

AD-A051 659

FEDERAL ELECTRIC CORP VANDENBERG AFB CALIF
COMPUTATION OF STATE VECTOR AND TRANSITION MATRIX FOR TRAM.(U)

F/G 9/2

JUN 77 G T TOMPSON

F04701-72-C-0203

UNCLASSIFIED

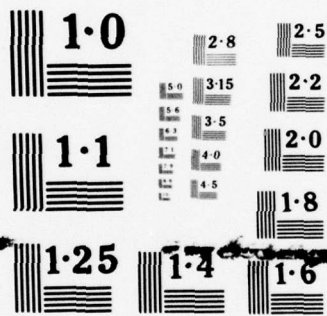
PA100-77-18

SAMTEC-TR-77-223

NL

1 OF 1
ADA
051659

END
DATE
FILMED
4 -78
DDC



NATIONAL BUREAU OF STANDARDS
MICROCOPY RESOLUTION TEST CHART

SAMTEC TR 77-223

AD A051659

2

COMPUTATION OF STATE VECTOR
AND TRANSITION MATRIX FOR TRAM

FEDERAL ELECTRIC CORPORATION
VANDENBERG AFB, CALIF. 93437

1 JUNE 1977

UNLIMITED DISTRIBUTION

Prepared for

SPACE AND MISSILE TEST CENTER
Vandenberg AFB, Calif. 93437

DDC
RECEIVED
MAR 22 1978
D

AD A051659
DDC FILE COPY

This final report was submitted by Federal Electric Corporation, Vandenberg AFB, CA 93437 under Contract FO 4701-72-C-0203 with the Space and Missile Test Center, Vandenberg AFB, CA 93437. Operations Research Analyst, Mr. John McConnell, XRQR, was the Division Scientist-In-Charge.

This report has been reviewed by the Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

John McConnell
JOHN McCONNELL GS-13
Project Scientist

E. Clary
E. CLARY GS-15
Chief, Requirements & Eval.
Division

FOR THE COMMANDER

Robert E. Foster
ROBERT E. FOSTER, Colonel, USAF
Director of Plans, Programs & Resources

020120AQA

19 REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM	
1. REPORT NUMBER 18 SAMTEC TP-77-223	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9 rest.	4. TITLE (and Subtitle) 6 COMPUTATION OF STATE VECTOR AND TRANSITION MATRIX FOR TRAM
7. AUTHOR(s) 19 Dr. Gene T. Tompson	8. CONTRACT OR GRANT NUMBER(s)	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 15 Space and Missile Test Center Contract No. F04701-72-C-0203	11. CONTROLLING OFFICE NAME AND ADDRESS Space and Missile Test Center Code XRQ Vandenberg AFB, CA 93437
9. PERFORMING ORGANIZATION NAME AND ADDRESS Federal Electric Corporation P.O. Box 1886 Vandenberg AFB, CA 93437	12. REPORT DATE 11 1 Jun 77	13. NUMBER OF PAGES 71	14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Same
16. DISTRIBUTION STATEMENT (of this Report) Unlimited distribution	15. SECURITY CLASS. (of this report) 12 75p. Unclassified		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) Unlimited distribution	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A		
18. SUPPLEMENTARY NOTES			
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) State Vector, Transition Matrix Numerical Integration, Numerical Partial Differentiation, Spline Polynomials, Filtering, Algorithms, Subroutines			
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report contains the analysis and documentation of subroutines which contain the capability to efficiently compute the state vector and transition matrix needed for the Kalman Filter in TRAM, a post flight estimation program. The analysis contains the development and testing of algorithms to numerically integrate, perform numerical partial differentiation and interpolate using Spline polynomials. The subroutines, which are documented in this report, use these algorithms to compute the state			

vector and transition matrix for the Kalman Filter. This capability has more general use than for TRAM.

ACCESSION TO		
DTIC	White Section	<input checked="" type="checkbox"/>
DDI	Soft Section	<input type="checkbox"/>
UNANNOUNCED		<input type="checkbox"/>
JUSTIFICATION.....		
BY.....		
DISTRIBUTION/AVAILABILITY CODES		
Dist.	AVAIL. MOD/OF	SPECIAL
A		

TABLE OF CONTENTS

Section		Page
I	INTRODUCTION	1
	1.1 TRAM	1
	1.2 The Estimation in TRAM	1
	1.3 The Purpose of this Study	2
II	CONCLUSIONS	4
	2.1 The Three Areas of Study	4
	2.2 The Conclusions Concerning Numerical Integration	4
	2.3 Conclusions Concerning Numerical Partial Differentiation	5
	2.4 Conclusions on Using Spline Interpolating Polynomials	6
	2.5 Summary of Results	6
III	MATHEMATICAL DESCRIPTION	8
	3.1 The Mathematical Description of the Integration Algorithm	8
	3.2 Algorithms for Numerical Partials	10
	3.3 Spline Interpolation	11
	3.4 The Generalized Transition Matrix	14
IV	DESCRIPTION OF SUBROUTINES	16
	4.1 The Capabilities of the Subroutines	16
	4.2 Subroutine INTCON	16
	4.3 Subroutine INTERP	16
	4.4 Other Subroutines	16
V	NUMERICAL VERIFICATION	17
	5.1 Verification of the Numerical Integration	17
	5.2 Verification of the Accuracy of the Numerical Partials	17
	5.3 Verification of Accuracy of Spline Interpolation	18
	5.4 Numerical Error Summary	19

TABLE OF CONTENTS (Cont)

Section		Page
VI	DOCUMENTATION OF SUBROUTINES	21
	6.1 General Comments on Documentation	21
	6.2 INTCON Subroutine Description	22
	6.3 INTERP Subroutine Description	26
	6.4 INT Subroutine Description	29
	6.5 PVXTRP Subroutine Description	32
	6.6 INTQ Subroutine Description	38
	6.7 DPHINV Subroutine Description	41
	6.8 FMATRX Subroutine Description	44
	6.9 DRGPAR Subroutine Description	47
	6.10 ROTPAR Subroutine Description	49
	6.11 GRVPAR Subroutine Description	52
	6.12 EGRAV Subroutine Description	55
	6.13 DIFFUN Subroutine Description	58
	6.14 FVPMAT Subroutine Description	61
	6.15 CON Subroutine Description	65

COMPUTATION OF STATE VECTOR AND TRANSITION MATRIX FOR TRAM

I. INTRODUCTION

1.1 TRAM (Trajectory Reconstruction Analysis Method)

TRAM will be a system of subroutines produced for SAMTEC which, running under the control of a main program, will use telemetered inertial guidance data and metric sensor data to estimate selected parameters of the missile, the sensors, the earth and the atmosphere. These parameters will include, for example, the position and velocity of the vehicle, coefficients of the error models for the guidance system and for the metric sensors. Normally, parameters of the earth and atmosphere will not be estimated. It will be possible, however, to propagate the effect of errors of these, or any of the parameters not being estimated, on those being estimated. TRAM will be useful for many purposes, such as mission planning and special studies, but it will be used mainly for metric sensor performance analysis and will subsequently become a part of the Metric Integrated Processing System (MIPS).

1.2 The Estimation in TRAM

The parameters being estimated are called the state vector. The estimation is an iterative process which takes place in two steps. The first is the optimal estimation of the state vector and its covariance matrix at any given time using all the data up to and including that time. The second step is to compute the optimal estimate of the state vector and its covariance matrix at any time using all the data. The first step is called filtering. The second is called smoothing.

The filtering in TRAM will be done by the Carlson square root adaptation of the extended Kalman filtering algorithm [1]. The Kalman algorithm is applicable to linear systems and in this application the equations are linearized about a nominal state vector.

There will be two options for smoothing, dependent upon the segment of the trajectory where the data is taken from. In powered flight a fixed interval smoother will be used, and during free fall, the smoothing will be done by retrograde integration. Figure 1 gives a functional flow of the filtering and smoothing TRAM will perform.

1.3 The Purpose of this Study

The calculations within the dashed box in Figure 1 were studied in this effort. The nominal state vector is computed by integrating the position and velocity from one data time to the next. With the nominal state vector, the transition matrix can be computed. The computation of the transition matrix will be by numerical integration also. Since the intervals between data times are normally small (.1 sec or less), and the state vector and transition matrix are needed at each data point, computation times would be excessive. Because of this, more efficient algorithms were required. These were devised and numerical experiments were made to test their execution time and accuracy.

Therefore, the main purpose of this effort was to establish efficient algorithms and subroutines that will compute (1) the state vectors and transition matrices, (2) numerical derivatives and (3) numerical integrals that are needed for TRAM.

TRAM ESTIMATION FLOW CHART

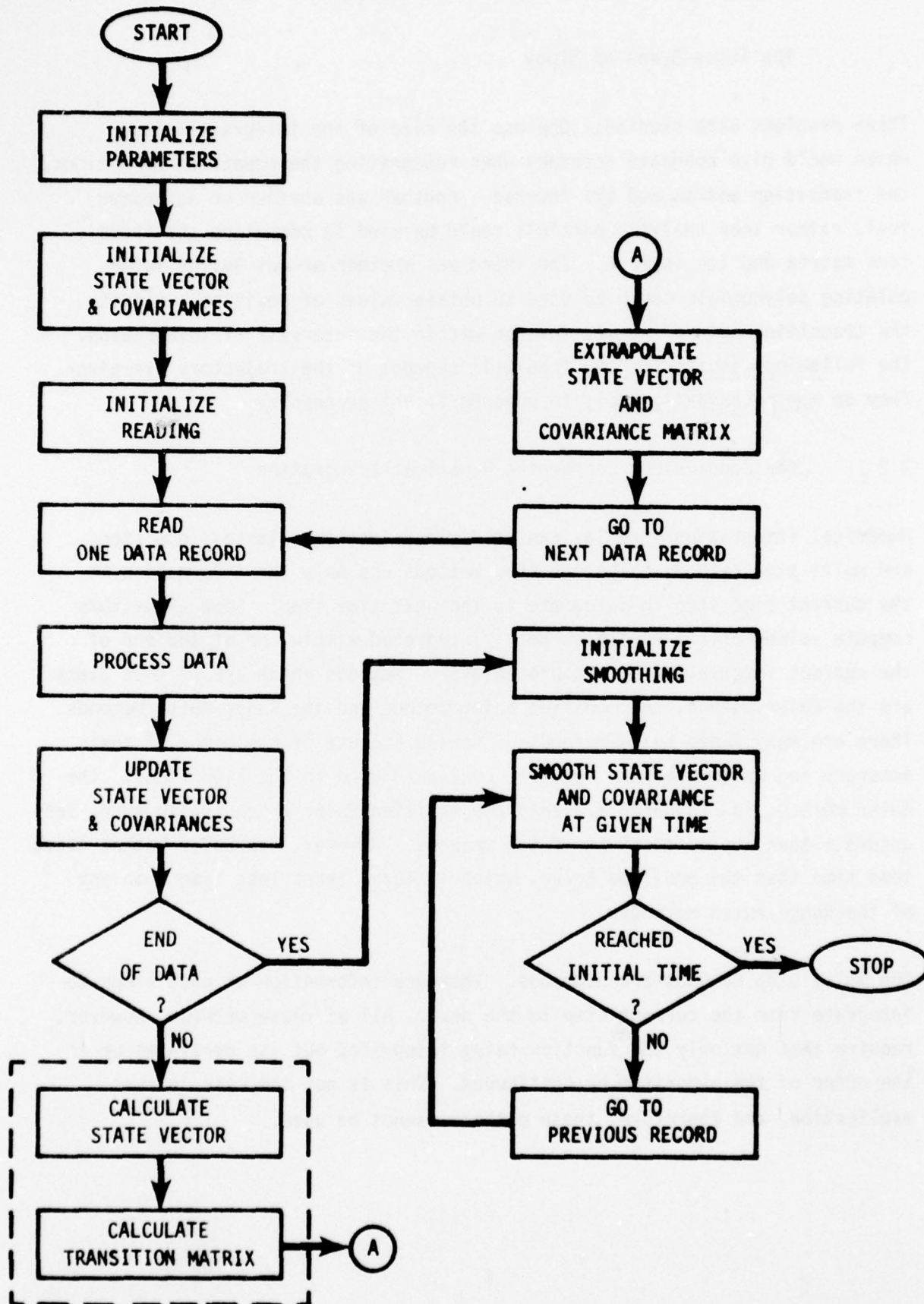


FIGURE 1

II CONCLUSIONS

2.1 The Three Areas of Study

Three problems were studied. One was the size of the integration step which would give adequate accuracy when integrating the equations of motions, the transition matrix and its inverse. Another was whether or not numerical, rather than analytic partials could be used in computing the transition matrix and its inverse. The third was whether or not Spline interpolating polynomials could be used to obtain values of position, velocity, the transition matrix and its inverse within the intervals of integration. The following results for the free-fall segment of the trajectory are given. They do not necessarily apply to powered flight or reentry.

2.2 The Conclusions Concerning Numerical Integration

Numerical integration formulas can be divided into two classes, one step and multi step methods. The one step methods use only the information at the current time step to integrate to the next time step. Some algorithms compute values of the variables being integrated within, or at the end of the current interval being integrated over. Methods which are of this class are the Euler method, the modified Euler method and the Runge-Kutta methods. There are many Runge-Kutta methods differing because of the order of their accuracy and because of the specific constants used in the algorithms. The Euler method, is less accurate than the modified Euler which is, in turn, less accurate than any of the Runge-Kutta methods. However, the Euler method takes less time than the modified Euler, which in turn, takes less time than any of the Runge-Kutta methods.

The multi step methods are numerous. They use information at past steps to integrate from the current step to the next. All of these methods, however, require that not only the function being integrated but its derivatives up to the order of the algorithm be continuous. This is not the case in this application, and therefore, these methods cannot be used.

Results from numerical experimentation have determined that the fourth order Runge-Kutta gives the best accuracy and computation time when compared with the Euler and modified Euler methods. It has further been determined that a step size of five (5) seconds may be used when integrating the equations of motion, the transition matrix and its inverse. These results are valid because, even though the Runge-Kutta takes more time per step than the Euler or modified Euler methods take, the step size can be appreciably larger because of the much greater accuracy of the Runge-Kutta algorithm. Another reason for the larger step size is that during free-fall, acceleration is slowly varying. During powered flight the acceleration is much greater, and a smaller step size will be needed.

2.3 Conclusions Concerning Numerical Partial Differentiation

The differential equations of motion need to be differentiated with respect to position and velocity in order to obtain the transition matrix and its inverse. This study has shown that these partial derivatives can be computed numerically, rather than analytically, with a central difference formula. The study was made by comparing the results of the numerical differentiation with the analytic differentiation.

If the analytic differentiation is done without mathematical errors, and the evaluation of the formulas is done with infinite precision, then the values would be exact. However, analytic partial differentiation normally becomes very detailed and numerous terms are developed. As a result, chances of error are high and very thorough checking is needed. Moreover, if there are a large number of terms, round off errors can degrade the accuracy of the computation.

Often, numerical partial differentiation is simple to program compared with the analytic approach. Once a subroutine to evaluate the function has been developed, all that is required to obtain the partial derivative with respect to a given variable is to evaluate the function for two different values of the variable, leaving all the other variables the same; then, form a quotient of the difference of the values of the function divided by the difference of the two values of the variable. Although this is easy to perform on a

computer, there are truncation and round off errors associated with this method which the analytic approach does not have. The reduction of these errors to an insignificant value was one of the goals which this study achieved.

There is another advantage to obtaining partial derivatives numerically and that is it is much more flexible. The acceleration calculations depend upon the model of gravity acceleration, of course. There are many gravity models and even the form of them can change. To write down the partial derivatives of all of them would be a monumental task indeed. However, partials can be obtained relatively simply by numerical differentiation.

2.4 Conclusions on Using Spline Interpolating Polynomials

The position, velocity, transition matrix can be integrated accurately at step sizes up to five (5) seconds during free-fall, but the values of these quantities will be needed at every tenth second or at smaller intervals. In this study it was determined that values at the finer mesh can be obtained satisfactorily by interpolation with Spline polynomials.

Spline polynomials are constructed such that the function and a specified number of its derivatives fit data at the end points of an interval. In the case of the state vector, the position, velocity, and acceleration are given at the end points. For each of the three variables this specifies six conditions, and therefore, a quintic polynomial is determined. For the transition matrix and its inverse, the function and its derivative are given at the end points of the interval. This specifies four conditions, and therefore, a cubic is determined.

This study has shown also that the velocities in the state vector can be obtained satisfactorily at the finer mesh by evaluating the derivatives of the Spline interpolating polynomials that have been derived for position.

2.5 Summary of Results

The amount of computation time can be reduced appreciably by integrating the state vector, the transition matrix and its inverse over a five second interval and then obtaining these quantities at the finer intervals of the data rate (0.1 second or less) by using Spline interpolating polynomials.

It has been determined that the accuracy of these algorithms is sufficient for the estimation performed in TRAM.

It has been determined also that the partial derivatives needed to calculate the transition matrix can be computed by numerical partial differentiation with accuracy sufficient for TRAM. Computation of derivatives numerically will improve computation times and allow for greater flexibility.

Therefore, the purpose of this effort has been accomplished and algorithms and subroutines have been developed that will compute, (1) the state vectors and transition matrices, (2) partial derivatives, and (3) the numerical integration needed for TRAM.

III MATHEMATICAL DESCRIPTION

3.1 The Mathematical Description of the Integration Algorithm

The differential equations of the motion of a vehicle are given by

$$\begin{aligned}\dot{p}(t) &= v(t) \\ \dot{v}(t) &= a(p(t), v(t))\end{aligned}\tag{3.1.1}$$

with the initial condition of

$$\begin{aligned}p(0) &= p_0 \\ v(0) &= v_0\end{aligned}$$

where $p(t)$ is the position at t , $v(t)$ is the velocity at time t and $a(p(t), v(t))$ is the acceleration. The acceleration has the form

$$a(p(t), v(t)) = a_g(p(t)) + a_r(p(t), v(t)) + a_d(v(t))\tag{3.1.2}$$

Where a_g is the gravitational term and is a function of position, a_r is the acceleration which would be sensed because the equations are in a rotating coordinate system, and a_d is the drag and lift term. In the case under study free fall,

$$a_d = 0.$$

The differential equations of the transition matrix are given by

$$\begin{aligned}\dot{Q}(t) &= F(t)Q(t) \\ Q(0) &= I\end{aligned}\tag{3.1.3}$$

Q is a square matrix of order 6 and F is determined by differentiating equations (3.1.1) with respect to position and velocity. F is, therefore, equal to

$$F = \begin{pmatrix} 0 & I \\ \frac{\partial a(p, v)}{\partial p^T} & \frac{\partial a(p, v)}{\partial v^T} \end{pmatrix}\tag{3.1.4}$$

where each partition of F is a 3 x 3 matrix.

Both (3.1.1) and (3.1.3) are integrated by the same algorithm and, therefore, the equations are written in general form

$$\begin{aligned}\dot{X}(t) &= f(X, t) \\ X(0) &= X_0\end{aligned}\tag{3.1.5}$$

and the algorithm is applied to the general form.

The method used is the standard fourth order Runge-Kutta method [2]. It is described as follows.

Let the values of the variables be given determined at time t_n . Therefore, X_n is known. The values of the variables at time t_{n+1} are then determined by these equations.

$$\begin{aligned}K_1 &= hf(X_n, t_n) \\ K_2 &= hf(X_n + K_1, t_n + h/2) \\ K_3 &= hf(X_n + 1/2 K_2, t_n + h/2) \\ K_4 &= hf(X_n + K_3, t_n + h) \\ X_{n+1} &= X_n + \frac{1}{6} [K_1 + 2K_2 + 2K_3 + K_4]\end{aligned}$$

In the formulas above h is the step size and the function f is the right hand side of the differential equations. X represents the unknown variables being integrated.

3.2 Algorithms for Numerical Partial

From equation (3.1.4) it is seen that the partials of acceleration with respect to position and velocity are needed. Equation (3.1.2) gives the form of the acceleration, with $a_d = 0$.

The gravity a_r is the rotation term and has a very simple form which can be differentiated analytically with ease.

The gravity term a_g may vary markedly in its form and the number of terms it has. This leads to complicated analytic partials and, therefore, the numerical approach was used. The equation can be derived simply from a Taylor series expansion.

Let

$$X_0 + \Delta X = X_1$$

$$X_0 - \Delta X = X_{-1}$$

$$f(X_1) = f(X_0) + \Delta X f'(X_0) + \frac{\Delta X^2}{2!} f''(X_0) + \frac{\Delta X^3}{3!} f'''(\eta) \quad (3.2.1)$$

where

$$X_0 \leq \eta \leq X_1$$

$$f(X_{-1}) = f(X_0) - \Delta X f'(X_0) + \frac{\Delta X^2}{2!} f''(X_0) - \frac{\Delta X^3}{3!} f'''(\xi) \quad (3.2.2)$$

where

$$X_{-1} \leq \xi \leq X_0$$

Subtracting the second equation from the first, we have

$$f(X_1) - f(X_{-1}) = 2\Delta X f'(X_0) + \frac{\Delta X^3}{3!} [f'''(\eta) + f'''(\xi)] \quad (3.2.3)$$

Therefore

$$\frac{f(X_1) - f(X_{-1})}{2\Delta X} = f'(X_0) + \frac{\Delta X^2}{12} [f'''(\eta) + f'''(\xi)] \quad (3.2.4)$$

this is the central difference formula for the derivation of $r(x)$ at x_0 . The truncation error is given by the second term on the right hand side of the equation. The truncation error is seen to be of order $(\Delta X)^2$. Therefore, the formula used to compute the partials is

$$\frac{\partial A_g(x)}{\partial x} = \frac{A_g(x + \Delta X) - A_g(x - \Delta X)}{2\Delta X} \quad (3.2.4)$$

where the value of ΔX was determined to be five.

3.3 Spline Interpolation

Spline interpolating polynomials are used to obtain values of the state vector and the transition matrix within the five second interval of integration. The state vector uses a quintic Spline, and the transition matrix and its inverse uses a cubic Spline. The development that follows is general and therefore, applies to both cases.

Suppose a function $f(t)$ and its derivatives up to and including n th order are given over an interval $t_a \leq t \leq t_b$. Because of improvements in the round off errors, the interpolating polynomials are developed over the interval $[0, 1]$.

Let

$$u = (t - t_a) / T$$

where

$$T = t_b - t_a.$$

Define $g(u) \equiv f(t_a + uT) \quad 0 \leq u \leq 1$

Then

$$\begin{aligned} g'(u) &= T f'(t_a + uT) \\ g''(u) &= T^2 f''(t_a + uT) \\ &\vdots \\ g^{(n)}(u) &= T^n f^{(n)}(t_a + uT) \end{aligned}$$

where the prime denotes differentiation with respect to the variable in the parenthesis.

Let the spline for $g(u)$ on the interval $[0, 1]$ be $\tilde{g}(u)$; and $\tilde{f}(t)$ be defined as follows:

$$\tilde{f}(t) \equiv \tilde{g}((t - t_a)/T)$$

Then

$$\tilde{f}'(t) = \frac{1}{T} \tilde{g}'((t - t_a)/T)$$

$$\vdots$$

$$\tilde{f}^{(n)}(t) = \frac{1}{T^n} \tilde{g}^{(n)}((t - t_a)/T)$$

What is given is $f(t_a)$, $f'(t_a)$, ..., $f^{(n-1)}(t_a)$, $f(t_b)$, $f'(t_b)$, ..., $f^{(n-1)}(t_b)$. From these the following is computed.

$$g(0) = f(t_a)$$

$$g'(0) = Tf'(t_a)$$

$$\vdots$$

$$g^{(n-1)}(0) = T^{n-1} f^{(n-1)}(t_a)$$

$$g(1) = f(t_b)$$

$$g'(1) = Tf'(t_b)$$

$$\vdots$$

$$g^{(n-1)}(1) = T^{n-1} f^{(n-1)}(t_b)$$

From these it is possible to get the coefficients of the normalized Spline on $[0, 1]$:

$$\tilde{g}(u) = A_0 + A_1 u + \dots + A_{2n-1} u^{2n-1},$$

by solving the system of linear equations

$$\tilde{g}(0) = g(0)$$

$$\tilde{g}'(0) = g'(0)$$

$$\vdots$$

$$\begin{aligned} \tilde{g}^{(n-1)}(0) &= g^{(n-1)}(0) \\ \tilde{g}(1) &= g(1) \\ \tilde{g}'(1) &= g'(1) \\ &\vdots \\ \tilde{g}^{(n-1)}(1) &= g^{(n-1)}(1) \end{aligned}$$

For the quintic Spline the system of equations would be

$$\begin{aligned} A_0 &= g(0) \\ A_1 &= g'(0) \\ A_2 &= g''(0)/2 \\ A_5 + A_4 + A_3 + A_2 + A_1 + A_0 &= g(1) \\ 5A_5 + 4A_4 + 3A_3 + 2A_2 + A_1 &= g'(1) \\ 20A_5 + 12A_4 + 6A_3 + 2A_2 &= g''(1) \end{aligned}$$

The first three equations give the values for A_0 , A_1 , and A_2 . The values for A_5 , A_4 and A_3 are determined by the last three equations. This, then, determines the Spline polynomial for position in the state vector. The velocity is determined by differentiating this polynomial and dividing the derivative by T . This would be done for each variable, x , y and z .

The equations for the transition matrix or its inverse would be

$$\begin{aligned} A_0 &= g(0) \\ A_1 &= g'(0) \\ A_3 + A_2 + A_1 + A_0 &= g(1) \\ 3A_3 + 2A_2 + A_1 &= g'(1) \end{aligned}$$

These equations can be easily solved and the coefficients for the cubic are thereby determined. A cubic Spline is needed for each of the 36 elements of the transition matrix and its inverse.

Then in general, \tilde{f} is the nth order Spline segment on $[t_a, t_b]$ approximating f , where

$$\tilde{f}(t) = \tilde{g}(u) \Big|_{u = \frac{t - t_a}{T}}$$

$$\tilde{f}'(t) = \frac{1}{T} \tilde{g}'(u) \Big|_{u = \frac{t - t_a}{T}}$$

⋮

$$\tilde{f}^{(n-1)}(t) = \frac{1}{T^{n-1}} \tilde{g}^{(n-1)}(u) \Big|_{u = \frac{t - t_a}{T}}$$

3.4 The Generalized Transition Matrix

The solution of the differential equation

$$\dot{X}(t) = F(t)X(t) + G(t)u(t)$$

$$X(t_0) = X_0$$

is given by [3].

$$X(t) = \Phi(t, t_0) [X_0 + \int_{t_0}^t \Phi(t_0, \sigma) G(\sigma) u(\sigma) d\sigma]$$

where $\Phi(t, t_0)$ satisfies the differential equation

$$\dot{\Phi}(t, t_0) = F(t)\Phi(t, t_0), \quad \Phi(t_0, t_0) = I$$

Now

$$\Phi(t, t_0)\Phi^{-1}(t, t_0) = I$$

Therefore

$$\dot{\Phi}(t, t_0)\Phi^{-1}(t, t_0) + \Phi(t, t_0)\dot{\Phi}^{-1}(t, t_0) = 0$$

and

$$\dot{\Phi}^{-1}(t, t_0) = -\Phi^{-1}(t, t_0)\dot{\Phi}(t, t_0)\Phi^{-1}(t, t_0)$$

$$\dot{\Phi}^{-1}(t, t_0) = -\Phi^{-1}(t, t_0)F(t)$$

$$\Phi^{-1}(t_0, t_0) = I$$

Now let $u(t) = 0$, and let x_0 be arbitrary.

Then

$$x(t) = \Phi(t, t_0)x_0$$

$$x(s) = \Phi(s, t_0)x_0$$

But also

$$x(s) = \Phi(s, t)x(t)$$

Therefore

$$\Phi(s, t_0) = \Phi(s, t)\Phi(t, t_0)$$

Now let $s = t_0$

$$I = \Phi(t_0, t)\Phi(t, t_0)$$

and so

$$\Phi(t_0, t) = \Phi^{-1}(t, t_0)$$

From the equation

$$\Phi(s, t) = \Phi(s, t_0)\Phi(t_0, t)$$

It is clear that

$$\Phi(s, t) = \Phi(s, t_0)\Phi^{-1}(t, t_0) \quad (3.4.1)$$

Since s and t are arbitrary, (3.4.1) is a way to calculate a transition matrix from one arbitrary point to another.

IV DESCRIPTION OF SUBROUTINES

4.1 The Capabilities of the Subroutines

With the mathematics set forth, it is now possible to describe how it was implemented. There were two subroutines developed. INTCON integrates the state vector and the transition matrix, and then calculates the coefficients for the interpolating Spline polynomials. It uses the Runge-Kutta integration scheme, and when the transition matrix is integrated, the partial derivatives in the F matrix, are calculated numerically. Moreover, it computes the Spline polynomial coefficients as described in Section 3.4.

The second subroutine developed was INTERP. It calculates the interpolated values for the state vector and the transition matrix. The transition matrix is from one arbitrary point to another. The mathematics of Sections 3.3 and 3.4 have been employed in the algorithms in this subroutine. INTERP must be called after INTCON.

4.2 Subroutine INTCON

The subroutine INTCON calls two subroutines INT and CON. INT performs the integration and CON calculates the coefficients of the Spline polynomials.

4.3 Subroutine INTERP

The subroutine INTERP first calculates the interpolated values of the state vector for an arbitrary time t_i . It then calculates the transition from the last time t_{i-1} to the current t_i . It does this by finding the inverse of the transition matrix from the beginning of the interval to t_{i-1} . Then it finds by interpolation the transition matrix from the beginning of the interval to t_i and multiplies the two matrices together.

4.4 Other Subroutines

These two subroutines call others. Altogether there were fourteen subroutines developed.

V. NUMERICAL VERIFICATION

5.1 Verification of the Numerical Integration

The first steps taken to determine which of the one step methods would be adequate were to run the three methods, the Euler method, the modified Euler method and the Runge-Kutta method, over a large segment of the trajectory and see if adequate answers could be obtained. Only the Runge-Kutta gave answers which were accurate enough.

With the method of integration selected it was then possible to determine an optimum step size which would give the best accuracy. A step size of one second and a step size of five seconds gave answers which agreed to more than 0.001 foot in position and to 0.00001 foot per second in velocity when integrated over an interval of 1500 seconds. Consequently, the five second step size was chosen.

The magnitudes of the round off and transition errors caused by the numerical integration are well below errors from other sources and therefore, will not affect the estimation capability in TRAM. The total CPU time to integrate the state vector, the transition matrix and its inverse 1500 seconds was two minutes and 32 seconds. This is an acceptable execution time.

5.2 Verification of the Accuracy of the Numerical Partialis

The transition matrix is initialized to the identity matrix every five seconds and then integrated for a five second span. To bound the error caused by the use of the numerical partialis, this was done twice, once using analytic partialis and once using numerical partialis, and the difference between the two transition matrices was computed. This difference matrix is then multiplied by a vector which has as its elements the largest error which can occur at any step in position and velocity. The product of the matrix times the vector gives the maximum error that can occur at each step caused by the use of numerical partialis. If the resulting vector is multiplied by the number of steps, a bound for the total error due to numerical partialis is obtained.

In the error matrix which was obtained by subtracting the transition matrix computed using analytic partials from the transition matrix using numerical partials no term was larger than 10^{-14} . This meant that the terms which propagate position and velocity errors into position and the terms which propagate position and velocity errors into velocity were all less than 10^{-14} . Further, the maximum TRAM estimation error which can be made in position at any step is less than 100 feet and in velocity it is one foot per second. Since the data could be taken at one tenth second for 1500 seconds, there would be 15000 steps. These numbers provide the following bounds for the errors caused by the use of numerical partials.

$$\begin{pmatrix} \epsilon_{1p} \\ \epsilon_{1v} \end{pmatrix} \leq 15000 \times \begin{pmatrix} 10^{-14} & 10^{-14} \\ 10^{-14} & 10^{-14} \end{pmatrix} \times \begin{pmatrix} 10^2 \\ 1 \end{pmatrix}$$

$$\epsilon_{1p} \leq 15.15 \times 10^{-9} \text{ feet}$$

$$\epsilon_{1v} \leq 15.15 \times 10^{-9} \text{ feet/sec}$$

where ϵ_{1p} is the error in position caused by the use of numerical partials and ϵ_{1v} is the error in velocity caused by the use of numerical partials. Clearly, these errors are insignificant.

5.3 Verification of Accuracy of Spline Interpolation

To determine what the accuracy of the Spline interpolation was, the trajectory was integrated to 1500 seconds. At the beginning, at mid trajectory and at the end of the trajectory, Spline polynomials were fit over the five second interval. The state vector, the transition matrix and its inverse were approximated when analytic and also when numerical partials were used.

The error in position in the state vector was always less than 10^{-4} feet and the error in velocity was less than 10^{-5} feet per second. In the transition matrix, the term which propagates errors in position into position is less than 10^{-9} , and the term which propagates velocity error into position is less than 10^{-7} . The term which propagates position error into velocity was less than 10^{-11} , and the term which propagates velocity error into

velocity was 10^{-9} . Since position estimation errors are less than 100 feet per step, and velocity estimation errors are less than one foot per step and there are 15000 steps, the error terms are bounded by

$$\begin{pmatrix} \epsilon_{2p} \\ \epsilon_{2v} \end{pmatrix} \leq \begin{pmatrix} 10^{-9} & 10^{-7} \\ 10^{-11} & 10^{-9} \end{pmatrix} \times \begin{pmatrix} 10^2 \\ 1 \end{pmatrix} \times 15000$$

where ϵ_{2p} and ϵ_{2v} are errors in position and velocity respectively which are caused because of Spline interpolation.

Therefore

$$\begin{aligned} \epsilon_{2p} &\leq 3 \times 10^{-3} \text{ feet} \\ \epsilon_{2v} &\leq 3 \times 10^{-5} \text{ feet/sec.} \end{aligned}$$

5.4 Numerical Error Summary

For the state vector, the two sources of error, one from numerical integration and the other from Spline interpolation cause the following total error

$$\begin{aligned} \eta_p &\leq 10^{-3} + 10^{-4} = .0011 \text{ feet} \\ \eta_v &\leq 10^{-5} + 10^{-5} = 2 \times 10^{-5} \text{ feet/second} \end{aligned}$$

where η_p and η_v are position and velocity errors because of numerical approximation. Both of these errors are insignificant.

For the transition matrix the following total error bounds are valid.

$$\begin{aligned} \epsilon_{pt} &\leq \epsilon_{1p} + \epsilon_{2p} = 2 \times 10^{-8} + 3 \times 10^{-3} < 3.1 \times 10^{-3} \\ \epsilon_{vt} &\leq \epsilon_{1v} + \epsilon_{2v} = 2 \times 10^{-8} + 3 \times 10^{-5} < 3.1 \times 10^{-5} \end{aligned}$$

Both of these errors are insignificant. It may be concluded, therefore, that the errors due to the numerical approximations discussed here will in no way impair the estimation accuracy in TRAM and further, it may be concluded that these algorithms will greatly improve the speed and flexibility of the computation of the estimation.

VI DOCUMENTATION OF SUBROUTINES

6.1 GENERAL COMMENTS ON DOCUMENTATION

The documentation which follows is intended to be sufficient enough so that the capabilities of these subroutines can be understood, used and modified. The function that the subroutine performs is stated first. This is followed by a description of the input and output variables. The restrictions, if any, under which the subroutine must be used come next, and then a list of the subroutines this subroutine calls is given. The documentation also includes a listing of the subroutine and a flow chart.

The main subroutines are INTCON and INTERP, but these call others which are documented also.

6.2 INTCON SUBROUTINE DESCRIPTION

FUNCTION: Integrates the state vector, the transition matrix and its inverse over the interval of integration and then computes the coefficients for the interpolating Spline polynomials. A quintic Spline is determined for the state vector and a cubic Spline is determined for the transition matrix and its inverse.

INPUT:

/VAR/

X Nine dimensional state vector containing position, velocity and acceleration. Only position and velocity are needed as input to INTCON.

T Time at the beginning of the interval of integration.

TS Start time of the trajectory

H Step size of the integration

ID Flag to indicate whether or not the inverse of the transition matrix should be integrated.

OUTPUT:

/VAR/

X Nine dimensional state vector which contains position, velocity and acceleration integrated to the end of the interval

PHI 6x6 dimensional array containing elements of the transition matrix at the end of the interval of integration

PHID 6x6 dimensional array containing the derivatives of PHI

PHIN 6x6 dimensional array containing the elements of the inverse of PHI

PHIND 6x6 dimensional array containing the derivatives of PHIN

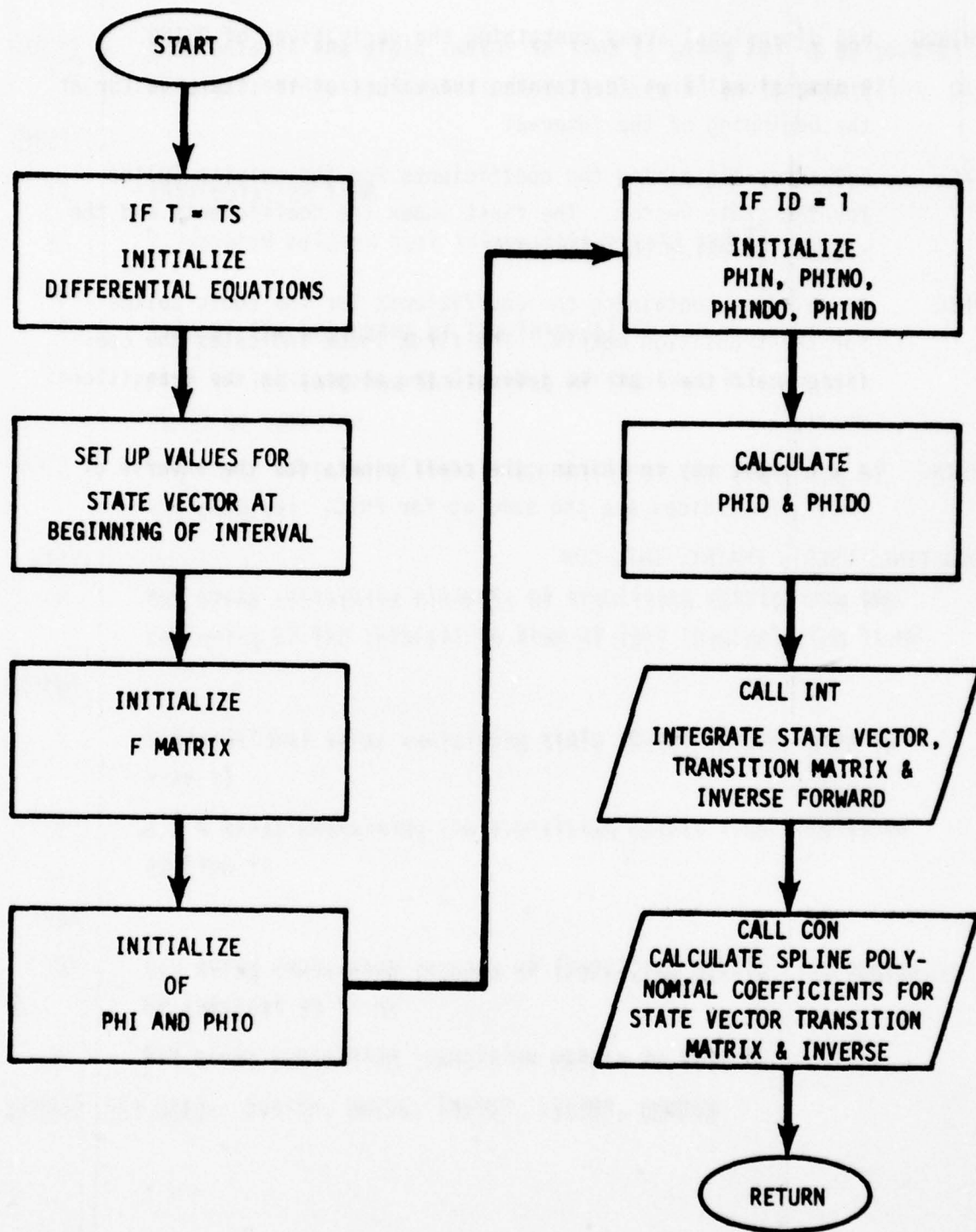
PHIO 6x6 dimensional array containing the elements of the transition matrix at the beginning of the interval of integration

PHIDO 6x6 dimensional array containing the derivatives of the elements of PHIO

- PHINO 6x6 dimensional array containing the inverse of PHIO
- PHINDO 6x6 dimensional array containing the derivatives of PHINO
- XO 9 dimensional array containing the values of the state vector at the beginning of the interval
- XC 6x3 array containing the coefficients for the quintic Spline for the state vector. The first index the coefficients and the second indicates the variable.
- PHIC 4x6x6 array containing the coefficients for the cubic Spline for the transition matrix. The first index indicates the coefficient and the last two indicate the element in the transition matrix
- PHINC 4 x 6 x 6 array containing the coefficients for the inverse of PHI. The indices are the same as for PHIC.

SUBROUTINES USED: FMATRX, INT, CON

INTCON FLOWCHART



COMPILER OPTIONS - NAME = MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

```

ISN 0002      SUBROUTINE INTCOM
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      COMMON/VAR/X(9),PHI(6,6),PHID(6,6),PHIN(6,6),PHIND(6,6),
1 PHIO(6,6),PHIOO(6,6),PHINO(6,6),PHINDO(6,6),XO(9),XC(6,3),
2 PHIC(4,6,6),PHINC(4,6,6),TS,T,H,IO
ISN 0005      COMMON/PHIM/F(6,6)
ISN 0006      DIMENSION DX(6)
ISN 0007      IF(T,NE,TS) GO TO 5
ISN 0009      CALL DIFFUN(6,T,X,DX)
ISN 0010      DO 3 I=1,3
ISN 0011      3 X(I*6)=DX(I*3)
ISN 0012      5 CONTINUE
ISN 0013      DO 10 I=1,9
ISN 0014      10 XO(I)=X(I)
ISN 0015      CALL FMATRX(X)
ISN 0016      DO 20 I=1,6
ISN 0017      DO 20 J=1,6
ISN 0018      PHI(I,J)=0D0
ISN 0019      IF(I.EQ.J) PHI(I,J)=1D0
ISN 0021      PHIO(I,J)=PHI(I,J)
ISN 0022      20 CONTINUE
ISN 0023      IF(IO.EQ.0) GO TO 19
ISN 0025      DO 22 I=1,6
ISN 0026      DO 22 J=1,6
ISN 0027      PHIN(I,J)=PHI(I,J)
ISN 0028      PHINO(I,J)=PHI(I,J)
ISN 0029      22 CONTINUE
ISN 0030      DO 18 J=1,6
ISN 0031      DO 18 I=1,6
ISN 0032      PHINDO(I,J)=0D0
ISN 0033      DO 17 K=1,6
ISN 0034      17 PHINDO(I,J)=-PHINO(I,K)*F(K,J)+PHINDO(I,J)
ISN 0035      18 PHINDO(I,J)=PHINDO(I,J)
ISN 0036      19 CONTINUE
ISN 0037      CALL DPHI(PHI,PHID)
ISN 0038      DO 25 I=1,6
ISN 0039      DO 25 J=1,6
ISN 0040      25 PHIDO(I,J)=PHID(I,J)
ISN 0041      CALL INT(X,DX,PHI,PHID,PHIN,PHIND,T,H,IO)
ISN 0042      CALL CON
ISN 0043      RETURN
ISN 0044      END

```

OPTIONS IN EFFECT NAME = MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 43 ,PROGRAM SIZE = 1132

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

67K BYTES OF CORE NOT USED

BEST AVAILABLE COPY

6.3 INTERP SUBROUTINE DESCRIPTION

FUNCTION: Interpolates the state vector to time TI using Spline polynomials, obtains a transition matrix from TI-HS to TI by interpolation

INPUT:

TI Interpolation time

HS Difference between last interpolation time and TI

/VAR/

T Time at the beginning of the interval

ID Flag to indicate how the inverse of the transition matrix is to be computed

PHIO 6x6 array containing transition matrix at the beginning of the interval

/TST/

S 6x6 array containing elements of transition matrix from the beginning of the interval to time of last interpolation TI-HS

OUTPUT:

Y 6 dimensional array containing state vector interpolated to time TI

Q 6 x 6 array containing the transition matrix from time TI-HS to time TI

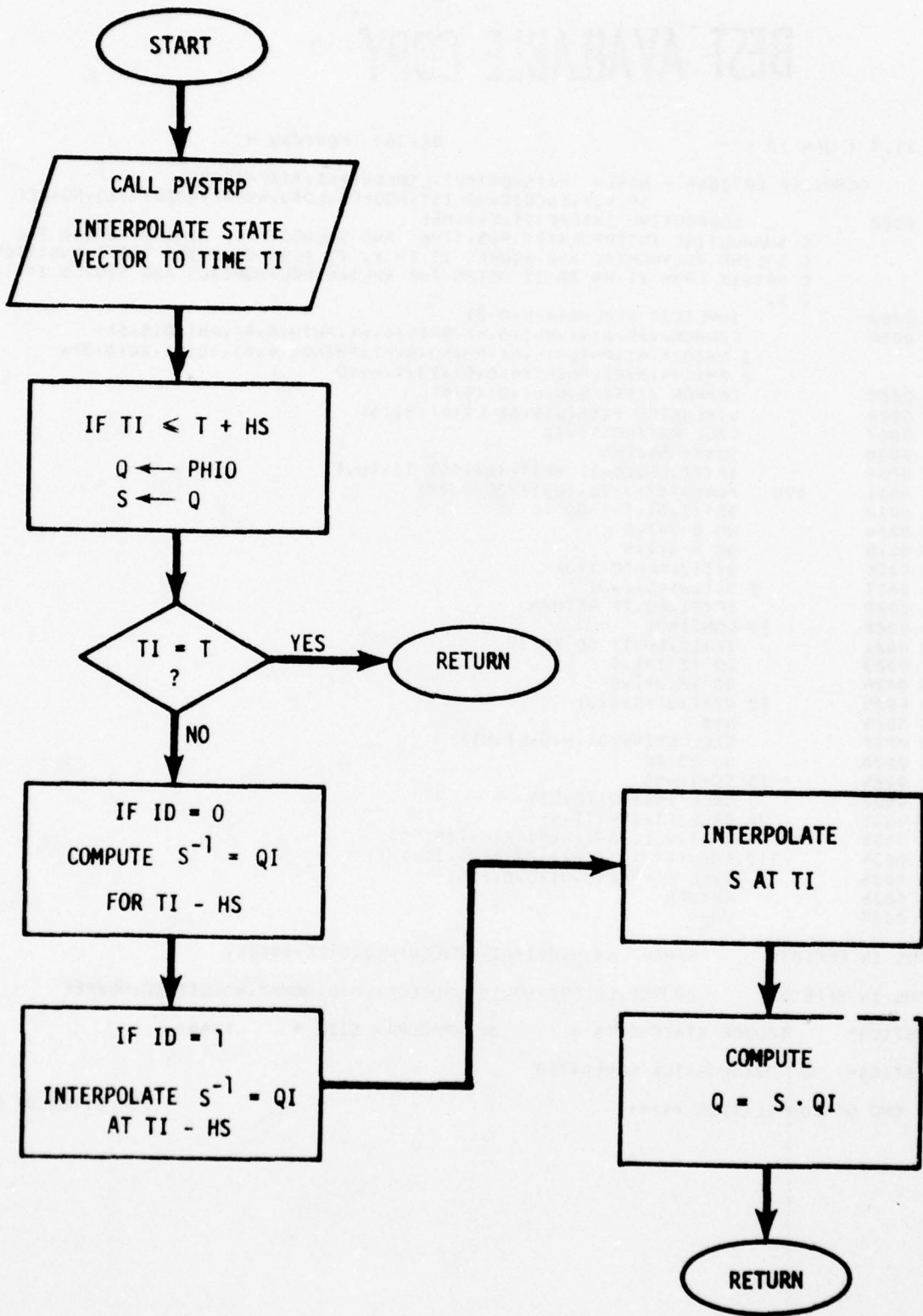
/TST/

QI 6x6 array containing inverse of transition matrix from beginning of interval to TI-HS

S 6x6 array containing transition matrix to time TI

SUBROUTINES USED: PVXTRP, DMINV, INXTRP, FIXTRP, DGMPRD

INIERP FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN M

DATE

```
COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF
ISN 0002      SUBROUTINE INTERP(TI,Y,Q,HS)
C SUBROUTINE INTERPOLATES POSITION AND VELOCITY TO TIME TI USING THE
C SPLINE POLYNOMIAL AND STORES IT IN Y. IT ALSO COMPUTES THE TRANSITION
C MATRIX FROM TI-HS TO TI USING THE SPLINE POLYNOMIALS AND STORES IT IN
C Q.
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      COMMON/VAR/X(9),PHI(6,6),PHID(6,6),PHIN(6,6),PHIND(6,6),
1 PHIO(6,6),PHIDO(6,6),PHINO(6,6),PHINDO(6,6),XO(9),XC(6,3),
2 PHIC(4,6,6),PHINC(4,6,6),TS,T,H,ID
COMMON /TST/ S(6,6),QI(6,6)
ISN 0005      DIMENSION Y(6),Q(6,6),L1(6),M1(6)
ISN 0006      CALL PVXTRP(TI,Y)
ISN 0007      TU=T+HS*1D-6
ISN 0008      IF(TI.LE,2D-1) WRITE(6,100) TI,TU,T
ISN 0009      100 FORMAT(2X,'TI,TU,T',3E20.8/)
ISN 0010      IF(TI.GT,TU) GO TO 10
ISN 0011      DO 5 I=1,6
ISN 0012      DO 5 J=1,6
ISN 0013      Q(I,J)=PHIO(I,J)
ISN 0014      5 S(I,J)=Q(I,J)
ISN 0015      IF(TI.EQ,T) RETURN
ISN 0016      10 CONTINUE
ISN 0017      IF(IID.EQ,1) GO TO 15
ISN 0018      DO 12 I=1,6
ISN 0019      DO 12 J=1,6
ISN 0020      12 QI(I,J)=S(I,J)
ISN 0021      N=6
ISN 0022      CALL DMINV(QI,N,D,L1,M1)
ISN 0023      GO TO 20
ISN 0024      15 TE=TI-HS
ISN 0025      CALL INXTRP(TE,QI)
ISN 0026      20 CALL FIXTRP(TI,S)
ISN 0027      IF(TI.LE,2D-1) WRITE(6,110) S
ISN 0028      110 FORMAT(2X,'S',/6(2X,6E20.10/)/)
ISN 0029      CALL DGMPPD(S,QI,Q,6,6,6)
ISN 0030      RETURN
ISN 0031      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF

STATISTICS SOURCE STATEMENTS = 36 ,PROGRAM SIZE = 1046

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

75K BYTES OF CORE NOT USED

6.4 INT SUBROUTINE DESCRIPTION

FUNCTION: Integrates state vector, transition matrix and its inverse from T to T+H

INPUT:

X 9 dimension array state vector containing position, velocity and acceleration

PHI 6x6 array containing transition matrix at time T

PHIN 6x6 array containing inverse of PHI at time T

ID Flag to indicate if inverse of transition matrix should be integrated

H Step size of integration

OUTPUT:

PS 6x4 array contains four intermediate values of the state vector used in the Runge-Kutta integration algorithm for integrating the transition matrix and its inverse

X 9 dimensional array state vector containing position, velocity and acceleration at T+H

PHI 6x6 array containing transition matrix at T+H

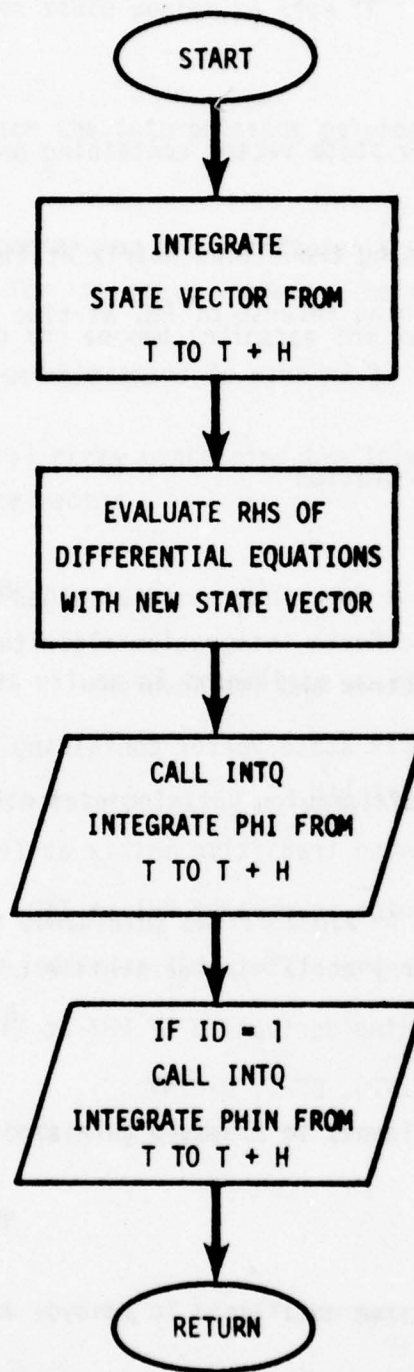
PHIN 6x6 array containing inverse of PHI at T+H

DX 6 dimensional array containing derivative of X at T+H

PHID 6x6 array containing derivative of PHI at T+H

SUBROUTINES USED: DIFFUN, INTQ, DPHI, DPHINV

INT FLOWCHART



COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

ISN 0002

SUBROUTINE INT(X,DX,PHI,PHID,PHIN,PHIND,T,H,ID)

C SUBROUTINE INT GIVES THE VALUES OF POSITION AND VELOCITY, THE
C TRANSITION MATRIX AND ITS DERIVATIVE AND, AS AN OPTION, THE INVERSE
C OF THE TRANSITION AND ITS DERIVATIVE AT TIME T+H. ID IS A FLAG WHICH
C INDICATES WHETHER THE INVERSE OF THE TRANSITION MATRIX AND ITS
C DERIVATIVE IS TO BE COMPUTED. THE POSITION AND VELOCITY ARE STORED IN X,
C THE TRANSITION MATRIX AND ITS INVERSE ARE STORED IN PHI AND PHID.
C THE INVERSE OF TRANSITION MATRIX AND ITS DERIVATIVE ARE STORED IN
C PHIN AND PHIND.

ISN 0003

IMPLICIT REAL*8(A-H,O-Z)

ISN 0004

DIMENSION X(1),DX(1),PHI(6,6),PHID(6,6),PHIN(6,6),PHIND(6,6),
PV(6),PS(6,4),Y(6),Q(6),DY(6),
& C(3),D(4)

ISN 0005

EXTERNAL DPHI,DPHIN

ISN 0006

DATA C/2*0.500,100 /,0/100,2*200,100/

ISN 0007

DO 5 I=1,6

ISN 0008

PV(I)=X(I)

ISN 0009

5 PS(I,1)=X(I)

ISN 0010

N=6

ISN 0011

DO 30 I=1,4

ISN 0012

IF(I.EQ.1) GO TO 15

ISN 0014

K=I-1

ISN 0015

DO 10 J=1,6

ISN 0016

PV(J)=PS(J,1)+C(K)*Q(J)

ISN 0017

10 PS(J,1)=PV(J)

ISN 0018

15 CONTINUE

ISN 0019

CALL DIFFUN(N,T,PV,DY)

ISN 0020

DO 20 J=1,6

ISN 0021

20 Q(J)=H*DY(J)

ISN 0022

DO 25 J=1,6

ISN 0023

25 X(J)=X(J)+Q(J)*D(I)/600

ISN 0024

30 CONTINUE

ISN 0025

CALL DIFFUN(N,T,X,DX)

ISN 0026

DO 31 I=1,3

ISN 0027

31 X(I+6)=DX(I+3)

ISN 0028

CALL INTQ(PS,PHI,PHID,T,H,X,DPHI)

ISN 0029

IF(ID.EQ.1) CALL INTQ(PS,PHIN,PHIND,T,H,X,DPHIN)

ISN 0031

RETURN

ISN 0032

END

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 31 ,PROGRAM SIZE = 1464

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

75K BYTES OF CORE NOT USED

BEST AVAILABLE COPY

6.5 PVXTRP SUBROUTINE DESCRIPTION

FUNCTION: Interpolates state vector to time TE

INPUT:

TE Time at which the interpolating polynomial is to be evaluated.

/VAR/

XC 6x3 array containing the coefficients for the Spline interpolating polynomial for the state vector. The first indicates the coefficient and the second indicates the variable being interpolated.

OUTPUT:

Y 6 dimensional array containing the interpolated values of position of the state vector

ENTRY FIXTRP

FUNCTION: Interpolates values of transition matrix to time TE

INPUT:

TE Time at which interpolating polynomials are to be evaluated

/VAR/

PHIC 4x6x6 array containing coefficients of Spline polynomials. The first index indicates the coefficient and the last two indicate the variable

OUTPUT:

Q 6x6 array containing elements of transition matrix at time TE

ENTRY INXTRP

FUNCTION: Interpolates inverse of transition matrix to time TE

INPUT:

TE Time at which the interpolating polynomial is to be evaluated

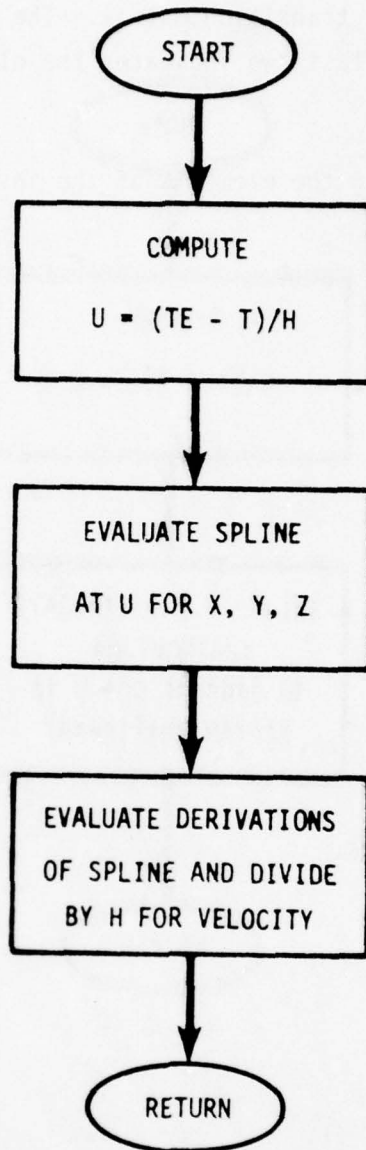
/VAR/

PHINC 4x6x6 array containing coefficients of the Spline polynomials for the inverse of the transition matrix. The first index indicates the array and the last two indicates the element.

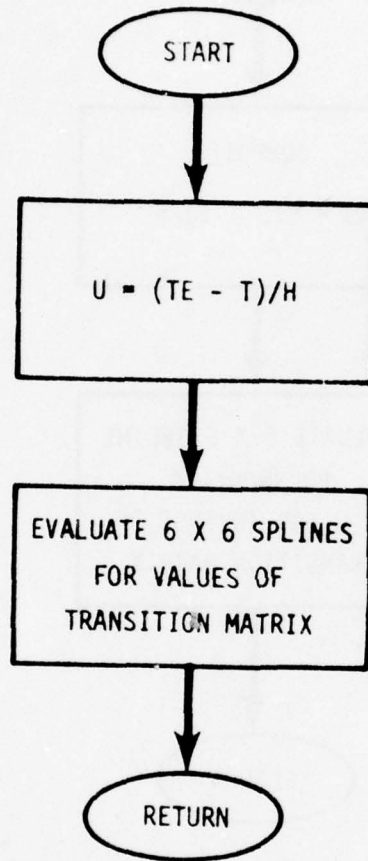
OUTPUT:

S 6x6 array containing the elements of the inverse of the transition matrix for time TE

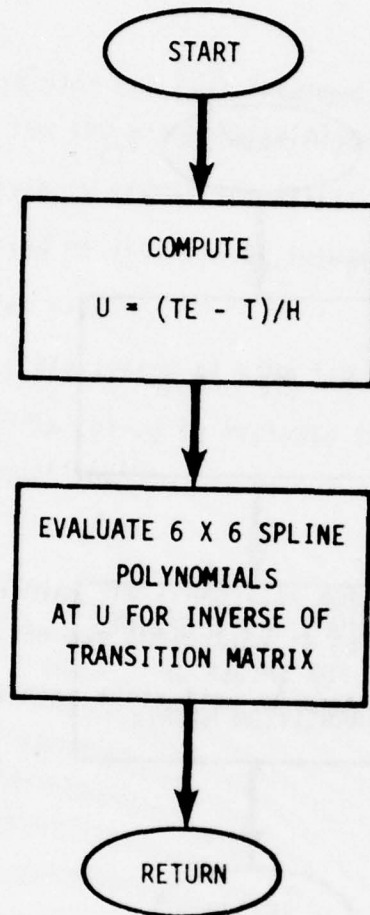
PVXTRP FLOWCHART



ENTRY FIXTRP



ENTRY INXTRP



COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K.

SOURCE=EBDCIC,NOLIST,NODECK,LOAD,NOAP,NOEDIT,NOXREF

```

ISN 0002      SUBROUTINE PVXTRP(TE,Y)
C SUBROUTINE PVXTRP(TE,Y) EXTRAPOLATED POSITION AND VELOCITY TO TIME TE
C USING A QUINTIC SPLINE FOR POSITION AND ITS DERIVATIVE FOR VELOCITY.
C THE COEFFICIENTS FOR THE QUINTIC SPLINE ARE COMPUTED IN SUBROUTINE
C INTCON. INTERPOLATED POSITION AND VELOCITY ARE STORED IN Y.
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      COMMON/VAP/X(9),PHI(6,6),PHID(6,6),PHIN(6,6),PHIND(6,6),
1 PHIO(6,6),PHIDO(6,6),PHINDO(6,6),PHINDO(6,6),XO(9),XC(6,3),
2 PHIC(4,6,6),PHINC(4,6,6),TS,T,H,ID
ISN 0005      DIMENSION Q(6,6)
ISN 0006      DIMENSION S(6,6)
ISN 0007      DIMENSION Y(6)
ISN 0008      U=(TE-T)/H
ISN 0009      DO 10 I=1,3
ISN 0010      J=1+3
ISN 0011      Y(I)= U*U*U*U*XC(1,I)+U*U*U*XC(2,I)+U*U*XC(3,I)+U*XC(4,I)
1 +U*XC(5,I)+XC(6,I)
ISN 0012      Y(J)=500*U*U*U*XC(1,I)+400*U*U*XC(2,I)+300*U*XC(3,I)
1 +200*XC(4,I)+XC(5,I)
ISN 0013      Y(J)=Y(J)/H
ISN 0014      10 CONTINUE
ISN 0015      RETURN
ISN 0016      ENTRY FIXTRP(TE,Q)
C SUBROUTINE FIXTRP(TE,Q) EXTRAPOLATES PHI TO TIME TE USING A CUBIC
C SPLINE. THE COEFFICIENTS FOR THE CUBIC SPLINE ARE COMPUTED IN
C SUBROUTINE INTCON. INTERPOLATED VALUES OF PHI ARE STORED IN Q.
ISN 0017      U=(TE-T)/H
ISN 0018      DO 15 I=1,6
ISN 0019      DO 15 J=1,6
ISN 0020      15 Q(I,J)=U*U*PHIC(1,I,J)+U*PHIC(2,I,J)+PHIC(3,I,J)+PHIC(4,I,J)
ISN 0021      RETURN
ISN 0022      ENTRY INXTRP(TE,S)
C SUBROUTINE INXTRP(TE,S) EXTRAPOLATES THE INVERSE OF PHI TO TIME TE
C USING A CURIC SPLINE. THE COEFFICIENTS FOR THE CURIC SPLINE ARE
C COMPUTED IN SUBROUTINE INTCON. INTERPOLATED VALUES OF PHI INVERSE ARE
C STORED IN S.
ISN 0023      U=(TE-T)/H
ISN 0024      DO 20 I=1,6
ISN 0025      DO 20 J=1,6
ISN 0026      20 S(I,J)=U*U*PHINC(1,I,J)+U*PHINC(2,I,J)+PHINC(3,I,J)+
2 PHINC(4,I,J)
ISN 0027      RETURN
ISN 0028      END

```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K.

OPTIONS IN EFFECT SOURCE=EBDCIC,NOLIST,NODECK,LOAD,NOAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 27 ,PROGRAM SIZE = 1354

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

71K BYTES OF CORE NOT USED

BEST AVAILABLE COPY

6.6 INTQ SUBROUTINE DESCRIPTION

FUNCTION: Integrates both the transition matrix and its inverse from T to T+H. [The subroutine for evaluating the differential equations is passed in via the calling list.]

INPUT:

PS 6x4 array containing the four intermediate values of the state vector needed for the Runge-Kutta algorithm

PHI 6x6 array containing transition matrix or inverse at time T

T Time at beginning of interval of integration

H Integration step size

X 9 dimensional state vector at time T+H

FQ Procedure which is called to evaluate the right hand side of the differential equations

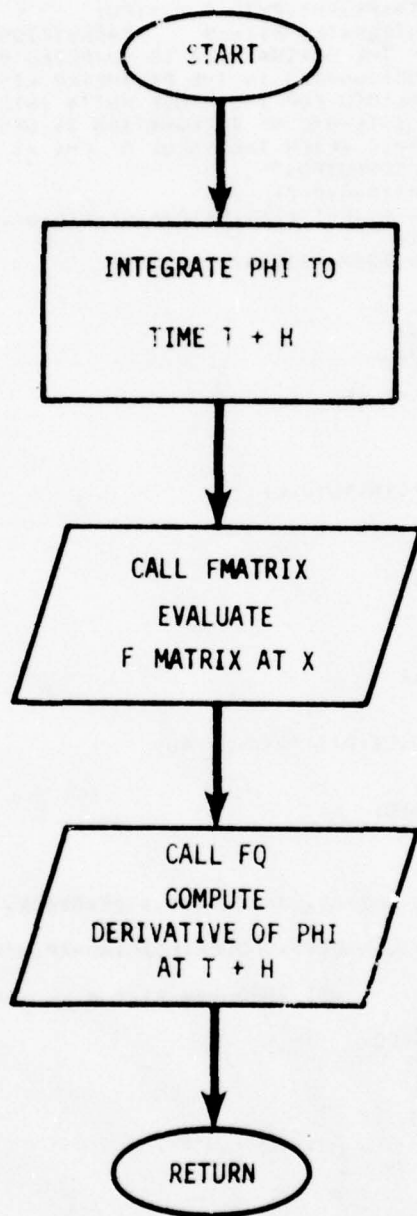
OUTPUT:

PHI 6x6 array containing the transition matrix of its inverse, dependent upon the way subroutine is called, at T+H

PHID 6x6 array containing derivative of what is in PHI at time T+H

SUBROUTINES USED: FMATRIX, FQ

INTQ FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN H

DATE

```
COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF
ISN 0002      SUBROUTINE INTO(PS,PHI,PHID,T,H,X,FQ)
C SUBROUTINE INTO INTEGRATES EITHER      TRANSITION MATRIX FROM T TO T+H.
C THE COMPUTATION OF THE DERIVATIVES IS SUPPLIED BY AN EXTERNAL
C PROCEDURE AND IS DESIGNATED IN THE PARAMETER LIST AS FQ. FS ARE THE
C FOUR VALUES OF X NEEDED FOR THE RUNGE KUTTA INTEGRATION ALGORITHM. THE
C VALUE OF X AT THE INTERVAL OF INTEGRATION IS DESIGNATED AS X. IT IS
C USED TO EVALUATE PHID AFTER THE VALUE OF PHI AT THE END OF THE
C INTERVAL HAS BEEN COMPUTED.
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      DIMENSION PS(6,4),PHI(6,6),PHID(6,6),S(6,6),O(6,6),U(6,6),Y(6),
& C(3),D(4),X(1)
ISN 0005      DATA C/2*0.500,100/,D/100,2*200,100/
ISN 0006      DO 5 I=1,6
ISN 0007      DO 5 J=1,6
ISN 0008      U(I,J)=PHI(I,J)
ISN 0009      5 S(I,J)=PHI(I,J)
ISN 0010      DO 30 I=1,4
ISN 0011      IF(I.EQ.1) GO TO 15
ISN 0013      K=I-1
ISN 0014      DO 10 J=1,6
ISN 0015      DO 10 L=1,6
ISN 0016      10 U(J,L)=S(J,L)*C(K)*Q(J,L)
ISN 0017      15 CONTINUE
ISN 0018      DO 16 J=1,6
ISN 0019      16 Y(J)=PS(J,I)
ISN 0020      CALL FMATRX(Y)
ISN 0021      CALL FQ(U,O)
ISN 0022      DO 17 J=1,6
ISN 0023      DO 17 L=1,6
ISN 0024      17 Q(J,L)=H*Q(J,L)
ISN 0025      DO 25 J=1,6
ISN 0026      DO 25 L=1,6
ISN 0027      25 PHI(J,L)=PHI(J,L)+D(I)*Q(J,L)/600
ISN 0028      30 CONTINUE
ISN 0029      CALL FMATRX(X)
ISN 0030      CALL FQ(PHI,PHID)
ISN 0031      RETURN
ISN 0032      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 31 ,PROGRAM SIZE = 2008

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

71K BYTES OF COPE NOT USED

6.7 DPHINV SUBROUTINE DESCRIPTION

FUNCTION: Computes derivative of the inverse of the transition matrix

INPUT:

Q 6x6 array containing inverse of transition matrix

/FHIM/

F 6x6 array containing coefficients of differential equations for
inverse of transition matrix

OUTPUT:

DQ 6x6 array containing derivatives of the inverse of the transition
matrix

ENTRY DPHI

FUNCTION: Computes derivatives of transition matrix

INPUT:

Q 6x6 array containing transition matrix

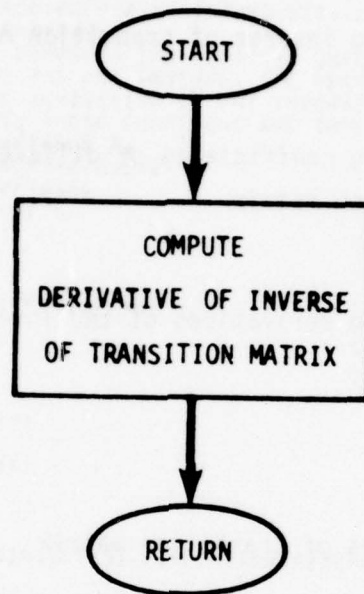
/PHIM/

F 6x6 array containing coefficients of differential equations for
transition matrix

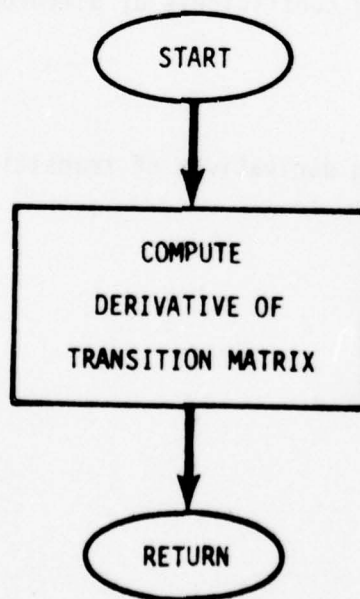
OUTPUT:

DQ 6x6 array containing derivatives of transition matrix

DPHINV FLOWCHART



ENTRY DPHI



```

COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
                  SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF
ISN 0002          SUBROUTINE DPHINV(O,DQ)
ISN 0003          IMPLICIT REAL*8(A-H,O-Z)
ISN 0004          COMMON/PHIM/F(6,6)
ISN 0005          DIMENSION Q(6,6),DQ(6,6)
ISN 0006          DO 10 I=1,6
ISN 0007          DO 10 J=1,6
ISN 0008          DQ(I,J)=000
ISN 0009          DO 10 K=1,6
ISN 0010          10 DQ(I,J)=DQ(I,J)-Q(I,K)*F(K,J)
ISN 0011          RETURN
ISN 0012          ENTRY DPHI(O,DQ)
ISN 0013          DO 20 I=1,6
ISN 0014          DO 20 J=1,6
ISN 0015          DQ(I,J)=000
ISN 0016          DO 20 K=1,6
ISN 0017          20 DQ(I,J)=DQ(I,J)+F(I,K)*Q(K,J)
ISN 0018          RETURN
ISN 0019          END

```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 18 ,PROGRAM SIZE = 800

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

79K BYTES OF COPE NOT USED

BEST AVAILABLE COPY

6.8 FMATRIX SUBROUTINE DESCRIPTION

FUNCTION: Computes matrix which relates transition matrix and its inverse to their derivatives

INPUT:

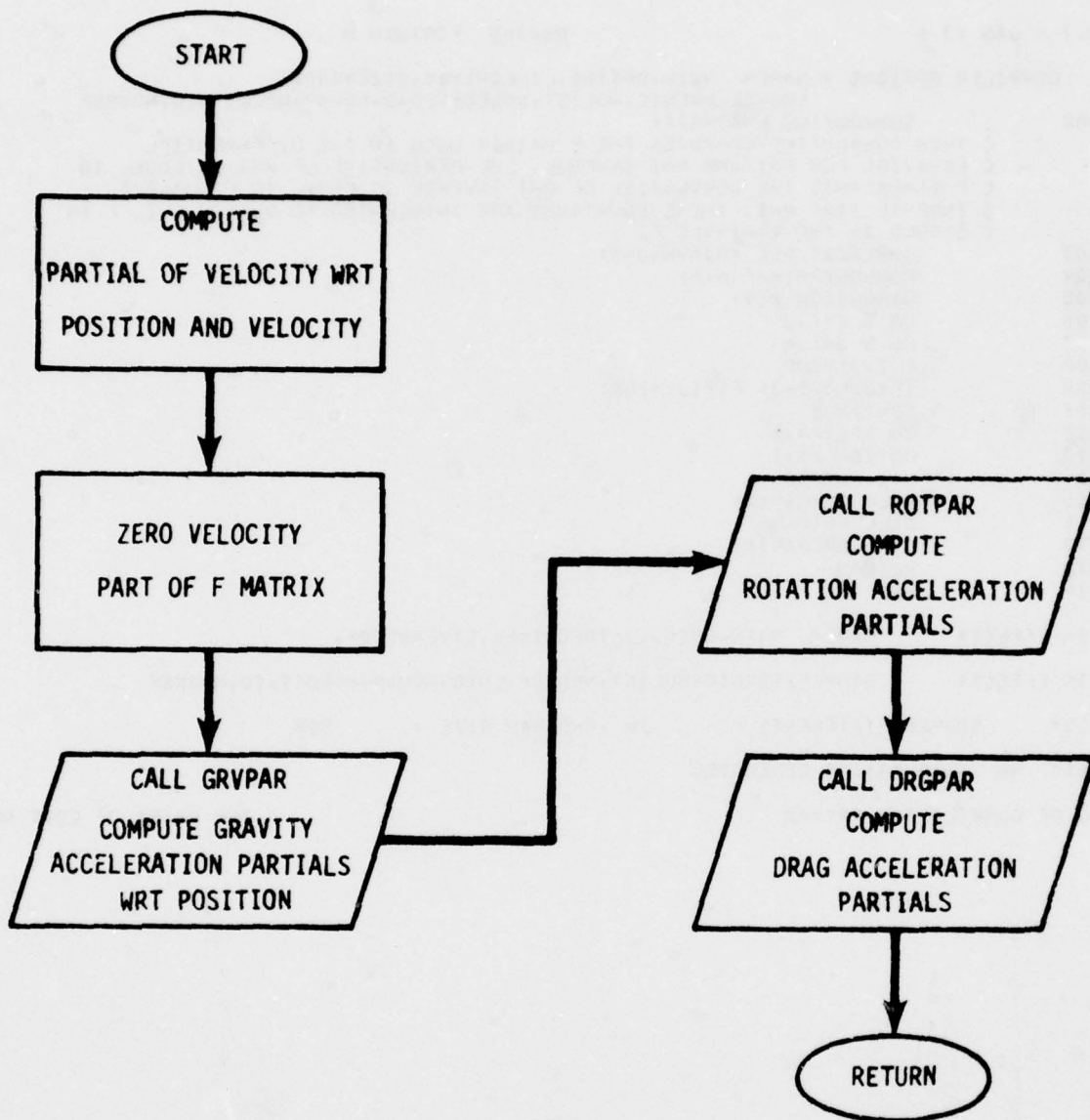
X 9 dimensional array containing position, velocity and acceleration

OUTPUT:

F 6x6 dimensional array containing matrix which relates transition matrix and its inverse to their derivatives,

SUBROUTINES USED: GRVPAR, ROTPAR, DRGPAR

F MATRIX FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN H

DATE

COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,IO,NOXREF

```
ISN 0002      SUBROUTINE FMATRIX(X)
              C THIS SUBROUTINE COMPUTES THE F MATRIX USED IN THE DIFFERENTIAL
              C EQUATION FOR PHI AND PHI INVERSE. THE DERIVATIVE OF PHI IS EQUAL TO
              C F TIMES PHI. THE DERIVATIVE OF PHI INVERSE IS EQUAL TO MINUS PHI
              C INVERSE TIME PHI. THESE EQUATIONS ARE INTEGRATED TO OBTAIN PHI. F IS
              C STORED IN THE VARIABLE F.
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      COMMON/PHIM/F(6,6)
ISN 0005      DIMENSION X(9)
ISN 0006      DO 5 I=1,3
ISN 0007      DO 5 J=1,6
ISN 0008      F(I,J)=000
ISN 0009      IF(J.EQ.1+3) F(I,J)=100
ISN 0011      5 CONTINUE
ISN 0012      DO 10 I=4,6
ISN 0013      DO 10 J=1,6
ISN 0014      10 F(I,J)=000
ISN 0015      CALL GRVPAR(X)
ISN 0016      CALL ROTPAR
ISN 0017      CALL DRGPAR(X)
ISN 0018      RETURN
ISN 0019      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,IO,NOXREF

STATISTICS SOURCE STATEMENTS = 18 ,PROGRAM SIZE = 508

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

79K BYTES OF CORE NOT USED

6.9 DRGPAR SUBROUTINE DESCRIPTION

In this application this subroutine is a dummy subroutine. Drag is not modeled in either the powered flight or free fall segments of the trajectory. However, when the reentry portion of the trajectory is considered, these partials will have to be computed.

BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN H

DATE

```
COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,  
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF  
ISN 0002      SUBROUTINE DRGPAR(X)  
C SUBROUTINE DRGPAR COMPUTES THE PARTIALS OF ACCELERATION DUE TO DRAG  
C WITH RESPECT TO POSITION AND VELOCITY. THESE ARE ADDED INTO THE MATRIX  
C F.  
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)  
ISN 0004      COMMON/PHIM/F(6,6)  
ISN 0005      DIMENSION I(9)  
ISN 0006      S=100  
ISN 0007      RETURN  
ISN 0008      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 7 ,PROGRAM SIZE = 238

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

79K BYTES OF CORE NOT USED

6.10 ROTPAR SUBROUTINE DESCRIPTION

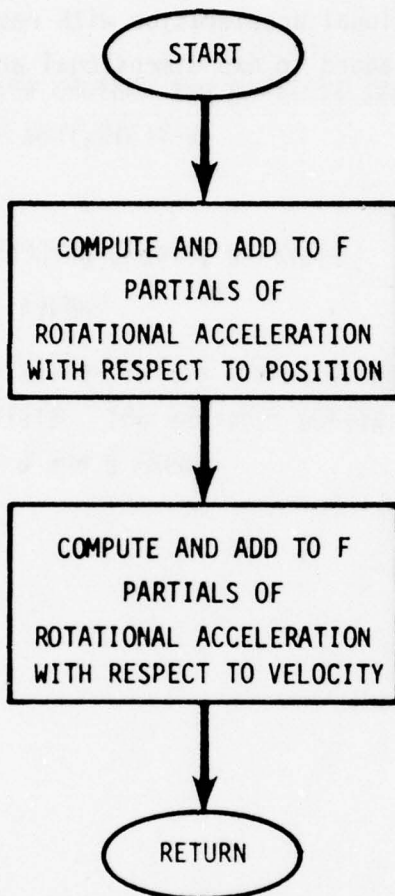
FUNCTION: Computes partials of acceleration due to the rotation of the coordinate system with respect to position and velocity

OUTPUT:

F Partials of rotational acceleration with respect to position and velocity are added to 6x6 dimensional array containing F

SUBROUTINES USED: None

ROTPAR FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN H

DATE

COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

ISN 0002

SUBROUTINE POTPAR

C SUBROUTINE POTPAR COMPUTES THE PARTIALS OF ACCELERATION DUE TO THE
C ROTATION OF THE EARTH WITH RESPECT TO POSITION AND VELOCITY. THESE ARE
C ADDED INTO THE F MATRIX.

ISN 0003

IMPLICIT REAL*8(A-H,O-Z)

ISN 0004

COMMON/PHIM/F(6,6)

ISN 0005

DATA W/0.7291151D-4/

ISN 0006

W2=W*W

ISN 0007

F(4,1)=F(4,1)*W2

ISN 0008

F(5,2)=F(5,2)*W2

ISN 0009

F(4,5)=F(4,5)*W*2D0

ISN 0010

F(5,4)=F(5,4)-W*2D0

ISN 0011

RETURN

ISN 0012

END

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 11 *PROGRAM SIZE = 290

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

79K BYTES OF CORE NOT USED

6.11 GRVPAR SUBROUTINE DESCRIPTION

FUNCTION: Computes partials of gravity acceleration with respect to position

INPUT:

X 9 dimensional state vector. Only position terms are used
/TIG/
IN Flag to indicate whether the partials are to be computed numerically or analytically

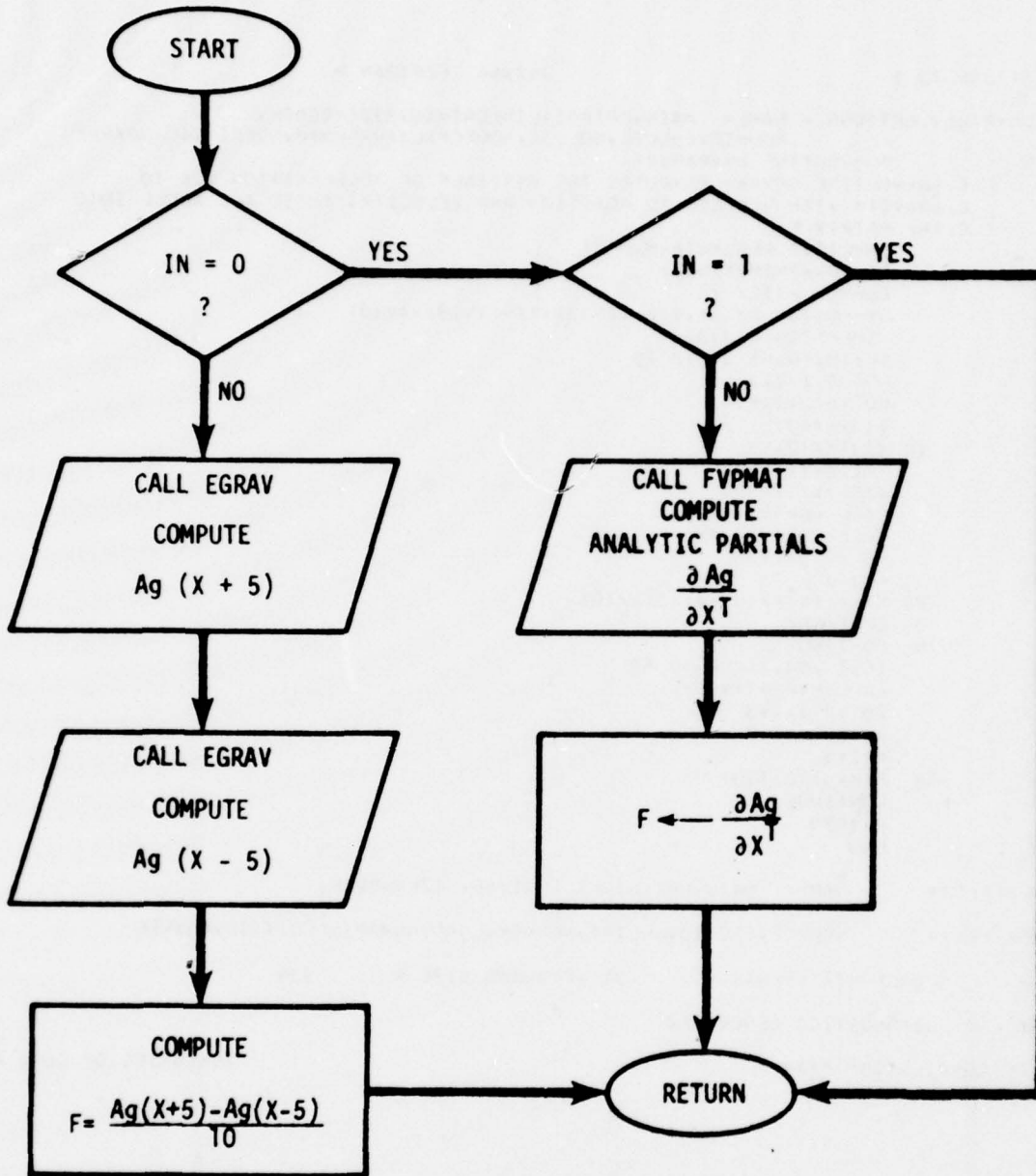
OUTPUT:

F 6x6 array containing gravity partials

SUBROUTINES USED: EGRAV, FVPMAT

COMMENTS: GRVPAR must be called before ROTPAR or DRGPAR. IN = 0 gives numerical partials. The analytic partials and the gravity is based upon the J and D model.

GRV PAR FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN M

DATE

COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBODIC,NOLIST,NOBFCK,LOAD,NOMAP,NOEDIT,NOXREF

```
ISN 0002      SUBROUTINE GRVPAR(X)
C SUBROUTINE GRVPAR COMPUTES THE PARTIALS OF ACCELERATION DUE TO
C GRAVITY WITH RESPECT TO POSITION AND VELOCITY. THESE ARE ADDED INTO
C THE MATRIX F.
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      COMMON/PHIM/F(6,6)
ISN 0005      COMMON /TIC/ IN
ISN 0006      DIMENSION AX(3),Y(6),AY(3),X(9),V(9),AV(3)
ISN 0007      DIMENSION G(3,3)
ISN 0008      IF (IN.EQ.0) GO TO 70
ISN 0010      DO 30 I=1,3
ISN 0011      DO 10 J=1,6
ISN 0012      V(J)=X(J)
ISN 0013      10 Y(J)=X(J)
ISN 0014      Y(I)=Y(I)+500
ISN 0015      V(I)=V(I)-500
ISN 0016      CALL FGRVAV(Y,AY)
ISN 0017      CALL EGRVAV(V,AV)
ISN 0018      DO 20 J=1,3
ISN 0019      K=J+3
ISN 0020      20 F(K,I)=(AY(J)-AV(J))/101
ISN 0021      30 CONTINUE
ISN 0022      70 CONTINUE
ISN 0023      IF (IN.EQ.1) GO TO 60
ISN 0025      CALL FVPMAT(X,G)
ISN 0026      DO 50 I=1,3
ISN 0027      DO 50 J=1,3
ISN 0028      K=I+3
ISN 0029      50 F(K,J)=G(I,J)
ISN 0030      60 CONTINUE
ISN 0031      RETURN
ISN 0032      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBODIC,NOLIST,NOBFCK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 31 *PROGRAM SIZE = 934

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

75K BYTES OF COPE NOT USED

6.12 EGRAV SUBROUTINE DESCRIPTION

FUNCTION: Computes gravity acceleration using J and D model of gravity

INPUT:

Y 9 dimensional array containing state vector. Only first 3 elements are used

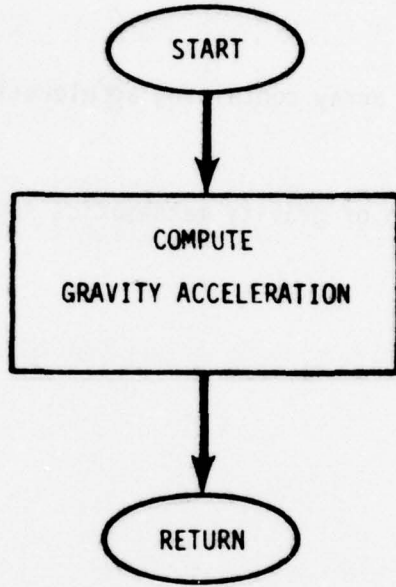
OUTPUT:

GY 3 dimensional array containing acceleration for three variables

SUBROUTINES USED: None

COMMENTS: For descripton of gravity mathematics see [4].

E GRAV FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN H

DATE

COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K.

SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOCEDIT,ID,NOXREF

```
ISN 0002      SUBROUTINE EGRAV(Y,GY)
ISN 0003      IMPLICIT REAL*8 (A-H,O-Z)
ISN 0004      DIMENSION Y(1),GY(1)
ISN 0005      DATA GM, COND,CONA,CJCAS/1.46764685 D16,
              * 0.10279500 D-4, 20925672.6 D0, 0.711155219 D17/
C.....
C COMPUTE THE ACCELERATION IN EFG DUE TO GRAVITY
C* THIS SUBROUTINE IS TAKEN FROM A TRW REPORT ON IUT SIMULATION *
C.....
ISN 0006      A = CONA
ISN 0007      D = COND
ISN 0008      R2 = 0.
ISN 0009      DO 10 I=1,3
ISN 0010      10 R2 = R2 + Y(I)*Y(I)
ISN 0011      R4 = R2*R2
ISN 0012      ZR2 = Y(3)*Y(3)/R2
ISN 0013      A2 = A*A
ISN 0014      DA4 = D*A2*A2
ISN 0015      H = (GM/R2)*(1.00 + (CJCAS/R2)*(1.00 - 5.00*ZR2) + (DA4/R4)*
              * (0.4285714300 - ZR2*(16.00 - 9.00*ZR2)))
ISN 0016      A2 = DSORT(R2)
ISN 0017      GY(3) = -(Y(3)/A2)*(H + (GM/R4)*(2.00*CJCAS + (DA4/R2)*
              * (1.71428571400 - 4.00*ZR2)))
ISN 0018      GY(1) = -(H*Y(1)/A2)
ISN 0019      GY(2) = -(H*Y(2)/A2)
ISN 0020      RETURN
ISN 0021      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K.

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOCEDIT,ID,NOXREF

STATISTICS SOURCE STATEMENTS = 20 *PROGRAM SIZE = 814

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

79K BYTES OF CORE NOT USED

6.13 DIFFUN SUBROUTINE DESCRIPTION

FUNCTION: Computes acceleration due to gravity and acceleration sensed because of rotating coordinate. Computes velocity of body in motion

INPUT:

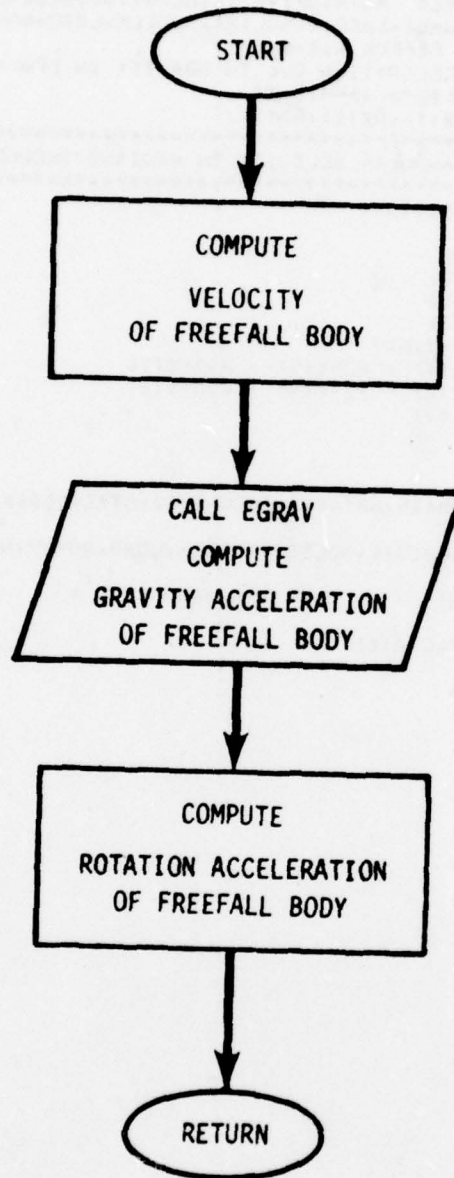
- N Number of differential equations (never used because it is always 6)
- T Time (not used)
- Y 9 dimension array containing position, velocity and acceleration. Only position and velocity are used

OUTPUT:

- DY 6 dimensional array containing gravity and rotation accelerations and velocities.

SUBROUTINES USED: EGRAV

DIFFUN FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN M

DATE

```
COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF
ISN 0002      SUBROUTINE DIFFUN(N,T,Y,DY)
ISN 0003      C COMPUTE THE ACCELERATION DUE TO GRAVITY IN EFG COORDINATES
ISN 0004      IMPLICIT REAL*8 (A-H,O-Z)
              DIMENSION Y(1),DY(1),GY(3)
              C.....
              C* W IS EARTH'S ANGULAR VELOCITY IN RADIANS/SECOND *
              C.....
ISN 0005      W = 0.72921151D-4
ISN 0006      WSO = W*W
ISN 0007      W2 = 2.D0*W
ISN 0008      20 DY(1) = Y(4)
ISN 0009      DY(2) = Y(5)
ISN 0010      DY(3) = Y(6)
ISN 0011      CALL EGRAV(Y,GY)
ISN 0012      DY(4) = GY(1) + W2*Y(5) + WSO*Y(1)
ISN 0013      DY(5) = GY(2) - W2*Y(4) + WSO*Y(2)
ISN 0014      DY(6) = GY(3)
ISN 0015      RETURN
ISN 0016      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,EBCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,NOXREF

STATISTICS SOURCE STATEMENTS = 15 ,PROGRAM SIZE = 496

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

79K BYTES OF CORE NOT USED

6.14 FVPMAT SUBROUTINE DESCRIPTION

FUNCTION: Computes partials of gravity acceleration with respect to position

INPUT:

X 9 dimensional array containing position, velocity and acceleration.
Only position is used

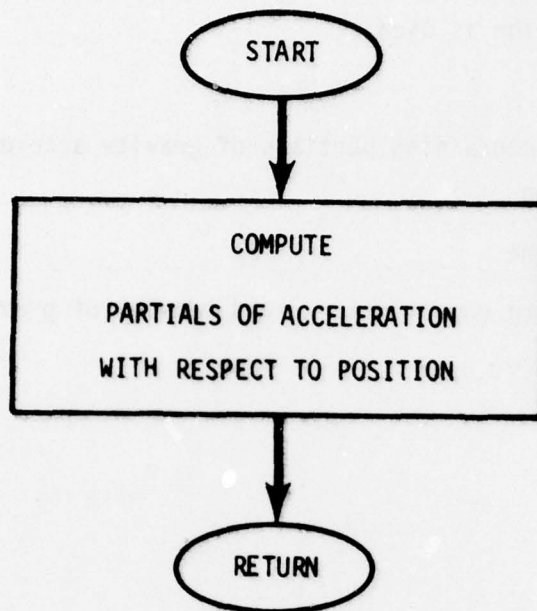
OUTPUT:

F 3x3 array containing partials of gravity acceleration with respect
to position

SUBROUTINES USED: None

COMMENTS: Partial s are computed for J and D model of gravity

FVP MAT FLOWCHART



BEST AVAILABLE COPY

LEVL 21.7 (JAN 73)

05/360 FORTRAN H

DATE

```

COMPILER OPTIONS - NAME = MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE=ERCUDIC,NOLIST,NOUFCK,LOAD,NOMAP,NOEDIT,IO,NOXREF
ISN 0002      SUBROUTINE FVPMAT(X,F)
C.....
C*
C*   COMPUTE THE PARTIALS OF ACCELERATION WITH RESPECT TO POSITION
C*
C*   P1 = P( R, (1./R**2), (1./R**4) ) / P( X, Y, Z)
C*
C*   P2 = P( Z**2/R**2 ) / P( X, Y, Z)
C*
C*   P3 = P( X/R, Y/R, Z/R ) / P( X, Y, Z)
C*
C*   P4 = P( H ) / P( X, Y, Z)
C*
C.....
ISN 0003      IMPLICIT REAL*8 (A-H,O-Z)
ISN 0004      REAL*8 GM,JA2
ISN 0005      DIMENSION X(1),P1(3,3),P2(3),P3(3,3),P4(3),F(3,3)
ISN 0006      DATA COND/ 0.10279500 D-4 /
ISN 0007      DATA CJCAS/ 0.711155219 D12 /
ISN 0008      DATA W/ 0.72411510-4 /
ISN 0009      DATA GM/ 1.40764665 D16 /
ISN 0010      DATA A/ 20925672.6 D0 /
ISN 0011      EQUIVALENCE (CJCAS,JA2),(D,COND)
ISN 0012      P2 = X(1)*X(1) + X(2)*X(2) + X(3)*X(3)
ISN 0013      R = DSORT(R2)
ISN 0014      R3 = R*P2
ISN 0015      R4 = R2*R2
ISN 0016      R5 = R3*R3
ISN 0017      D0 10 I=1,3
ISN 0018      P1(1,I) = X(I)/R
ISN 0019      P1(2,I) = -2.00*X(I)/R4
ISN 0020      10 P1(3,I) = -4.00*X(I)/R6
ISN 0021      Z2 = X(3)*X(3)
ISN 0022      P2(1) = -2.00*X(1)*Z2/P4
ISN 0023      P2(2) = -2.00*X(2)*Z2/R4
ISN 0024      P2(3) = 2.00*X(3)*(R2 - Z2)/R4
ISN 0025      P3(1,1) = (R2 - X(1)*X(1))/R3
ISN 0026      P3(1,2) = -X(1)*X(2)/R3
ISN 0027      P3(1,3) = -X(1)*X(3)/R3
ISN 0028      P3(2,1) = P3(1,2)
ISN 0029      P3(2,2) = (R2 - X(2)*X(2))/R3
ISN 0030      P3(2,3) = -X(2)*X(3)/R3
ISN 0031      P3(3,1) = P3(1,3)
ISN 0032      P3(3,2) = P3(2,3)
ISN 0033      P3(3,3) = (R2 - Z2)/R3
ISN 0034      Z2 = Z2/R2
ISN 0035      DA4 = D*A*A*A*A
ISN 0036      H = (GM/R2)*(1.00*(JA2/R2)*(1.00 - 5.00*Z2) + (DA4/R4)*
1      (0.4285714300 - Z2*(6.00 - 9.00*Z2)))
ISN 0037      D0 20 I=1,3
ISN 0038      P4(I) = (GM/R2)*(JA2/R2)*(-5.00*P2(I) + JA2*(1.00 - 5.00*Z2)
1      *P1(2,I) - (DA4/W4)*(Z2*(-9.00*P2(I) + (6.00 - 9.00*Z2)
2      *P2(I) + DA4*(0.4285714300 - Z2*(6.00 - 9.00*Z2))
3      *P1(3,I)) + H*R2*P1(2,I)
ISN 0039      20 CONTINUE
ISN 0040      GZ = -(X(3)/R)*(H + (GM/R4)*(2.00*JA2 + (DA4/R2)

```

BEST AVAILABLE COPY

```
ISN 0041      1      *(1.71428571400 - 4.00*Z2)))
ISN 0042      DO 30 I=1,3
ISN 0043      F(1,I) = -H*P3(1,I) - (X(1)/R)*P4(I)
ISN 0044      F(2,I) = -H*P3(2,I) - (X(2)/R)*P4(I)
              F(3,I) = -(X(3)/R)*P4(I) + (GM/R4)*((DA4/R2)*(-4.00*P2(I))
              1      + DA4*(1.71428571400 - 4.00*Z2)*P1(2,I))
              2      + GM*(2.00*JA2 + (UA4/R2)*(1.71428571400 - 4.00*Z2))
              3      *P1(3,I) + (R*GZ/X(3))*P3(3,I)
ISN 0045      30 CONTINUE
ISN 0046      #SO = W*W
ISN 0047      RETURN
ISN 0048      END
```

OPTIONS IN EFFECT NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

OPTIONS IN EFFECT SOURCE,ERCDIC,NOLIST,NODECK,LOAD,NOMAP,NOEDIT,ID,NOXREF

STATISTICS SOURCE STATEMENTS = 47 ,PROGRAM SIZE = 1854

STATISTICS NO DIAGNOSTICS GENERATED

***** END OF COMPILATION *****

67K BYTES OF COPE NOT USED

STATISTICS NO DIAGNOSTICS THIS STEP

6.15 CON SUBROUTINE DESCRIPTION

FUNCTION: Computes constants for the Spline interpolating polynomials for the position in the state vector and for the transition matrix and its inverse

INPUT:

/VAR/

H Step size of integration

X0 9 dimensional array containing position and velocity at the beginning of the interval

X 9 dimensional array containing position and velocity at the end of the interval

PHIO 6x6 dimensional array containing transition matrix at the beginning of the interval

PHI 6x6 dimensional array containing transition matrix at the end of the interval

PHIDO 6x6 dimensional array containing derivatives of PHIO

PHID 6x6 array containing derivatives of PHI

PHINO 6x6 array containing the inverse of PHIO

PHINDO 6x6 array containing derivatives of PHINO

PHIN 6x6 array containing the derivatives of PHIN

PHIND 6x6 array containing the derivatives of PHIN

ID FLAG TO COMPUTE COEFFICIENTS FOR SPLINES FOR TRANSITION INVERSE

OUTPUT:

/VAR/

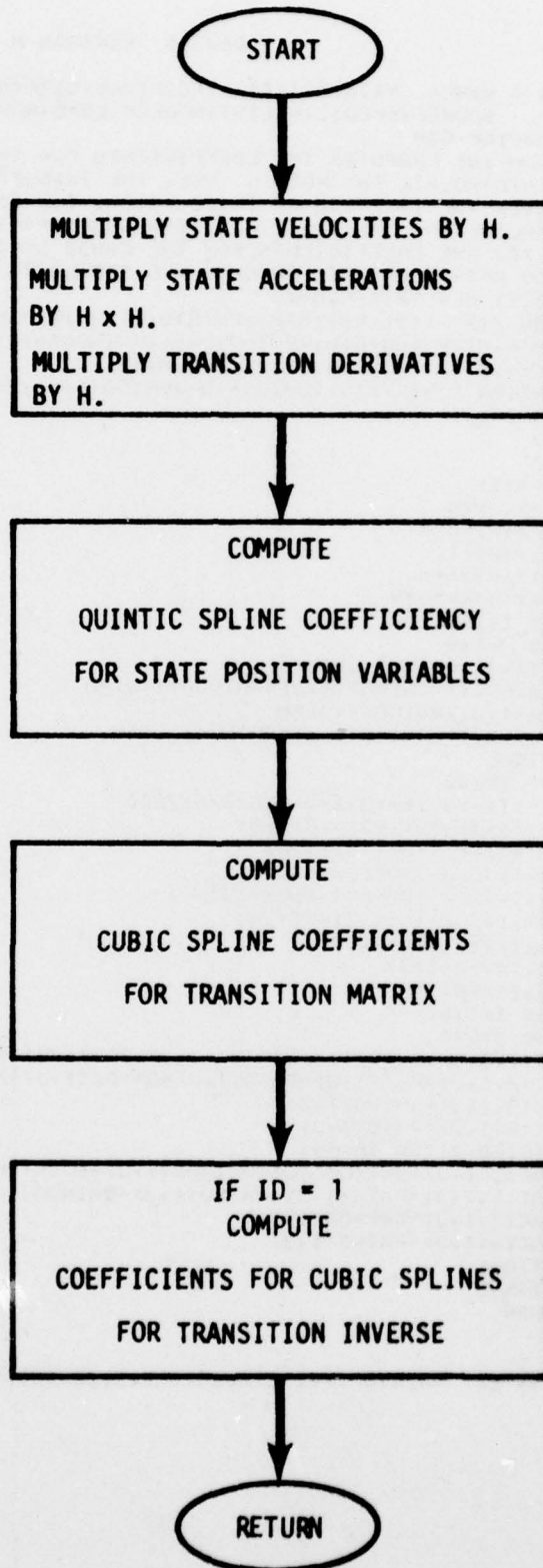
XC 6x3 array containing coefficients for the quintic Splines for position. The first index indicates the polynomial coefficient and the second indicates the variable

PHIC 4x6x6 array containing coefficients for the cubic Splines for the transition matrix. The first index indicates the coefficient and the second and third indices indicates the matrix element.

PHINC 4x6x6 array containing coefficients for the cubic Splines for the
inverse of the Transition matrix. The indices are the same for
RHIC

SUBROUTINES USED: None

CON FLOWCHART



BEST AVAILABLE COPY

LEVEL 21.7 (JAN 73)

OS/360 FORTRAN H

DATE

COMPILER OPTIONS - NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,
SOURCE,ERCDIC,NOLIST,NODECK,LOAD,NOVAP,NOEDIT,IO,NOXREF

```

ISN 0002      SUBROUTINE CON
C THIS SUBROUTINE COMPUTES THE COEFFICIENTS FOR THE SPLINE POLYNOMIALS
C USED TO INTERPOLATE THE POS. & VEL. THE TRANSITION MATRIX AND ITS
C INVERSE OVER THE INTERVAL OF INTEGRATION. THE COEFFICIENTS OF THE
C QUINTIC SPLINE FOR THE THREE POSITION COORDINANTS ARE STORED IN THE
C VARIABLE XC. THE COEFFICIENTS FOR THE CUBIC SPLINES FOR THE
C TRANSITION MATRIX AND ITS INVERSE ARE STORED IN PHIC AND PHINC.
ISN 0003      IMPLICIT REAL*8(A-H,O-Z)
ISN 0004      COMMON /VAR/X(9),PHI(6,6),PHID(6,6),PHIN(6,6),PHIND(6,6),
1 PHIO(6,6),PHIDO(6,6),PHINO(6,6),PHINDO(6,6),XO(9),XC(6,3),
2 PHIC(4,6,6),PHINC(4,6,6),TS,T,H,IO
ISN 0005      DIMENSION Y(9),YO(9),QHID(6,6),QHIDO(6,6),QHIND(6,6),QHINDO(6,6)
ISN 0006      DO 35 I=1,3
ISN 0007        J=I+3
ISN 0008        K=I+6
ISN 0009        Y(I)=X(I)
ISN 0010        Y(J)=X(J)*H
ISN 0011        Y(K)=X(K)*H*H
ISN 0012        YO(I)=XO(I)
ISN 0013        YO(J)=XO(J)*H
ISN 0014      35 YO(K)=XO(K)*H*H
ISN 0015        DO 40 I=1,6
ISN 0016          DO 40 J=1,6
ISN 0017            QHID(I,J)=PHID(I,J)*H
ISN 0018            IF (ID.EQ.1) QHINDO(I,J)=PHINDO(I,J)*H
ISN 0020            QHIDO(I,J)=PHIDO(I,J)*H
ISN 0021            IF (ID.EQ.1) QHIND(I,J)=PHIND(I,J)*H
ISN 0023      40 CONTINUE
ISN 0024        DO 45 I=1,3
ISN 0025          EL1=Y(I)-YO(I)-YO(I+3)-YO(I+6)/2D0
ISN 0026          EL2=Y(I+3)-YO(I+3)-YO(I+6)
ISN 0027          EL3=(Y(I+6)-YO(I+6))/2D0
ISN 0028          XC(1,I)=EL3-3D0*EL2+6D0*EL1
ISN 0029          XC(2,I)=EL2-3D0*EL1-2D0*XC(1,I)
ISN 0030          XC(3,I)=EL1-XC(1,I)-XC(2,I)
ISN 0031          XC(4,I)=YO(I+6)/2D0
ISN 0032          XC(5,I)=YO(I+3)
ISN 0033      45 XC(6,I)=YO(I)
ISN 0034          DO 50 I=1,6
ISN 0035            DO 50 J=1,6
ISN 0036              PHIC(1,I,J)=QHID(1,J)+QHIDO(1,J)-2D0*(PHI(1,J)-PHIO(1,J))
ISN 0037              PHIC(2,I,J)=PHI(1,J)-PHIO(1,J)-QHIDO(1,J)-PHIC(1,I,J)
ISN 0038              PHIC(3,I,J)=QHIDO(1,J)
ISN 0039              PHIC(4,I,J)=PHIO(1,J)
ISN 0040              IF (ID.EQ.0) GO TO 49
ISN 0042              PHINC(1,I,J)=QHIND(1,J)+QHINDO(1,J)-2D0*(PHIN(1,J)-PHINDO(1,J))
ISN 0043              PHINC(2,I,J)=PHIN(1,J)-PHINDO(1,J)-QHINDO(1,J)-PHINC(1,I,J)
ISN 0044              PHINC(3,I,J)=QHINDO(1,J)
ISN 0045              PHINC(4,I,J)=PHINDO(1,J)
ISN 0046      49 CONTINUE
ISN 0047      50 CONTINUE
ISN 0048      RETURN
ISN 0049      END

```

OPTIONS IN EFFECT

NAME= MAIN,OPT=01,LINECNT=60,SIZE=0000K,

REFERENCES

1. Carlson, N.A., "Fast Triangular Formulation of the Square Root Filter", AIAA Journal, Vol. 11, No. 9, September 1973, pp. 1259-1265.
2. Ralston, Anthony, A First Course in Numerical Analysis, p. 200, New York: McGraw-Hill, 1965.
3. Bellman, Richard, Introduction to Matrix Analysis, p. 169, New York: McGraw-Hill, 1960.
4. Miller, A. IDT Simulation, Vehicle Dynamics, Detailed Mathematics and Flowcharts, p. 4-3, TRW Systems Group, 1972.