INTEGRAL AND TIME
C
      TIMP = 0.
      TPR =0.
      THRP = 0.
      CGSHWP = (RDCGO-RDCGP)/WP
      RETURN
      END
```

```
      SUBROUTINE RK4(N,V,Q,K)
C ***
C     THIS ROUTINE INCREMENTS VARIABLES, GIVEN THEIR DERIVATIVES ACCORDING
C     TO THE RUNGE-KUTTA 4 POINT SCHEME.
C ***
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J1,LAUNCH
      DOUBLE PRECISION T,DT
      DIMENSION V(N),Q(N)
      DO 50 I=1,N
      J=N+I
      GO TO(10,20,30,40),K
   10 Q(J) = V(J)
      Q(I) = V(I)
      V(I) = V(I)+DTOV2*V(J)
      GO TO 50
   20 V(I) = Q(I)+DTOV2*V(J)
      Q(J) = Q(J)+V(J)+V(J)
      GO TO 50
   30 V(I) = Q(I)+DT1*V(J)
      Q(J) = Q(J)+V(J)+V(J)
      GO TO 50
40    V(I) = Q(I)+DT1*(Q(J)+V(J))*0.1666667
   50 CONTINUE
      RETURN
      ENTRY INRK4
      DTOV2 = SNGL(DT*.5D+0)
      DT1 = SNGL(DT)
      RETURN
      END
```

```
      FUNCTION XLIMIT(V,VLIM)
      IF(ABS(V)-VLIM)40,40,10
   10 IF (V)20,30,30
   20 XLIMIT = -VLIM
      RETURN
   30 XLIMIT = VLIM
      RETURN
   40 XLIMIT = V
      RETURN
      END
```

```
      SUBROUTINE DTLUX1
C ***
C     THIS ROUTINE OBTAINS THRUST AND AERODYNAMIC COEFFICIENTS AND
C     DERIVATIVES FROM TABLE INTERPOLATION.  TABULATED FUNCTIONS ARE
C     HELD IN BLANK COMMON AND ROUTINE INTRP3 IS CALLED TO PERFORM THE
C     ACTUAL INTERPOLATION. RESULTS ARE RETURNED IN COMMON BLOCK /COEFS/
C ***
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VSS,RHO
      COMMON /TIMES/T,DT,TBO,TSTOP,IPR,J,LAUNCH
      COMMON /VANES/SKP1(9),DELQ,DELR,DELP
      COMMON /COEFS/THR,CMQ,CNR,CNP,CY2,CL3,CXO,CMO,CDCM,CNF,CN2,CLP,CL2
     1,CXC,CNQ,CMDQP,CLDRP,CMR,CLD
      COMMON /UTILTY/G,RTD
      DIMENSION ONEDM(4), TWODM(7)
      EQUIVALENCE (ONEDM(1),CNP),(TWODM(1),CMO)
      IF (T.GT..14)GO TO 10
      CALL INTRP3(T,0.,0.,1,THR)
      GO TO 20
   10 CALL INTRP3(T,0.,0.,2,THR)
      GO TO 20
      ENTRY DTLUX2
 20   ALF = ABS(ALFA)*RTD
      BET = ABS(BETA)*RTD
      ALFP = ALFAP*RTD
      DQ = ABS(DELQ)
      DR = ABS(DELR)
      CALL INTRP3(ALF,0.,0.,3,CMQ)
      CALL INTRP3(BET,0.,0.,3,CNR)
      DO 30 I=4,6
   30 CALL INTRP3(ALFP,0.,0.,I,ONEDM(I-3))
      CALL INTRP3(XMN,0.,0.,7,CXO)
      DO 40 I=8,14
   40 CALL INTRP3(ALFP,XMN,0.,I,TWODM(I-7))
      CALL INTRP3(ALFP,XMN,DQ,15,CNQ)
      CALL INTRP3(ALFP,XMN,DR,15,CLDRP)
      CALL INTRP3(ALFP,XMN,DQ,16,CMDQP)
      CALL INTRP3(ALFP,XMN,DR,16,CMR)
      CALL INTRP3(ALFP,XMN,ABS(DELP),17,CLD)
      RETURN
      END
```

```
      SUBROUTINE INTRP3(X,Y,Z,I,FXYZ)
C ***
C     THIS ROUTINE PERFORMS LINEAR INTERPOLATION IN TABULATED FUNCTIONS
C     OF 1, 2 OR 3 INDEPENDENT VARIABLES. THE FUNCTIONS MUST BE
C     TABULATED FOR VALUES OF INDEPENDENT VARIABLES WHICH START AT ZERO
C     AND INCREASE WITH CONSTANT INTERVALS. THE TABLES USED ARE DEFINED
C     FOR POSITIVE RANGES OF INDEPENDENT VARIABLES BUT IF REQUIRED
C     THE VARIABLE INCREMENT MAY BE NEGATIVE.
C ***
      COMMON DXDYDZ(60),IADD(20),AERO(1360)
      J = 3*I-2
      DX = DXDYDZ(J)
      DY = DXDYDZ(J+1)
      DZ = DXDYDZ(J+2)
      J = IFIX(X/DX)
      DELX = X/DX-FLOAT(J)
      IF (DY.EQ.0.)GO TO 40
      IF (J.GT.16) J=16
      K = IFIX(Y/DY)
      DELY = Y/DY-FLOAT(K)
      IF (K.GT.4)K=4
      IF (DZ.EQ.0.)GO TO 50
      L = IFIX(Z/DZ)
      DELZ = Z/DZ-FLOAT(L)
      IF (L.GT.4)L=4
      M = J+16*K+64*L+IADD(I)
      N = 1
      NN = 2
      GO TO 30
   10 M = M+64
      N = 2
      FXY1 = FXY
      GO TO 30
   20 FXYZ = FXY1+(FXY-FXY1)*DELZ
      RETURN
   40 M = J+IADD(I)
      NN = 1
      GO TO 30
   50 M = J+16*K+IADD(I)
      NN = 2
      N = 3
      GO TO 30
   60 FXYZ = FX1
      RETURN
   70 FXYZ = FXY
      RETURN
   30 FX1 = AERO(M)+(AERO(M+1)-AERO(M))*DELX
      GO TO(60,80),NN
   80 M = M+16
      FX2 = AERO(M)+(AERO(M+1)-AERO(M))*DELX
      M = M-16
      FXY = FX1+(FX2-FX1)*DELY
      GO TO(10,20,70),N
      END
```

```
      BLOCK DATA
      COMMON / SEEKR/ NS,VS(2),DVS(2),DSV(8)
      COMMON / SEEKK/ SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS
      COMMON /AUTOP/NA,VA(15),DVA(15),DV(7)
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /VANES/NV,VV(4),DVV(4), DEL(3)
      COMMON /VANEK /VGAIN,VLIM,VRLIM
      COMMON /VMG/ H,MS
      DATA H,MS/0.0025,31/
      DATA SKSP,SKSY,TSAMP,DTSAMP,CROSPT,CROSTP,SYGBIS,SZGBIS/3.,3.,0.,
     10.05,0.,0.,0.,0./
      DATA NS,VS/ 2,2*0.0/
      DATA WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,PYAK1,PYBK1,
     1PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS/373.,1.,15.,15.,2.,
     26750.,12.,60.,130.,.53,.33,40.,15.,2.8,115.,.64,15.,7.,1.,0.0,0.0/
      COMMON /MSINCG/SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WF,WQ,WR,BTHETA,BPH,BPS
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /UTILTY/G,RTD
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /INCEPT/UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      DATA G,RTD/32.17,57.2957795/
      DATA NT,NR/6,6/
      DATA PB,QB,RB,UE,VE,WE,THETA,PHI,PSI,X,Y,Z/0.,0.,0.,.1,0.,0.,5.,
     10.,0.,0.,0.,-40./
      DATA NA,VA/15,15*0./
      DATA NV,VGAIN,VLIM,VRLIM/4,15.,20.,200./
      DATA VV/4*0./
      DATA S,D,XTCG,YTCG,ZTCG/.267,.584,2.75,0.,0./
      DATA RL1,RL2,WUE,WVE,WWE/3.5,6.07,0.,0.,0./
      DATA SI,WO,WP,XIXO,XIYO,RLCGO,RDCGO,RDCGP/195.8,121.,19.4,.241,15.
     111,2.54,-.375,-.15/
      DATA UT/3*0./
      DATA XT/10000.,0.,0./
      END
```

```
      SUBROUTINE PRDATA
      COMMON /SEEKR/ NS,VS(2),DVS(2),OSV(8)
      COMMON /TIMES/T,DT,TBC,TSTOP,IPR,J,LAUNCH
      DOUBLE PRECISION T,DT
      COMMON /CNTRL/DUM(6),DATA(64)
      COMMON /AUTOP/NA,VA(15),DVA(15),OV(7)
      COMMON /VANES/NV,VV(4),DVV(4),DEL(3)
      COMMON /ROTATE/NR,PB,QB,RB,THETA,PHI,PSI,DPB,DQB,DRB,DTHA,DPHI
     1,DPSI,SNTHA,CSTHA,SNPHI,CSPHI,SNPSI,CSPSI,WP,WQ,WR,BTHETA,BPH,BPS
      COMMON /STATEV/NT,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      COMMON /ADDV/ALFAP,ALFA,BETA,XMN,CSPHIP,SNPHIP,QUE,VSS,RHO
      COMMON /COEFS/THR,AERC(18)
      COMMON /GEOMK/S,D,XTCG,YTCG,ZTCG,RL1,RL2,WUE,WVE,WWE
      COMMON /MSINCG/SI,WO,WF,XIXO,XIYO,RLCGO,RDCGO,RDCGP,XM,XIX,XIY,
     1RLCG,RDCG
      COMMON /FCEMOM/FXA,FYA,FZA,XMXA,XMYA,XMZA,FTHX,FTHY,FTHZ
      COMMON / INCEPT/ UT(3),XT(3),TMVEL,TMRNGE,BEPSZ,BEPSY
      COMMON / AUTOK/ WQG,DQG,TAUZ,TAUY,TAUL,GYZ,RA1,RB2,WP1,DP1,RK1,
     1PYAK1,PYBK1,PYIK1,WQ1,DQ1,PYLIM,RLIM,GBIAS,QBIAS,RBIAS
      COMMON /UTILTY/G,RTD
      DIMENSION RDRV(6), DDRV(6)
      EQUIVALENCE (RDRV(1),DPB)
      BTHETA = THETA*RTD
      BPS = PSI*RTD
      GO TO(40,50,60),ISW
   40 RETURN
   50 WRITE(  6,930) T,UE,VE,WE,X,Y,Z,WP,WQ,WR,BTHETA,BPH,BPS,UT,XT,
     1 TMRNGE,TMVEL,VS
      LINES = LINES+3
      IF(LINES .LT. 52) RETURN
      LINES = 1
      IPAGE = IPAGE+1
      WRITE (  6,940) IPAGE
      RETURN
   60 CONTINUE
      CONTINUE
      IPAGE = IPAGE+1
      WRITE (  6,940) IPAGE
      ALFAP = ALFAP*RTD
      ALFA = ALFA*RTD
      BETA = BETA*RTD
      CSPHIP = ATAN2(SNPHIP,CSPHIP)*RTD
      DO 70 I=1,6
   70 DDRV(I) = RDRV(I)*RTD
      WRITE(  6,950) T,UE,VE,WE,X,Y,Z,DUE,DVE,DWE,DX,DY,DZ
      WRITE(  6,960) WP,WQ,WR,BTHETA,BPH,BPS,DDRV
      WRITE(  6,970) VS,DVS
      WRITE(  6,980) VA,DVA
      WRITE(  6,990) VV,DVV
      WRITE(  6,1000) DEL,BEPSZ,BEPSY,OSV,OV
      WRITE(  6,1010) XMN,VSS,RHO,QUE,ALFAP,ALFA,BETA,CSPHIP,AERC,
     1 FXA,FYA,FZA,XMXA,XMYA,XMZA
      WRITE(  6,1020) FTHX,FTHY,FTHZ,XM,XIX,XIY,RDCG
      WRITE(  6,1030) UT,XT,TMRNGE,TMVEL
```

```
      RETURN
      ENTRY PRHEAD
      WRITE(  6,900) (DATA(I),I=1,20)
      WRITE(  6,920) S,D,RL1,RL2,WO,WF,XIXO,XIYO,RDCGO,RDCGP,QBIAS,
     1RBIAS,XTCG,YTCG,ZTCG,WUE,WVE,WWE,RLCGO,SI,DT
      LINES = 40
      IPAGE = 1
      IF (IPR)10,20,30
10    ISW = 3
      IPR = -IPR
      RETURN
20    ISW = 1
      RETURN
30    ISW = 2
      WRITE(  6,910)
      RETURN
900   FORMAT(1H1,120X,'PAGE 1',/48X,'TERMINAL HOMING SIMULATION (DIGITAL
     1', /48X,36('-'), //20X,20A4 //)
910   FORMAT(//25X,'RESULTS ROW 1:', /30X,'COLUMN 1 TIME IN SECONDS',
     125X,'COLUMN 2 UE IN FT/SEC', /30X,'COLUMN 3 VE IN FT/SEC',28X,
     2'COLUMN 4 WE IN FT/SEC', /30X,'COLUMN 5 MISSILE X COORD IN FT',
     319X,'COLUMN 6 MISSILE Y COORD IN FT', /30X,'COLUMN 7 MISSILE Z
     4COORD IN FT', 19X,'COLUMN 8 ROLL RATE IN DEG/SEC', /30X,'COLUMN 9
     5 PITCH RATE IN DEG/SEC', 19X,'COLUMN 10 YAW RATE IN DEG/SEC', /30
     6X,'COLUMN 11 THETA IN DEGREES', 24X,'COLUMN 12 PHI IN DEGREES', /
     730X,'COLUMN 13 PSI IN DEGREES', //25X,'RESULTS ROW 2:', /30X,
     8'COLUMN 2 TARGET U IN FT/SEC', 21X,'COLUMN 3 TARGET V IN FT/SEC'
     9, /30X,'COLUMN 4 TARGET W IN FT/SEC', 21X,'COLUMN 5 TARGET X COORD
     ARD IN FT',/30X,'COLUMN 6 TARGET Y COORD IN FT', 19X,'COLUMN  7 TA
     BRGET Z COORD IN FT', /30X,'COLUMN 8 MISSILE/TARGET RANGE IN FT',
     C13X,'COLUMN 9 MISSILE/TARGET CLOSING SPEED IN FT/SEC', /30X,'COLU
     DMN 10 GIMBAL ANGLE THETAG IN DEGREES', 9X,'COLUMN 11 GIMBAL ANGLE
     EPSIG IN DEGREES')
920   FORMAT (5X,'VEHICLE DETAILS:', //10X,'REFERENCE AREA', 15X,F8.3,
     1' SQ FT', 20X,'REFERENCE LENGTH', 12X,F8.3, ' FT', /10X,'FRONT LUG
     2 LAUNCHER TRAVEL',4X,F8.3,'FT',23X,'REAR LUG LAUNCHER TRAVEL', 4X,
     3F8.3, ' FT',/10X,'INITIAL TOTAL WEIGHT', 9X,F8.2, ' LBS', 22X,
     4'PROPELLANT WEIGHT', 10X,F8.2, ' LBS',/10X,'INITIAL X MOM. OF I.',
     5 9X,F8.3, ' SLUGS FT**2', 14X,'INITIAL Y MOM. OF I.',  8X,F8.3,
     6' SLUGS FT**2', /10X,'CG TOTAL SHIFT',15X,F8.3, ' FT', 23X,
     7'PROPELLANT CG TO CGO',  8X,F8.3, ' FT', /10X,'AUTOPILOT Q BIAS',
     813X,F8.3, ' DEG/SEC', 18X,'AUTOPILOT R BIAS', 12X,F8.3, ' DEG/SEC'
     9/10X,'THRUST POINT OFFSETS (X,Y,Z FT)',10X,3F10.2,/10X,'WIND SPEED
     A COMPONENTS (XE,YE,ZE F/S)', 5X,3F10.1, /10X,'REAR LUG TO CGO(FT)'
     B,22X,F10.3,/10X,'ENGINE SPECIFIC IMPULSE', 6X,F8.3, ' SECS', 21X,
     C    'INTEGRATION STEP LENGTH', 5X,F8.4, ' SECS')
930   FORMAT (/3X,F6.3, 2(3F10.2, 3F10.1),/9X,3F10.2, 4F10.1, 3F10.2)
940   FORMAT(1H1,30X,'TERMINAL HOMING CONTD ....', 61X,'PAGE', I3)
950   FORMAT (// 10X,'TIME',F8.3, ' SECONDS', //5X,'TRANSLATION VARIAB
     1LES IN F/SEC AND FT', 12X,3F10.2, 3F10.1, /5X,'TRANSLATION DERIVAT
     2IVES IN F/SEC**2 AND F/SEC',  5X,3F10.3, 3F10.2)
960   FORMAT (/5X,'ROTATION VARIABLES IN DEG/SEC AND DEGS', 11X,6F10.2,
     1/5X,'ROTATION DERIVATIVES IN DEG/SEC**2 AND DEG/SEC', 4X,6F10.3)
970   FORMAT (/5X,'SEEKER VARIABLES IN DEG AND DEG/SEC', 15X,2F10.3,/5X,
```

```
      1 'SEEKER DERIVATIVES IN DEG/SEC AND DEG/SEC**2',  8X,2F10.3)
980   FORMAT (/5X,'AUTOPILOT VARIABLES IN DEG ETC', 20X,6F10.3, /55X,
      1 6F10.3, /55X,7F10.3, /5X,'AUTOPILOT DERIVATIVES IN DEG ETC', 18X,
      26F10.3, /55X,6F10.3, /55X,7F10.3)
990   FORMAT (/5X,'VANE VARIABLES IN DEGREES', 25X,4F10.3, /5X,
      1 'VANE DERIVATIVES IN DEG/SEC', 23X,4F10.3)
1000  FORMAT (/5X,'DELQ,DELR, DELP(DEGREES)',11X,3F8.3,11X,'BEPSZ & BEP
      1SY(DEGS)', 2X,2F8.3,//5X,'SEEKER ADDITIONAL VARIABLES', 4X,8F10.3
      2,//5X,'AUTOPILOT ADDITIONAL VARIABLES', 10X,7F10.3)
1010  FORMAT (/5X,'MACH NO', F9.2, 4X,'SONIC SP', F8.1, 4X,'AIR DENS',
      2F8.6, 4X,'DYN ORES', F8.2, 4X,'ALFA P', F10.3, 4X,'ALFA', F12.3,
      2/5X 'BETA', F12.3, 4X,'PHI PR', F10.3,//5X,'AERODYNAMIC COEFFICIENT
      3TS', /5X,'CMO(A)', F10.4, 4X,'CNR(B)', F10.4, 4X,'CNP(A)', F10.4,
      44X,'CY2(A)', F10.4, 4X,'CL3(A)', F10.4, 4X,'CAO(M)', F10.4, 4X/5X,
      5'CMO(A,M)', F8.4, 4X,'CDCM(A,M)', F7.4, 4X,'CNF(A,M)', F8.4, 4X,
      6'CN2(A,M)', F8.4, 4X,'CLP(A,M)', F8.4, 4X,'CL2(A,M)', F8.4, /5X,
      7'CXC(A,M)', F8.4, 4X,'CNQ(A,M,Q)', F6.4, 4X,'CMDQP(3V)', F7.4, 4X,
      8'CLDRP(3V)', F7.4,4X,'CMR(A,M,R)', F6.4, 4X,'CLD(A,M,P)', F6.4,
      9// 5X,'AERODYNAMIC FORCES AND MOMENTS',/5X,'FXA(LB)', F9.2, 4X,
      A'FYA(LB)', F9.2, 4X,'FZA(LB)', F9.2, 4X,'MXA(LBFT)', F7.2, 4X,
      B'MYA(LBFT)', F7.2, 4X,'MZA(LBFT)', F7.2)
1020  FORMAT (/5X,'THRUST COMPONENTS (X,Y,Z LB)', 3F8.1, 4X,'MASS', F8.2
      1, 4X,'X M. OF I.', F8.2, 4X,'Y M. OF I.', F8.3,/5X,'CG SHIFT',20X,
      2 F8.3)
1030  FORMAT (/5X,'TARGET SPEED (X,Y,Z FT/SEC)', 3F8.1, 4X,'TARGET POSIT
      1ION (X,Y,Z FT)',3F10.1,/5X,'TARGET/MISSILE RANGE (FT)', F10.1,20X,
      2 'CLOSING SPEED (F/S)', 9X,F8.1)
      END
```

```
0.50      0.4602  0.4207  0.3821  0.3446  0.3085  0.2743  0.2420  0.2119  0.1841
0.1587    0.1357  0.1151  0.0968  0.0808  0.0668  0.0548  0.0446  0.0359  0.0287
0.0228    0.0179  0.0139  0.0107  0.00820 0.00621 0.00466 0.00347 0.00256 0.00187
  8          .02                          THRUST TABLE 1 FOR TIME 0 TO .14 SECS
 0.5        2850.      2660.      2240.     2230.      2205.      2180.      2170.
 48         .1                            THRUST TABLE 2 FOR TIME FROM .14 SECS
 0.5        2205.      2160.      2140.     2125.      2110.      2095.      2075.
 2060.      2040.      2020.      2005.     1990.      1970.      1950.      1910.
 1800.      1200.      610.       420.      320.       295.       220.       190.
 140.       120.       100.       90.       80.        75.        65.        55.
 48.        41.        35.        30.       20.        10.        0.

 16         2.                              TABLE OF RATE DAMPING DERIVATIVS CMQ
 -4.1       -5.25      -6.3       -7.4      -8.4       -9.3       -9.96      -10.45
 -10.78     -10.95     -11.0      -11.0     -11.0      -11.0      -11.0      -11.0
 16          2.                              DELTA CN PRIME
  0.         .05        .18        .4        .69       1.06       1.5        2.01
 2.59       3.22       3.86       4.73      4.73       4.73       4.73       4.73
 16         2.                              DELTA CY PRIME
 0.         -.015      -.07       -.17      -.3        -.47       -.65       -.87
 -1.1       -1.345     -1.6       -1.86     -1.86      -1.86      -1.86      -1.86
 16         2.                               DELTA CL PRIME LUGS
 0.         .015       .025       .032      .045       .051       .08        .11
 .145       .181       .215       .255      .255       .255       .255       .255
 16          .0916667                       CXO PRIME
 .465       .445       .43        .411      .397       .387       .379       .375
 .420       .558       .730       .970      1.2        1.2        1.2        1.2
 16  4      2.          .366667              CMO PRIME
 0.         -.95       -2.1       -3.6      -5.2       -7.2       -9.3       -11.55
 -13.8      -16.2      -18.55     -21.1     -21.1      -21.1      -21.1      -21.1
 0.         -.95       -2.1       -3.6      -5.2       -7.2       -9.3       -11.55
 -13.8      -16.2      -18.55     -21.1     -21.1      -21.1      -21.1      -21.1
 0.         -.95       -2.1       -3.6      -5.2       -7.2       -9.3       -11.55
 -13.8      -16.2      -18.55     -21.1     -21.1      -21.1      -21.1      -21.1
 0.         -.6        -1.6       -3.1      -4.75      -6.7       -8.8       -10.95
 -13.2      -15.5      -17.8      -20.2     -20.2      -20.2      -20.2      -20.2
 16  4      2.         .366667              DELTA CM PRIME
 0.         -.03       -.14       -.3       -.64       -1.19      -1.85      -2.63
 -3.46      -4.36      -5.38      -6.45     -6.45      -6.45      -6.45      -6.45
 0.         -.03       -.14       -.3       -.64       -1.19      -1.85      -2.63
 -3.46      -4.36      -5.38      -6.45     -6.45      -6.45      -6.45      -6.45
 0.         -.03       -.14       -.3       -.64       -1.19      -1.85      -2.63
 -3.46      -4.36      -5.38      -6.45     -6.45      -6.45      -6.45      -6.45
 0.         -.05       -.17       -.4       -.75       -1.32      -2.02      -2.8
 -3.68      -4.6       -5.65      -6.8      -6.8       -6.8       -6.8       -6.8
 16  4       2.        .366667              CN PRIME
 0.         .69        1.4        2.2       3.15       4.24       5.38       6.54
 7.72       9.04       10.55      12.2      12.2       12.2       12.2       12.2
 0.         .69        1.4        2.2       3.15       4.24       5.38       6.54
 7.72       9.04       10.55      12.2      12.2       12.2       12.2       12.2
 0.         .69        1.4        2.2       3.15       4.24       5.38       6.54
 7.72       9.04       10.55      12.2      12.2       12.2       12.2       12.2
 0.         .09        1.4        2.2       3.2        4.35       5.5        6.74
 8.0        9.37       10.7       12.0      12.        12.        12.        12.
```

55	16 4	2.	.366667		DELTA CN PRIME			
56	0.	.015	.06	.155	.31	.5	.75	1.05
57	1.395	1.78	2.2	2.63	2.63	2.63	2.63	2.63
58	0.	.015	.06	.155	.31	.5	.75	1.05
59	1.395	1.78	2.2	2.63	2.63	2.63	2.63	2.63
60	0.	.015	.06	.155	.31	.5	.75	1.05
61	1.395	1.78	2.2	2.63	2.63	2.63	2.63	2.63
62	0.	.015	.06	.155	.32	.53	.8	1.11
63	1.46	1.84	2.26	2.71	2.71	2.71	2.71	2.71
64	16 4	2.	.366667		ROLL DAMPING CLP			
65	-.232	-.315	-.39	-.464	-.527	-.579	-.62	-.649
66	-.668	-.675	-.67	-.645	-.645	-.645	-.645	-.645
67	-.232	-.315	-.39	-.464	-.527	-.579	-.62	-.649
68	-.668	-.675	-.67	-.645	-.645	-.645	-.645	-.645
69	-.232	-.315	-.39	-.464	-.527	-.579	-.62	-.649
70	-.668	-.675	-.67	-.645	-.645	-.645	-.645	-.645
71	-.25	-.333	-.41	-.482	-.55	-.609	-.657	-.698
72	-.728	-.75	-.72	-.72	-.72	-.72	-.72	-.72
73	16 4	2.	.366667		DELTA CLP PRIME			
74	0.	.007	.02	.045	.07	.101	.122	.193
75	.25	.297	.331	.354	.354	.354	.354	.354
76	0.	.007	.02	.045	.07	.101	.122	.193
77	.25	.297	.331	.354	.354	.354	.354	.354
78	0.	.007	.02	.045	.07	.101	.122	.193
79	.25	.297	.331	.354	.354	.354	.354	.354
80	0.	.008	.035	.07	.12	.186	.277	.387
81	.515	.672	.84	1.03	1.03	1.03	1.03	1.03
82	16 4	2.	.366667		CXC			
83	0.	0.	0.	0.	.002	.02	.055	.13
84	.24	.387	.642	1.09	1.09	1.09	1.09	1.09
85	0.	0.	0.	0.	.002	.02	.055	.13
86	.24	.387	.642	1.09	1.09	1.09	1.09	1.09
87	0.	0.	0.	0.	.002	.02	.055	.13
88	.24	.387	.642	1.09	1.09	1.09	1.09	1.09
89	0.	0.	0.	0.	.002	.008	.026	.07
90	.135	.23	.365	.56	.56	.56	.56	.56
91	16 4 4	2.	.366667	10.	CN PRIME PER DELTA R OR Q			
92	.143	.1425	.145	.151	.157	.162	.166	.1735
93	.182	.1867	.1895	.191	.191	.191	.191	.191
94	.143	.1425	.145	.151	.157	.162	.166	.1735
95	.182	.1867	.1895	.191	.191	.191	.191	.191
96	.143	.1425	.145	.151	.157	.162	.166	.1735
97	.182	.1867	.1895	.191	.191	.191	.191	.191
98	.179	.1795	.1825	.188	.196	.203	.210	.217
99	.227	.231	.232	.232	.232	.232	.232	.232
100	.143	.1425	.145	.151	.157	.162	.166	.1735
101	.182	.1867	.1895	.191	.191	.191	.191	.191
102	.143	.1425	.145	.151	.157	.162	.166	.1735
103	.182	.1867	.1895	.191	.191	.191	.191	.191
104	.143	.1425	.145	.151	.157	.162	.166	.1735
105	.182	.1867	.1895	.191	.191	.191	.191	.191
106	.179	.1795	.1825	.188	.196	.203	.210	.217
107	.227	.231	.232	.232	.232	.232	.232	.232
108	.175	.169	.171	.176	.184	.192	.201	.2095

109	.216	.219	.22	.22	.22	.22	.22	.22
110	.175	.169	.171	.176	.184	.192	.201	.2095
111	.216	.219	.22	.22	.22	.22	.22	.22
112	.175	.169	.171	.176	.184	.192	.201	.2095
113	.216	.219	.22	.22	.22	.22	.22	.22
114	.205	.204	.205	.209	.214	.22	.226	.233
115	.24	.247	.254	.262	.262	.262	.262	.262
116	.175	.169	.171	.176	.184	.192	.201	.2095
117	.216	.219	.22	.22	.22	.22	.22	.22
118	.175	.169	.171	.176	.184	.192	.201	.2095
119	.216	.219	.22	.22	.22	.22	.22	.22
120	.175	.169	.171	.176	.184	.192	.201	.2095
121	.216	.219	.22	.22	.22	.22	.22	.22
122	.205	.204	.205	.209	.214	.22	.226	.233
123	.24	.247	.254	.262	.262	.262	.262	.262
124	16 4 4	2.	.366667	10.	CM PRIME	PER DELTA	R OR Q	
125	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
126	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
127	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
128	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
129	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
130	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
131	-.76	-.75	-.753	-.771	-.8	-.83	-.857	-.886
132	-.917	-.95	-.98	-1.01	-1.01	-1.01	-1.01	-1.01
133	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
134	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
135	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
136	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
137	-.69	-.678	-.68	-.69	-.71	-.73	-.76	-.787
138	-.81	-.83	-.84	-.85	-.85	-.85	-.85	-.85
139	-.76	-.75	-.753	-.771	-.8	-.83	-.857	-.886
140	-.917	-.95	-.98	-1.01	-1.01	-1.01	-1.01	-1.01
141	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
142	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
143	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
144	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
145	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
146	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
147	-.865	-.84	-.83	-.848	-.87	-.893	-.92	-.94
148	-.965	-.994	-1.02	-1.05	-1.05	-1.05	-1.05	-1.05
149	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
150	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
151	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
152	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
153	-.795	-.783	-.786	-.795	-.81	-.83	-.862	-.898
154	-.922	-.935	-.93	-.9	-.9	-.9	-.9	-.9
155	-.865	-.84	-.83	-.848	-.87	-.893	-.92	-.94
156	-.965	-.994	-1.02	-1.05	-1.05	-1.05	-1.05	-1.05
157	16 4 4	2.	.366667	10.	CL PRIME	PER DELTA	P	
158	.13	.127	.125	.124	.123	.122	.1225	.124
159	.124	.123	.12	.116	.116	.116	.116	.116
160	.13	.127	.125	.124	.123	.122	.1225	.124
161	.124	.123	.12	.116	.116	.116	.116	.116
162	.13	.127	.125	.124	.123	.122	.1225	.124

163	.124	.123	.12	.116	.116	.116	.116	.116
164	.143	.14	.1375	.135	.133	.131	.13	.129
165	.128	.1285	.13	.132	.132	.132	.132	.132
166	.13	.127	.125	.124	.123	.122	.1225	.124
167	.124	.123	.12	.116	.116	.116	.116	.116
168	.13	.127	.125	.124	.123	.122	.1225	.124
169	.124	.123	.12	.116	.116	.116	.116	.116
170	.13	.127	.125	.124	.123	.122	.1225	.124
171	.124	.123	.12	.116	.116	.116	.116	.116
172	.143	.14	.1375	.135	.133	.131	.13	.129
173	.128	.1285	.13	.132	.132	.132	.132	.132
174	.142	.1455	.146	.144	.14	.138	.137	.136
175	.1355	.1345	.134	.134	.134	.134	.134	.134
176	.142	.1455	.146	.144	.14	.138	.137	.136
177	.1355	.1345	.134	.134	.134	.134	.134	.134
178	.142	.1455	.146	.144	.14	.138	.137	.136
179	.1355	.1345	.134	.134	.134	.134	.134	.134
180	.148	.146	.144	.142	.14	.139	.138	.137
181	.136	.136	.1355	.135	.135	.135	.135	.135
182	.142	.1455	.146	.144	.14	.138	.137	.136
183	.1355	.1345	.134	.134	.134	.134	.134	.134
184	.142	.1455	.146	.144	.14	.138	.137	.136
185	.1355	.1345	.134	.134	.134	.134	.134	.134
186	.142	.1455	.146	.144	.14	.138	.137	.136
187	.1355	.1345	.134	.134	.134	.134	.134	.134
188	.148	.146	.144	.142	.14	.139	.138	.137
189	.136	.136	.1355	.135	.135	.135	.135	.135
190	999							
191	1							
192	TOTAL SYSTEM CHECKOUT RUN FOR DR J. ROWLAND, 7 APRIL 1972.							
193	.0025	15.0		40				

VITA

Vijayendra Mohan Gupta

Candidate for the Degree of

Master of Science

Thesis: AN EFFICIENT COVARIANCE MATRIX IMPLEMENTATION FOR LARGE-SCALE SYSTEMS

Major Field: Electrical Engineering

Biographical:

Personal Data: Born in Chiswick, London, U.K, January 11, 1949, the son of Mr. and Mrs. M. C. Gupta.

Education: Graduated from Government Higher Secondary School, New Delhi, India, in May, 1966; received Bachelor of Engineering (Honours) degree from Birla Institute of Technology and Science, Pilani, Rajasthan, India in 1971; completed the requirements for Master of Science degree in May, 1973.

Professional Experience: Graduate Research Assistant, School of Electrical Engineering, Oklahoma State University, beginning in the spring 1972.

Professional Organizations: Member of the Institute of Electrical and Electronics Engineers.