NASA TM-102,600

NASA-TM-102600 19920022213

NASA Technical Memorandum 102600

Comparison of Effects of Copropagated and Precomputed Atmosphere Profiles on Monte Carlo Trajectory Simulation

Eric M. Queen and
Thomas M. O'Mara

August 1990

~~NOTICE~~

~~FOR EARLY DOMESTIC DISSEMINATION~~

~~Because of its significant early commercial potential, this information, which has been developed under a U.S. Government program, is being disseminated within the United States in advance of general publication. This information may be duplicated and used by the recipient with the express limitation that it not be published. Release of this information to other domestic parties by the recipient shall be made subject to these limitations.~~

~~Foreign release may be made only with prior NASA approval and appropriate export licenses. This legend shall be marked on any reproduction of this information in whole or in part.~~

Review for general release August 31, 1992

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

LIBRARY COPY

AUG 0 7 1990

LANGLEY RESEARCH CENTER
LIBRARY NASA
HAMPTON, VIRGINIA

SUMMARY

A realization of a stochastic atmosphere model for use in simulations is presented. The model provides pressure, density, temperature, and wind velocity as a function of latitude, longitude, and altitude, and is implemented in a three-degree of freedom simulation package. This implementation is used in the Monte Carlo simulation of an aeroassisted orbital transfer maneuver and results are compared to those of a more traditional approach.

ABBREVIATIONS AND SYMBOLS

Roman

g	Gaussian random number
h	altitude (km)
L	correlation scale distance (km)
P(n)	pressure at time step n (N/m^2)
R_ρ	density autocorrelation
T(n)	temperature realized at time step n (K)
$\bar{T}$(n)	mean temperature at time step n (K)
u(n)	eastward wind at step n (m/sec)
v(n)	northward wind at step n (m/sec)
z	distance measure (km)

Greek

ρ(n)	realized density at time step n (kg/m^3)
ρ_L(n)	large scale density perturbation at time step n (per mill)
ρ_S(n)	small scale density perturbation at time step n (per mill)
$\bar{\rho}$(n)	mean density at time step n (kg/m^3)
ρ'	total density perturbation $\rho_L + \rho_S$ (per mill)
σ_ρ	density standard deviation
ϕ	latitude (degrees)

Subscripts

H	horizontal

N92-31457#
~~X90-10471#~~

L	large scale
n	time step n
P	pressure
S	small scale
T	temperature
U	wind speed
ρ	density
K	either L or S, as appropriate

Miscellaneous

$\bar{\ }$	(overbar) mean value
$'$	(prime) total perturbation

INTRODUCTION

In order to perform rigorous simulations of the trajectories of advanced aerospace vehicles, it is desirable to have an accurate model of the Earth's atmosphere. Unfortunately for the engineers designing such vehicles, the atmosphere is much too large and complex to completely determine its state at any given time. Such a system is best modeled as a stochastic process.

This report documents an implementation of the Global Reference Atmosphere Model (GRAM) [1-2] as an option for the Program to Optimize Simulated Trajectories (POST) [3]. A brief description of the random atmosphere model is given in the first section. In the next section, implementation of the model in POST is discussed. The following section describes an application of the model to a Monte Carlo simulation. The final section presents conclusions. An appendix gives the detailed source code.

ATMOSPHERIC SIMULATION

The GRAM model was taken as the baseline for the atmospheric simulation development. The GRAM model describes the Earth's atmosphere as a tabular function of altitude, longitude, latitude, and calendar month, plus a random component which accounts for such effects as gravity waves, tides and turbulence.

One aspect of the model which is important for trajectory simulations is the spatial correlation of the random variations. In the atmosphere, the pressure and density at a given point have values close to that of the pressure and density at a nearby point. Thus, in the model, some correlation function is needed to limit gradients in pressure and density to realistic values.

Atmospheric properties at any point along a trajectory are given as a mean at that location plus random components which are correlated with the previous trajectory point [2]. For density,

$$\rho(n) = \rho_L(n) + \rho_S(n) + \bar{\rho}(n) \tag{1}$$

where $\rho(n)$ is the density at the n^{th} time step, ρ_L and ρ_S are the large and small scale density perturbations, and $\bar{\rho}$ is the mean density. The large and small scale perturbations are so called because of their relative wavelengths. The large scale perturbations are tightly correlated so they vary gradually with spatial displacement, while the small scale perturbations are loosely correlated and vary more rapidly. In (1),

$$\rho_K(n) = \bar{\rho}(n)[R_{\rho K}\sigma_{\rho K}(n)(\rho(n-1)-\bar{\rho}(n-1)/(\sigma_{\rho K}(n-1)\bar{\rho}(n-1))$$

$$+ \sigma_{\rho K}(n)\ g(1-R_{\rho K}{}^2)^{1/2}] \tag{2}$$

where $\sigma_{\rho K}(n)$ is the standard deviation of density at the present point, $\sigma_{\rho K}(n-1)$ is its standard deviation from the previous time step, $\rho(n-1)$ is the realized density from the previous time step, and g is a Gaussian random variable with zero mean and unity standard deviation. Throughout the document, the subscript K can be replaced consistently by L or S for large or small scale, as appropriate,

Note that the σ's for all parameters are stored in tables as a total variance, s, and a fraction of the variance due to large scale effects, f_L, so

$$\sigma_L = \sqrt{f_L\ s} \tag{3}$$

$$\sigma_S = \sqrt{(1 - f_L)\ s} \tag{4}$$

An equation similar to (2) is used to determine $\rho_S(n)$. The large scale density autocorrelation used in (2) is

$$R_{\rho K} = \exp\left[-((\Delta x/L_{H_{\rho K}})^2 + (\Delta z/L_{V_{\rho K}})^2)^{1/2}\right] \tag{5}$$

Here Δx is the horizontal distance in kilometers from the last trajectory point to the present point, Δz is the vertical distance, and $L_{H_{\rho K}}$ and $L_{V_{\rho K}}$ are scale distances determined by altitude and latitude as discussed below. The correlations R_{TL} and R_{TS} are used to compute the temperature variation, and are calculated in a similar manner.

In the calculation of the random temperature

$$T(n) = \bar{T}(n) + T_L(n) + T_S(n) \tag{6}$$

where

$$T_K(n) = \bar{T}(n)[C_K(T(n-1)-\bar{T}(n-1))/\bar{T}(n-1)$$

$$+ D_K(\rho(n)-\bar{\rho}(n))/\bar{\rho}(n) + E_K g] \tag{7}$$

with

$$C_K = \frac{[\sigma_T(n)/\sigma_T(n-1)][(R_{TK} - R_{\rho K}R_{22}R_{11})]}{(1-R_{\rho K}^{\,2}\,R_{11}^{\,2})} \tag{8}$$

$$D_K = [R_{TK}\sigma_T(n)-C_K\sigma_T(n-1)]/(R_{\rho K}R_{11}\sigma_{\rho K}(n)) \tag{9}$$

$$E_K = [\sigma_T^{\,2}(n) - C_K^{\,2}\sigma_T^{\,2}(n-1) - D_K^2\,\sigma_\rho^2\,(n)$$

$$- 2C_K D_K R_{\rho K}R_{11}\sigma_T(n-1)\sigma_{\rho K}(n)]^{1/2} \tag{10}$$

where

$$R_{22} = \frac{[(\sigma_{PK}(n)/\bar{P}(n))^2 - (\sigma_{\rho K}(n)/\bar{\rho}(n))^2 - (\sigma_{TK}(n)/\bar{T}(n))^2]}{2(\sigma_{\rho K}(n)/\bar{\rho}(n))(\sigma_{TK}(n)/\bar{T}(n))} \tag{11}$$

and R_{11} has the same functional form as R_{22}, but is evaluated at step n-1.

The pressure variation is determined from the temperature and density variations by expanding the ideal gas law to first order:

$$P(n) = \bar{P}(n) + P_S(n) + P_L(n) \tag{12}$$

where

$$P_L(n) = \bar{P}(n)[(\rho_L(n)/\bar{\rho}(n)) + (T_L(n)/\bar{T}(n))] \tag{13}$$

and similarly for $P_S(n)$.

The eastward wind component is given by:

$$u = \bar{u} + u_L + u_S \tag{14}$$

with

$$u_K(n) = \bar{u}(n)\left[F_K u_K(n-1) + G_K \rho_K(n) + H_K g\right] \tag{15}$$

where u(n) is the perturbed eastward wind at trajectory point n, $\bar{u}(n)$ is the mean east wind at point n, $u_K(n-1)$ is the east wind perturbation from the previous point, $\rho_K(n)$ is either the large or small scale density perturbation calculated from (2). The coefficients F,G, and H are given by:

$$F_K = \frac{[\sigma_U(n)/\sigma_U(n-1)][(R_{UK} - R_{\rho K}R_{U2\rho 2}R_{U1\rho 1})]}{(1-R^2_{\rho K} \quad R^2_{U1\rho 1})} \tag{16}$$

$$G_K = [R_{UK}\sigma_U(n)-F_K\sigma_U(n-1)]/(R_{\rho K}R_{U1\rho 1}\sigma_{\rho K}(n)) \tag{17}$$

$$H_K = [\sigma_U{}^2(n) - F_K{}^2\sigma_U{}^2(n-1) - G_K{}^2\sigma_\rho{}^2(n)$$

$$- 2F_K G_K R_{\rho K} R_{U1\rho 1}\sigma_U(n-1)\sigma_{\rho K}(n)]^{1/2} \tag{18}$$

where R_{UK} is given by a form analogous to (5) and $R_{U2\rho 2}$ and $R_{U1\rho 1}$ are taken from tables.

The northward wind component is given by:

$$v = \bar{v} + v_L + v_S \tag{19}$$

with

$$v_K(n) = \bar{v}(n)\left[I_K v_K(n-1) + J_K u_K(n) + K_K g\right] \tag{20}$$

where v(n) is the perturbed eastward wind at trajectory point n, $\bar{v}(n)$ is the mean east wind at point n, $v_K(n-1)$ is the east wind perturbation from the previous point, $\rho_K(n)$ is either the large or small scale density perturbation calculated from (2). The coefficients I, J, and K are given by:

$$I_K = \frac{[\sigma_V(n)/\sigma_V(n-1)][(R_{VK} - R_{\rho K}R_{V2\rho 2}R_{V1\rho 1})]}{(1-R^2_{\rho K} \quad R^2_{V1\rho 1})} \tag{21}$$

$$J_K = [R_{VK}\sigma_V(n) - I_K\sigma_V(n-1)]/(R_{\rho K}R_{V1\rho 1}\sigma_{\rho K}(n)) \tag{22}$$

$$K_K = [\sigma_V{}^2(n) - I_K{}^2\sigma_V{}^2(n-1) - J_K{}^2\sigma_\rho{}^2(n)$$

$$- 2I_K J_K R_{\rho K} R_{V1\rho 1}\sigma_U(n-1)\sigma_{\rho K}(n)]^{1/2} \tag{23}$$

where $R_{VK} = R_{UK}$, and $R_{V2\rho 2}$ and $R_{V1\rho 1}$ are taken from tables.

In equation (5), for $R_{\rho L}$, there is one scale distance for vertical displacements, L_V, and another for horizontal displacements, L_H. The small scale correlation, $R_{\rho S}$, is defined using another pair of scale distances. Similarly, there are vertical and horizontal scale distances for each of the temperature correlations. The large scale, horizontal scale distance, L_{H_L}, is a linear function of altitude:

$$L_{H_L} = 900 + 6z \tag{24}$$

where z is the altitude and both z and L_{H_L} are in kilometers. The small scale, horizontal scale distance is given by

$$L_{H_S} = 20 + 0.0125z^2 \tag{25}$$

The vertical scale distances are functions of latitude as well as of altitude, specifically

$$L_V = [a + b(90 - \phi)^2][0.22 + 0.00258z^{1.5}] \tag{26}$$

where ϕ is the latitude in degrees, and a and b are empirical coefficients. The coefficients a and b are different for large and for small scale and also different for ρ and T. Thus there are different $L_{V_{\rho L}}$, $L_{V_{\rho S}}$, $L_{V_{TL}}$, $L_{V_{TS}}$. The horizontal scale distances are the same for density and temperature; $L_{H_{\rho S}} = L_{H_{TS}}$ and $L_{H_{\rho L}} = L_{H_{TL}}$. The coefficients a and b are given in Table 1.

Note that in (2) and (7) (and similarly for the small scale temperature and density perturbations) there is a gaussian random term which is added directly to the perturbation. Since gaussian random variables can take on arbitrarily large values, it is clearly possible that the perturbations can become unbounded. This could force the atmosphere attributes to become negative, which is physically unrealizable. In the GRAM code this modelling error is compensated by placing the following lower bounds on density, pressure and temperature perturbations:

$$\begin{aligned} \rho' &= -0.9\,\bar{\rho} \\ P' &= -0.9\,\bar{P} \\ T' &= -0.9\,\bar{T} \end{aligned} \tag{27}$$

where,

$$\begin{aligned} \rho' &= \rho_L + \rho_S \\ P' &= P_L + P_S \\ T' &= T_L + T_S \end{aligned} \tag{28}$$

Thus, the density, temperature, and pressure are never allowed to fall below 1/10 of their respective mean values. Unfortunately, this does not guarantee that pressure, density, and temperature will satisfy the perfect gas law. This problem is resolved in the current work by recalculating the perturbations whenever one is realized below 1/10 of its mean. Only the density and temperature are checked since the pressure is obtained by the perfect gas law from temperature and density. Either of these approaches will slightly skew the statistics by raising the mean and reducing the standard deviation.

In the GRAM program as implemented, there can be discontinuities in the winds between 90 and 115 km altitude. In this range, the GRAM model fairs between the Groves section of the model, which is taken from tables, and the Jacchia section, which is analytic. Shown in figure 1 are the magnitudes of north wind at three altitudes, over a range of latitudes and longitudes. These are mean wind magnitudes taken from a series of GRAM runs at constant altitude and latitude and varying longitude from 0 to 360°. At 95 km the wind velocity exhibits step changes with small changes in longitude. Just below this faired region, at 89 km, and above this region, at 116 km altitude, the winds vary continuously over the entire area. The winds at 95 km are representative of the winds at other altitudes in this region. Figure 2 shows that the same phenomenon occurs in the eastward wind velocities. Because of this discontinuity and the fact that the latest modification to GRAM deletes winds above 125 km, winds are not used above 90km in the present implementation.

IMPLEMENTATION OF SIMULATION IN POST

This section describes the software implementation of the preceding model. The three main points to be considered are: how the theory is implemented into the existing POST program, the code changes necessary, and the results of these changes.

The atmosphere model does not employ the standard POST atmosphere setup. A Runge-Kutta integrator must evaluate the state at several times in each integration step, effectively taking a finite difference derivative across each integration step. Since the POST atmospheres are easily differentiable, this integration scheme is entirely appropriate. An atmosphere model which includes Gaussian random terms is not differentiable except in a least squares sense. Because of this, use of an atmosphere model which varies randomly within an integration step will lead to unpredictable results and possibly divergence of the vehicle simulation.

Note that for a stochastic system, trajectory optimization is not defined in the usual sense. None of the optimization options in POST will be meaningful while using the random atmosphere.

POST is a very large and complicated program. Therefore, it is desirable to make as few changes as possible to the core code. Adding a random atmosphere function in the existing framework would involve rerouting of the flow of control in several places.

In this implementation, the random atmosphere is called through POST's closed-loop guidance routine, CLGM. This insures that the atmosphere is updated at every time step. Since the CLGM routine is only called once per time step, the atmospheric terms remain constant across each integration interval, assuring well-behaved state integration. Thus implemented, the random atmosphere requires only one change to POST's flow of control.

The required changes are of four types: addition of new variables, addition of new tables, accessing new subroutines, and required settings to run the random atmosphere. The POST manual, [5, pp. 2-19 through 2-22] describes the addition of new tables and variables. The specific additions are discussed in appendix A.

All of the new variables added were placed in a new common block, called RAND. The variables in the common block RAND are XOLD(4) and AROLD(12), where XOLD contains the state information at the last updated trajectory point, latitude, longitude, altitude, and dot product of downrange and crossrange. AROLD contains the mean and standard deviations of air density, pressure, and temperature for both large and small scale and the realizations of random pressure, density, and temperature at the last trajectory point. AROLD also contains the most recent realized perturbations in density (ρ_L, ρ_S) and temperature (T_L, T_S).

Six new tables were added to the program. One each for large scale standard deviations and small scale standard deviations of pressure, density, and temperature. These tables may be functions of one, two, or three variables. The standard deviations used are from the GRAM-86 model [2]. If other tables of standard deviations were available, they could be substituted in the POST NAMELIST. The GRAM data give each as a function of altitude and latitude.

A mean atmosphere is supplied as a set of three tables [4, pp. 6.c-1 through 6.d-9]. This table need not have a regular grid; that is, it can have a larger number of data points in the most critical areas in the atmosphere and have relatively few data points in the less critical areas. The atmosphere table can be one, two, or three dimensional.

There are a few settings for the POST NAMELIST that need to be made for the random atmosphere case.

1. NPC(5)=0: This is the "no atmosphere" option. An atmosphere table is still input by the user and read as if NPC(5)=1 (the user supplied atmosphere) but the flag should be set to 0 to prevent the main POST atmosphere routine from overwriting the random values. [4, pg. 6.a.4-1]
2. NPC(12)≠0: This insures that the downrange and crossrange will be calculated, therefore the distance traveled in each time step is known and the atmosphere correlation is calculated correctly. Typically, NPC(12)=2. [4, pg. 6.a.19-1]
3. MDL=8: This directs the integration to continue until the event criterion is met or exceeded. Thus the program will not iterate to find the exact time a criterion was met, but will stop at the next integer time step. [4, pp. 6.a.6-1 through 6-4]
4. IGUID(14)=2: This invokes the closed loop guidance routine which contains the call to the random atmosphere routine. [4, pp. 6.b.7-1 and 7-2]
5. NPC(2)=1: Uses the fourth order Runge-Kutta integration. [4, pg. 6.a.15-3]

Three subroutines were added to the code. RNDATM is accessed through the existing routine CLGM as discussed previously. The others, RNDATI and XKR, are called by RNDATM. RNDATM is the random atmosphere generator. RNDATI is an initialization routine called only once from RNDATM. RNDATI gives values for the "previous" random point needed to start the recursive algorithm. The Gaussian random number generating subroutine, XKR, takes approximately two realizations from a uniformly distributed sequence and transforms them into a single realization from normal distribution [6].

On the first pass, RNDATM calls RNDATI and returns. RNDATI establishes initial values needed for the iterative formula used in

RNDATM. On successive calls, RNDATM computes pressure, density, temperature and speed of sound as described above.

The result of the RNDATM subroutine is an atmosphere profile which varies from the mean atmosphere in a random fashion according to the GRAM statistics.

RESULTS

The traditional method for doing Monte Carlo simulations through a random atmosphere involves precomputing a series of random atmosphere altitude profiles and using these profiles in the vehicle simulation. This procedure is illustrated in Figure 3. First the vehicle simulation is run using a deterministic atmosphere model. This produces a 'nominal' trajectory which is edited to be monotonic in altitude and to insure the data points are properly spaced. This nominal trajectory is input into GRAM repeatedly to produce a series of atmosphere profiles which are dependent on altitude. These profiles are then reformatted to be used in the vehicle simulation. Reference 10 provides an excellent application of this approach.

The atmosphere model described in previous sections of this report propagates an atmosphere state which depends continuously on the vehicle state and a vehicle state which is a function of the atmosphere state. Use of this model in Monte Carlo simulations will be referred to as the copropagated approach. Figure 4 shows the general flow of the copropagated approach. The POST plant model determines the vehicle state as a function of time while the atmosphere model determines the atmosphere state as a function of location, r. In the figure, $1/z_t$ represents the backward time-shift operator, and $1/z_r$ represents the backward spatial-shift operator. The vehicle plant and the atmosphere model are connected by determining the spatial trajectory step, Δr, as a function of the time step. The atmosphere model is also driven by a white noise term, w(r).

To demonstrate the copropagated atmosphere trajectory implementation, an aeroassisted orbital transfer maneuver is simulated. The purpose of such a maneuver is to utilize aerodynamic forces and the energy losses associated with them to change a vehicle's orbit, either in radii, inclination, or both. The guidance law incorporated here was developed for the Aeroassisted Flight Experiment (AFE) and treats the trajectory as one made up of two phases [7]. The first, or entry phase is considered to be an equilibrium glide in which the rate of altitude change, dh/dt, is held constant. The second, exit phase, begins near the lowest altitude of the trajectory and is controlled by a predictor-corrector algorithm which continually updates control variable values to ensure a desired exit velocity and flight path angle.

Statistical information was obtained by performing Monte Carlo analysis of randomly varying atmospheres encountered during 100 simulated AFE flights. The results that follow include both trajectory and point statistics. Point statistics at atmosphere exit are compiled to illustrate the accumulated effects of the random atmosphere variations propagating throughout each flight (Table 2). Trajectory statistics are the result of averaging atmospheric variables (pressure, density, temperature) at each time step over the 100 flights to yield an "average" trajectory which is truncated to the minimum sample trajectory time duration. For presentation, these data are normalized about variable values corresponding to identical AFE trajectories simulated in a '76 Standard Atmosphere without using the copropagated Atmosphere/Trajectory capability.

Figure 5 shows altitude histories for both the average copropagated Atmosphere/Trajectory case and the same AFE profile employing a '76 Standard atmosphere. The shift in altitude developing in the latter portion of the flight can be attributed to a bias in air densities found between the '76 and GRAM models effectively allowing the copropagated runs to generate greater lift in a more dense atmosphere. Plotted as a function of velocity, we see in figure 6 the same average trajectory and '76 Standard trajectory. This is a good illustration of the different spacing of data points during both the entry and exit portions of the flights. We also notice here the near constant velocity entrance and exit bracketing a region of large velocity gradients at lower altitudes.

Normalized values of pressure, density and temperature using the copropagated atmosphere are plotted as functions of altitude in figures 7, 8, and 9. These atmospheric variables display smoother structure of mean value fluctuations during entry (122 km altitude) than on exit. This can be attributed to the wider spacing of data points in this region caused by the high entrance velocity. Any shift in means at a given altitude between the entrance and exit phases is partially a function of the latitude dependence of the GRAM atmosphere model, as the sample flights involved traversing a range of both latitudes and longitudes.

Variances of normalized atmospheric variables are plotted in figures 10, 11, and 12. On all plots variances start at zero, consistent with the output of GRAM which assigns only mean values to initial trajectory locations. When plotted against altitude, variances display a pattern indicating that towards the end of the AFE flight variance magnitude and fluctuation increase. Trends in variance altitude histories display the effects of spatial separation. It is clear that when large distances separate consecutive trajectory points the GRAM perturbation model yields a smooth transition from one point to the next.

A comparison of point statistics pertaining to Monte Carlo analysis is presented in Table 2. The copropagated atmosphere and trajectory determination method is compared to the precomputed atmosphere profile approach. The mean trajectories for the two approaches were nearly identical. The final statistics of the two approaches (energy, latitude, and longitude) are in agreement, but there is a discrepancy in maximum dynamic pressure and heating rate and their corresponding variances. This is because at the near-constant altitude portion of the trajectory, where maximum dynamic pressure and heating rate are expected to occur, much greater density variance is modeled when using the copropagated capability. When determining trajectory and atmospheric variables simultaneously, variances are calculated based on the magnitude of spatial separation. The precomputed approach bases variances solely on vertical spatial separation which, in the portion of the trajectory being discussed here, is very small. This difference in the two techniques is made apparent in figures 13 and 14. Density variance is reduced nearly to zero at the midpoint of the trajectory when using the precomputed method, while the trend in copropagated variances appears to be much more realistic. The effects of this difference can be seen carrying over into the resulting velocity variances derived from the two approaches. These figures represent data generated in a previous though similar experiment using 80 simulated trajectories in the Monte Carlo analysis [9].

CONCLUSIONS

An add-on software module for POST which simulates spacecraft trajectories through a stochastic atmosphere based on the GRAM atmosphere model has been developed and demonstrated. This package gives realistic random fluctuations in density, pressure, and temperature while preserving the perfect gas relation.

The software was written in such a way as to require only minor changes to the host program, POST. A complete listing of the changes to POST as well as the listings of the subroutines added are included in the appendices.

This software package was exercised in simulating an aeroassisted orbit transfer problem and the results were compared to a simulation in which precomputed random atmosphere profiles were used. A comparison of results demonstrated significant differences in the statistics of key trajectory parameters between the two simulation approaches. This strongly suggests that the copropagated approach is to be preferred for Monte Carlo simulation, since it is capable of representing atmosphere variation in horizontal flight which the precomputed approach failed to exhibit.

REFERENCES

1. Justus,C.G.; Woodrum, A.W.; Roper, R.G.; and Smith, O.E.: 4-D Global Reference Atmosphere Technical Description, NASA TM X-64871, Sept.1984.

2. Justus, C.G.; Fletcher, G.R.; Gramling, F.E.; and Pace, W.B.: The NASA/MSFC Global Reference Atmospheric Model - MOD 3 (With Spherical Harmonic Wind Model), NASA CR-3256,Georgia Institute of Technology, Atlanta, Georgia, 30332, 1980.

3. POST Formulation Manual vol.1., Martin Marietta Corporation, P.O. Box 179, Denver Colorado, G.L. Brauer, Program Manager, March 1977, NASA CR-132741.

4. POST Utilization Manual vol.2., Martin Marietta Corporation, G.L. Brauer, Program Manager, March 1977, NASA CR-132742.

5. POST Programmer's Manual vol.3., Martin Marietta Corporation, G.L. Brauer, Program Manager, March 1977, NASA CR-132743.

6. Kinderman, A.J.; and Ramage, J.G.;Computer Generation of Normal Random Variables, Journal of the American Statistical Association; Volume 71, Number 356,Theory and methods section; December 1976.

7. Cerimele, C.J.; and Gamble, J.D.:A Simplified Guidance Algorithm for Lifting Aeroassist Orbital Transfer Vehicles,AIAA paper 85-0348, presented at the 23rd. Aerospace Sciences Meeting; Reno, Nevada; Jan. 14-17,1985;.

8. Snook, T.E.:Monte Carlo Evaluation of Aerobraking Guidance Algorithms, JSC-22432, Mission Planning and Analysis Division, Lyndon B. Johnson Space Center, Houston, Texas, January, 1987.

9. Queen, E.M.; O'Mara, T.M.;and Moerder,D.D.:Causal Atmosphere Modeling for Aerospace Vehicle Trajectory Simulation. Presented at the Sixth National Aerospace Plane Symposium; Monterey, Ca.; April 26, 1989.

APPENDIX A: Addition of Common Blocks and Tables to POST

This appendix describes the changes made to the POST code to allow the use of the random atmosphere simulation. The procedure followed is as described in [5] except as required for the VAX operating system.

The common block RAND was added as follows:

1. The common block was referenced in subroutine DATA in file EXEC.FOR. In DATA a data statement was added that gave a default value for each variable in the common block.

```
DATA XOLD/4*0./
DATA AROLD/16*0./
```

2. In subroutine DICT, in file EXEC.FOR, each variable was assigned a Hollerith value. In subroutine DICT the name of the common block is different from that in DATA. In DICT the common was called RANDD but the variables were the same and in the same order. When the common was referenced in the execution of the code, it was referenced as RAND.

POST has a general table facility which allows the addition of new tables without adding dimensioned arrays, hard-wired table arguments, table types, table dimensions, etc. All of the user input tables are placed in an array in blank common. Each table is referenced by a pointer. This pointer directs a general table lookup routine (GENTAB) to the proper location in the blank common. Each table also has a numeric and a mnemonic multiplier associated with it. Thus, to add a new table one needs only add a pointer and two multipliers to the comdeck MOTBL.

1. A new common block, MOTBLO, was added to comdeck MOTBL. It contains the pointers for six tables: SDPRST, for small scale pressure standard deviation, SDPRLT, for large scale pressure standard deviation, SDRHST, SDRHLT, SDTMST, and SDTMLT, for density and temperature similarly. It also contains the multipliers for each of these tables: SDPRSM(2), etc.

2. An exact duplicate of this common, called MOTBLDO, was added to comdeck MOTBLD in file MOTBLD.INC.

3. Data statements were added to subroutine DICT (which referenced common MOTBLDO) to give Hollerith values to the pointers and multipliers as:

```
DATA SDPRST,SDPRSM/6HSDPRST,6HSDPRSM,5H=>ONE/
```

etc.

4. In subroutine RTBLML (file RTBLML.NML) the table multipliers were added to the namelist. The data statement for NAMTBL was extended to include the six multiplier names:

SDPRSM, SDPRLM, SDRHSM, SDRHLM, SDTMSM, SDTMLM.

The data statement for ATRTBL was extended to make room for the new tables.

APPENDIX B: Source Listings

The following is a source listing of the three subroutines added to POST for the random atmosphere. The first, RNDATM is called from the POST routine CLGM. RNDATM in turn calls the other two routines. The INCLUDE statements all access files used in the ordinary POST compilation, except for RAND.INC which is described in Appendix A. The scale height functions were read from the GRAM code from subroutine ERTRB.

```
      SUBROUTINE RNDATM
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER*4 IX
      REAL*8 LVLRHO,LVSRHO,LVLTMP,LVSTMP,LVLWND,LVSWND
      REAL*8 LHL,LHS
C
C** RNDATM ***
C
C
C       NOTE: THE FORMULAS USED IN THIS SUBROUTINE ARE FOUND
C       IN THE GRAM-86 REPORT BY JUSTUS, ET AL.
C
C       MAKE SURE NPC(12)<>0
C
C
      INCLUDE'POST3D_INCLUDE: MOTBL.INC/LIST'
      INCLUDE'POST3D_INCLUDE: NPC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: GCRADC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: MOTIC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: MOTVC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: SERVC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: RAND.INC/LIST'
      INCLUDE'POST3D_INCLUDE: GUIDJC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: SPECAL.INC/LIST'
      INCLUDE 'POST3D_INCLUDE: AUXVC.INC/LIST'
      INCLUDE 'POST3D_INCLUDE: ALTITO.INC/LIST'
      INCLUDE'POST3D_INCLUDE: TIME.INC/LIST'
      INCLUDE'POST3D_INCLUDE: VXIVC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: VELOVC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: FUDGE.INC/LIST'
C
C***EMQ***
C TRIAL COMMON
      COMMON/TRIAL/WORK(50)
C
C
      DIMENSION A(3),B(3),C(3)
C
      EQUIVALENCE(XOLD(1),OLAT)
      EQUIVALENCE(XOLD(2),OLONG)
      EQUIVALENCE(XOLD(4),OLDZ)
C
      EQUIVALENCE(AROLD(1),OLDPRS)
      EQUIVALENCE(AROLD(2),OLDRHO)
      EQUIVALENCE(AROLD(3),OLDTMP)
      EQUIVALENCE(AROLD(4),OSPRSS)
      EQUIVALENCE(AROLD(5),OSRHOS)
      EQUIVALENCE(AROLD(6),OSTMPS)
      EQUIVALENCE(AROLD(7),OGMPRS)
      EQUIVALENCE(AROLD(8),OGMRHO)
      EQUIVALENCE(AROLD(9),OGMTMP)
      EQUIVALENCE(AROLD(10),OSPRSL)
      EQUIVALENCE(AROLD(11),OSRHOL)
      EQUIVALENCE(AROLD(12),OSTMPL)
      EQUIVALENCE(AROLD(13),ORHOPL)
      EQUIVALENCE(AROLD(14),OTMPPL)
      EQUIVALENCE(AROLD(15),ORHOPS)
      EQUIVALENCE(AROLD(16),OTMPPS)
      EQUIVALENCE(AROLD(17),OSUL)
      EQUIVALENCE(AROLD(18),OSUS)
```

```
      EQUIVALENCE(AROLD(19),OSVL)
      EQUIVALENCE(AROLD(20),OSVS)
      EQUIVALENCE(AROLD(21),OCURHL)
      EQUIVALENCE(AROLD(22),OCURHS)
      EQUIVALENCE(AROLD(23),OCVRHL)
      EQUIVALENCE(AROLD(24),OCVRHS)
      EQUIVALENCE(AROLD(25),OUPL)
      EQUIVALENCE(AROLD(26),OUPS)
      EQUIVALENCE(AROLD(27),OVPL)
      EQUIVALENCE(AROLD(28),OVPS)
C
      IF(NPC(5).NE.0.AND.IGF(1).EQ.0)RETURN
      if(npc(5).ne.0)goto 200
C
      IF(IGF(1).EQ.0)THEN
              CALL RNDATI
              ISEED=-INT(100.0*SECNDS(0.0))
              ISEED=INT(1.0D8*RAN0(ISEED))
              iseed=NINT(SPECI(8))
              rbar=0.0
              ictr=0
              RETURN
      ENDIF
C JATM DETERMINES WHICH ATMOSPHERE TO USE AS THE MEAN
      JATM=1
      IF(JATM.GE.2) GOTO 20
C MEAN ATMOSPHERE INPUT IN TABLES
      GMPRS=EXP(GENTAB(PREST))
      GMTMP=GENTAB(ATEMT)
      GMRHO=EXP(GENTAB(DENST))
      PRES=GMPRS
      ATEM=GMTMP
      DENS=GMRHO
C MEAN WINDS FROM TABLES
      GMU=GENTAB(VWUT)
      GMV=GENTAB(VWVT)
      VW(1)=GMU
      VW(2)=GMV
      VW(3)=GENTAB(VWWT)
C     VW(1)=0.0
C     VW(2)=0.0
C     VW(3)=0.0
      CS = DSQRT(ATMOSK(1)*GMPRS)
      GOTO 100
   20 IF(JATM.GE.3) GOTO 30
C  1962 STANDARD ATMOSPHERE AS MEAN
      CALL ATMOS2
      GOTO 100
   30 IF(JATM.GE.4) GOTO 40
C 1963 PATRICK AFB ATMOSPHERE AS MEAN
      CALL ATMOS3
      GOTO 100
   40 IF(JATM.GE.5) GOTO 50
C 1971 VANDENBURG AS MEAN
      CALL ATMOS4
      GOTO 100
   50 CONTINUE
C  1976 STANDARD ATMOSPHERE
      CALL ATMOS5
  100 CONTINUE
```

```
      IF(SPECI(9).LT.0.0)GOTO 200
C*********************************************************************
C     *                                                              *
C     *                                                              *
C*********************************************************************
      IF(GMPRS.LT.0.0)GMPRS=0.0
C
C   GET NEW STD.DEV. FROM TABLE
C
      SDTMPL=DSQRT(GENTAB(SDTMLT))*GENTAB(SDTMST)
      SDPRSL=DSQRT(GENTAB(SDPRLT))*GENTAB(SDPRST)
      SDRHOL=DSQRT(GENTAB(SDRHLT))*GENTAB(SDRHST)
C
      SDTMPS=DSQRT(FP1-GENTAB(SDTMLT))*GENTAB(SDTMST)
      SDPRSS=DSQRT(FP1-GENTAB(SDPRLT))*GENTAB(SDPRST)
      SDRHOS=DSQRT(FP1-GENTAB(SDRHLT))*GENTAB(SDRHST)
C
C  TYPE I CORRELATION
C     SDUL=DSQRT(GENTAB(SDULT))*GENTAB(SDUST)
C     SDVL=DSQRT(GENTAB(SDVLT))*GENTAB(SDVST)
C     SDUS=DSQRT(FP1-GENTAB(SDULT))*GENTAB(SDUST)
C     SDVS=DSQRT(FP1-GENTAB(SDVLT))*GENTAB(SDVST)
C
C TYPE II CORRELATION
      SDUL=DSQRT(GENTAB(SDULT)*DABS(GENTAB(SDUST)))
      SDVL=DSQRT(GENTAB(SDVLT)*DABS(GENTAB(SDVST)))
      SDUS=DSQRT(FP1-GENTAB(SDULT)*DABS(GENTAB(SDUST)))
      SDVS=DSQRT(FP1-GENTAB(SDVLT)*DABS(GENTAB(SDVST)))
C
C   GET DENSITY-VELOCITY CORRELATIONS FROM TABLE
C
      CURHOL=GENTAB(CURHLT)
      CURHOS=GENTAB(CURHST)
      CVRHOL=GENTAB(CVRHLT)
      CVRHOS=GENTAB(CVRHST)
C
C*********************************************************************
C     *                                                              *
C     *                                                              *
C*********************************************************************
C
      IF(GMRHO.LT.0.0)GMRHO=0.0
C
C*** CALCULATE NEW RANDOM DENSITY
C
C  CONVERT FEET TO KILOMETERS
      FTKM=12.0/39370.0
C  CONVERT NAUTICAL MILES TO KILOMETERS
      XNMIKM=6076.115*FTKM
C  CALCULATE HORIZONTAL DISTANCE TRAVELED USING POSITION VECTOR
C     DO 3 KJJ=1,3
C       A(KJJ)=XOLD(KJJ)
C   3   B(KJJ)=XI(KJJ)
C     CALL VCROSS(A,B,C)
C     CALL VUNIT(A,A,XMAG)
C     CALL VUNIT(B,B,XMAG)
C     CALL VUNIT(C,C,XMAG)
C IN ORDER TO AGREE WITH GRAM WE CALCULATE DISTANCE TRAVELED
C USING THE FLAT-EARTH THEORY
C
```

```
C   PER = POLAR EARTH RADIUS IN KM
C   EER = EQUATORIAL EARTH RADIUS IN KM
C   EPS = ECCENTRICITY
        PER=6356.7747
        EER=6378.160
        EPS=(FP1-PER*PER/(EER*EER))
        RADEG=3.1415927D0/180.0D0
        DLON=(LONG-OLONG)*RADEG
        RGDLAT=GDLAT*RADEG
        DLAT=(GDLAT-OLAT)*RADEG
        DELX=GCRAD*DSQRT(DLAT**2+(COS(RGDLAT)*DLON)**2)
        IF(IOFLAG.NE.3)DELX=DELX*FTKM
        DELZ=DABS(OLDZ-ALTITO)
        IF(IOFLAG.NE.3)DELZ=DELZ*FTKM
        Z=ALTITO*FTKM
        PHEE=DABS(GCLAT)
        DPHI=(90.0-PHEE)**2
        FZ=0.22+0.00258*Z**1.5
        IF(FZ.GT.5.0D0)FZ=5.0D0
C
C   LARGE-SCALE SCALE HEIGHT - DENSITY
        LVLRHO=(20.7-1.346E-3*DPHI)*FZ
C   LARGE-SCALE SCALE HEIGHT - TEMPERATURE
        LVLTMP=7.3*FZ
C   LARGE-SCALE SCALE HEIGHT - WIND
        LVLWND=(31.2-3.503E-3*DPHI)*FZ
C   SMALL-SCALE SCALE HEIGHT - DENSITY
        LVSRHO=(11.0-2.102E-4*DPHI)*FZ
C   SMALL-SCALE SCALE HEIGHT - TEMPERATURE
        LVSTMP=(3.0+5.146E-4*DPHI)*FZ
C   SMALL-SCALE SCALE HEIGHT - WINDS
        LVSWND=(6.2-3.615E-4*DPHI)*FZ
C   LARGE-SCALE HORIZONTAL SCALE
        LHL=900.0+6.0*Z
C   SMALL-SCALE HORIZONTAL SCALE
        LHS=20.0+0.0125*Z**2
        IF(LHS.GT.400.0D0)LHS=400.0D0
C
C ----- TERMS NEEDED FOR CALCULATING PERTERBATIONS -----
C
        RRHOL=EXP(-1.0*DSQRT((DELX/LHL)**2+(DELZ/LVLRHO)**2))
        RTEEL=EXP(-1.0*DSQRT((DELX/LHL)**2+(DELZ/LVLTMP)**2))
        RUL=EXP(-1.0*DSQRT((DELX/LHL)**2+(DELZ/LVLWND)**2))
C
        RRHOS=EXP(-1.0*DSQRT((DELX/LHS)**2+(DELZ/LVSRHO)**2))
        RTEES=EXP(-1.0*DSQRT((DELX/LHS)**2+(DELZ/LVSTMP)**2))
        RUS=EXP(-1.0*DSQRT((DELX/LHS)**2+(DELZ/LVSWND)**2))
C
        RVL=RUL
        RVS=RUS
        AL=RRHOL*SDRHOL/OSRHOL
        BL=SDRHOL*DSQRT(FP1-RRHOL**2)
        AS=RRHOS*SDRHOS/OSRHOS
        BS=SDRHOS*DSQRT(FP1-RRHOS**2)
   10  CONTINUE
C
C***   CALCULATE NEW RANDOM TEMPERATURE
C
        RTRHO2L=(SDPRSL**2-SDRHOL**2-SDTMPL**2)/(2.*SDRHOL*SDTMPL)
        RTRHO1L=(OSPRSL**2-OSRHOL**2-OSTMPL**2)/(2.*OSRHOL*OSTMPL)
```

```
C
         RTRHO2S=(SDPRSS**2-SDRHOS**2-SDTMPS**2)/(2.*SDRHOS*SDTMPS)
         RTRHO1S=(OSPRSS**2-OSRHOS**2-OSTMPS**2)/(2.*OSRHOS*OSTMPS)
C
         CEEL=(SDTMPL/OSTMPL)*((RTEEL-RRHOL*RTRHO2L*RTRHO1L)/
     1    (1.-RRHOL**2*RTRHO1L**2))
         CEES=(SDTMPS/OSTMPS)*((RTEES-RRHOS*RTRHO2S*RTRHO1S)/
     1    (1.-RRHOS**2*RTRHO1S**2))
C
         DEEL=(RTEEL*SDTMPL-CEEL*OSTMPL)/(AL*RTRHO1L*OSRHOL)
         DEES=(RTEES*SDTMPS-CEES*OSTMPS)/(AS*RTRHO1S*OSRHOS)
   25   CONTINUE
C        ISEED=INT(1.0D8*RAN0(ISEED))
         EEEL=DSQRT(SDTMPL**2-CEEL**2*OSTMPL**2-DEEL**2*SDRHOL**2-2.
     1 *CEEL*DEEL*RRHOL*RTRHO1L*OSTMPL*SDRHOL)
         EEES=DSQRT(SDTMPS**2-CEES**2*OSTMPS**2-DEES**2*SDRHOS**2-2.
     1 *CEES*DEES*RRHOS*RTRHO1S*OSTMPS*SDRHOS)
c ****compute new perterbations in same order used by GRAM****
c***TOMO***FORCED UNBIASED DETERMINATION of GAUSSIAN ERRORS (FUDGE)
            RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
            ICTR=ICTR+1
            RLIL1=XKR(ISEED)-RBAR
         PRINT*,'FROM WITHIN RNDATM, ICTR = ',ICTR
C         RLIL1=XKR(ISEED)
         RHOPS=GMRHO*(AS*((ORHOPS)/OGMRHO)+BS*RLIL1)
c***TOMO***
            RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
            ICTR=ICTR+1
            RLIL2=XKR(ISEED)-RBAR
C         RLIL2=XKR(ISEED)
         TEMPPS=GMTMP*(CEES*((OTMPPS)/OGMTMP)+DEES*
     1 ((RHOPS)/GMRHO)+EEES*RLIL2)
c
c **** calls for small scale U and V to keep number of random
c        calls the same as GRAM*****
c
c***TOMO***
            RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
            ICTR=ICTR+1
            RLIL3=XKR(ISEED)-RBAR
C             RLIL3=XKR(ISEED)
c***TOMO***
            RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
            ICTR=ICTR+1
            RLIL4=XKR(ISEED)-RBAR
C             RLIL4=XKR(ISEED)
C
c***TOMO***
            RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
            ICTR=ICTR+1
            RLIL5=XKR(ISEED)-RBAR
C             RLIL5=XKR(ISEED)
         RHOPL=GMRHO*(AL*((ORHOPL)/OGMRHO)+BL*RLIL5)
c***TOMO***
            RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
            ICTR=ICTR+1
            RLIL6=XKR(ISEED)-RBAR
C             RLIL6=XKR(ISEED)
         TEMPPL=GMTMP*(CEEL*((OTMPPL)/OGMTMP)+DEEL*
     1 ((RHOPL)/GMRHO)+EEEL*RLIL6)
```

```
c
c **** calls for large scale U and V to keep number of random
c       calls the same as GRAM*****
c
c***TOMO***
           RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
           ICTR=ICTR+1
           RLIL7=XKR(ISEED)-RBAR
C            RLIL7=XKR(ISEED)
c***TOMO***
           RBAR=((RBAR*ICTR)+XKR(ISEED))/(ICTR+1)
           ICTR=ICTR+1
           RLIL8=XKR(ISEED)-RBAR
C            RLIL8=XKR(ISEED)
c      WRITE(98,67)rlil1,rlil2,rlil3,rlil4,rlil5,rlil6,rlil7,rlil8
C     1     SDRHOS
  67    FORMAT(8F10.6)
C
        DENS=RHOPS+RHOPL+GMRHO
C       DENS=GMRHO+RHOPS
        RHOLIM=0.1*GMRHO
        IF(DENS.LT.RHOLIM) GOTO 25
        ATEM=TEMPPS+TEMPPL+GMTMP
C       ATEM=GMTMP+TEMPPS
        TEMLIM=0.1*GMTMP
        IF(ATEM.LT.TEMLIM)GOTO 25
C
C *** CALCULATE NEW SPEED OF SOUND ***
C
        CS=DSQRT(ATMOSK(1)*ATEM)
C
C***  CALCULATE NEW RANDOM PRESSURE
C
        PRESPL=GMPRS*((RHOPL/GMRHO)+(TEMPPL/GMTMP))
        PRESPS=GMPRS*((RHOPS/GMRHO)+(TEMPPS/GMTMP))
        PRES=PRESPL+GMPRS+PRESPS
C       PRES=GMPRS+PRESPS
C
C *** CALCULATE RANDOM WIND PERTURBATIONS ***
C
        FL=(SDUL/OSUL)*(RUL-RRHOL*CURHOL*OCURHL)/
     1      (FP1-RRHOL**2*OCURHL**2)
        FS=(SDUS/OSUS)*(RUS-RRHOS*CURHOS*OCURHS)/
     1      (FP1-RRHOS**2*OCURHS**2)
        GL=(RUL*SDUL-FL*OSUL)/(RRHOL*OCURHL*SDRHOL)
        GS=(RUS*SDUS-FS*OSUS)/(RRHOS*OCURHS*SDRHOS)
C
        HL=DSQRT(SDUL**2-FL*FL*OSUL**2-GL*GL*SDRHOL**2
     1          -FP2*FL*GL*RRHOL*OCURHL*SDRHOL*OSUL)
        HS=DSQRT(SDUS**2-FS*FS*OSUS**2-GS*GS*SDRHOS**2
     1          -FP2*FS*GS*RRHOS*OCURHS*SDRHOS*OSUS)
C
        UPL=FL*OUPL+GL*RHOPL+HL*RLIL7
        UPS=FS*OUPS+GS*RHOPS+HS*RLIL3
C
        XIL=(SDVL/OSVL)*(RVL-RRHOL*CVRHOL*OCVRHL)/
     1      (FP1-RRHOL**2*OCVRHL**2)
        XIS=(SDVS/OSVS)*(RVS-RRHOS*CVRHOS*OCVRHS)/
     1      (FP1-RRHOS**2*OCVRHS**2)
C
```

```
      XJL=(RVL*SDVL-XIL*OSVL)/(RRHOL*OCVRHL*SDRHOL)
      XJS=(RVS*SDVS-XIS*OSVS)/(RRHOS*OCVRHS*SDRHOS)
C
      XKL=DSQRT(SDVL**2-XIL*XIL*OSVL**2-XJL*XJL*SDRHOL**2
     1        -FP2*XIL*XJL*RRHOL*OCVRHL*SDRHOL*OSVL)
      XKS=DSQRT(SDVS**2-XIS*XIS*OSVS**2-XJS*XJS*SDRHOS**2
     1        -FP2*XIS*XJS*RRHOS*OCVRHS*SDRHOS*OSVS)
C
      VPL=XIL*OVPL+XJL*RHOPL+XKL*RLIL8
      VPS=XIS*OVPS+XJS*RHOPS+XKS*RLIL4
C
      WRITE(98,*)SDUS,OSUS,RUS,CURHOS,OCVRHS,SDVS,OSVS,CVRHOS
     1            ,OCVRHS
      WRITE(98,*)FS,GS,HS,XIS,XJS,XKS
      WRITE(98,*)SDUL,OSUL,RUL,CURHOL,OCVRHL,SDVL,OSVL,CVRHOL
     1            ,OCVRHL
      WRITE(98,*)FL,GL,HL,XIL,XJL,XKL
c     VW(1)=GMU*(1+UPL+UPS)
c     VW(2)=GMV*(1+VPL+VPS)
C
C ** UPDATE 'OLD' VARIABLES
C
      XOLD(1)=GDLAT
      XOLD(2)=LONG
      XOLD(3)=XI(3)
      OLDZ=ALTITO
      AROLD(1)=PRES
      AROLD(2)=DENS
      AROLD(3)=ATEM
      AROLD(4)=SDPRSS
      AROLD(5)=SDRHOS
      AROLD(6)=SDTMPS
      AROLD(7)=GMPRS
      AROLD(8)=GMRHO
      AROLD(9)=GMTMP
      AROLD(10)=SDPRSL
      AROLD(11)=SDRHOL
      AROLD(12)=SDTMPL
      AROLD(13)=RHOPL
      AROLD(14)=TEMPPL
      AROLD(15)=RHOPS
      AROLD(16)=TEMPPS
      AROLD(17)=SDUL
      AROLD(18)=SDUS
      AROLD(19)=SDVL
      AROLD(20)=SDVS
      AROLD(21)=CURHOL
      AROLD(22)=CURHOS
      AROLD(23)=CVRHOL
      AROLD(24)=CVRHOS
      AROLD(25)=UPL
      AROLD(26)=UPS
      AROLD(27)=VPL
      AROLD(28)=VPS
  200 ITTT=NINT(TIME)
      SAVPRES=PRES
      SAVDENS=DENS
      SAVATEM=ATEM
      CALL ATMOS5
      PRES76=PRES
```

```
      ATEM76=ATEM
      DENS76=DENS
      PRES=SAVPRES
      DENS=SAVDENS
      ATEM=SAVATEM
      WLONG=360.0D0-LONG
      WRITE(99,*)ITTT,ALTITO,GCLAT,WLONG,PRES,DENS,ATEM,VA,VW,
     1           PRES76,DENS76,ATEM76
  201 CONTINUE
      RETURN
      END
      SUBROUTINE RNDATI
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 LVLRHO,LVSRHO,LVLTMP,LVSTMP,LVLWND,LVSWND
      REAL*8 LHL,LHS
C
C  THIS INITIALIZES THE VARIABLES USED IN THE RANDOM
C ATMOSPHERE MODEL
      INCLUDE 'POST3D_INCLUDE:RAND.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:MOTVC.INC/LIST'
      INCLUDE'POST3D_INCLUDE: SPECAL.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:MOTIC.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:AUXVC.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:ALTITO.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:MOTBL.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:VXIVC.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:GCRADC.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:SERVC.INC/LIST'
      INCLUDE 'POST3D_INCLUDE:FUDGE.INC/LIST'
      INCLUDE'POST3D_INCLUDE: GUIDJC.INC/LIST'
C
C XOLD(1)= OLD GEODETIC ALTITUDE
C XOLD(2)= OLD LONGITUDE
C XOLD(3)= INERTIAL POSITION VECTOR
C XOLD(4)= ALTITUDE
C
C AROLD(1)= OLD PRESSURE
C AROLD(2)= OLD DENSITY
C AROLD(3)= OLD TEMPERATURE
C AROLD(4)= OLD PRESSURE STANDARD DEVIATION - SMALL SCALE
C AROLD(5)= OLD DENSITY STANDARD DEVIATION - S.S.
C AROLD(6)= OLD TEMPERATURE STANDARD DEVIATION - S.S.
C AROLD(7)= OLD MEAN PRESSURE
C AROLD(8)= OLD MEAN DENSITY
C AROLD(9)= OLD MEAN TEMPERATURE
C AROLD(10)= OLD PRESSURE STANDARD DEVIATION - LARGE SCALE
C AROLD(11)= OLD DENSITY STANDARD DEVIATION - L.S.
C AROLD(12)= OLD TEMPERATURE STANDARD DEVIATION - L.S.
C AROLD(13)= OLD L.S. DENSITY PERTURBATION
C AROLD(14)= OLD L.S. TEMPERATURE PERTURBATION
C AROLD(15)= OLD S.S. DENSITY PERTURBATION
C AROLD(16)= OLD S.S. TEMPERATURE PERTURBATION
C AROLD(17)= OLD L.S. NORTHERLY WIND STANDARD DEV.
C AROLD(18)= OLD S.S. NORTHERLY WIND STANDARD DEV.
C AROLD(19)= OLD L.S. EASTERLY WIND STANDARD DEV.
C AROLD(20)= OLD S.S. EASTERLY WIND STANDARD DEV.
C AROLD(21)= OLD L.S. DENSITY-NORTHERLY WIND CORRELATION
C AROLD(22)= OLD S.S. DENSITY-NORTHERLY WIND CORRELATION
C AROLD(23)= OLD L.S. DENSITY-EASTERLY WIND CORRELATION
C AROLD(24)= OLD S.S. DENSITY-EASTERLY WIND CORRELATION
```

```
C AROLD(25)= OLD L.S. NORTHERLY WIND PERTERBATION
C AROLD(26)= OLD S.S. NORTHERLY WIND PERTERBATION
C AROLD(27)= OLD L.S. EASTERLY WIND PERTERBATION
C AROLD(28)= OLD S.S. EASTERLY WIND PERTERBATION
C
C *** INITIALIZE MEAN VALUES ***
C
      AROLD(7)=EXP(GENTAB(PREST))
      AROLD(8)=EXP(GENTAB(DENST))
      AROLD(9)=GENTAB(ATEMT)
C
      IF(SPECI(9).LT.0.0)GOTO 200
C
C  ** INITIALIZE STANDARD ERRORS **
C
C
      AROLD(4)=DSQRT(FP1-GENTAB(SDPRLT))*GENTAB(SDPRST)
      AROLD(5)=DSQRT(FP1-GENTAB(SDRHLT))*GENTAB(SDRHST)
      AROLD(6)=DSQRT(FP1-GENTAB(SDTMLT))*GENTAB(SDTMST)
C
      AROLD(10)=DSQRT(GENTAB(SDPRLT))*GENTAB(SDPRST)
      AROLD(11)=DSQRT(GENTAB(SDRHLT))*GENTAB(SDRHST)
      AROLD(12)=DSQRT(GENTAB(SDTMLT))*GENTAB(SDTMST)
C
C TYPE I CORRELATION
C     AROLD(17)=DSQRT(GENTAB(SDULT))*GENTAB(SDUST)
C     AROLD(18)=DSQRT(FP1-GENTAB(SDULT))*GENTAB(SDUST)
C     AROLD(19)=DSQRT(GENTAB(SDVLT))*GENTAB(SDVST)
C     AROLD(20)=DSQRT(FP1-GENTAB(SDVLT))*GENTAB(SDVST)
C
C TYPE II CORRELATION
      AROLD(17)=DSQRT(GENTAB(SDULT)*DABS(GENTAB(SDUST)))
      AROLD(18)=DSQRT(FP1-GENTAB(SDULT)*DABS(GENTAB(SDUST)))
      AROLD(19)=DSQRT(GENTAB(SDVLT)*DABS(GENTAB(SDVST)))
      AROLD(20)=DSQRT(FP1-GENTAB(SDVLT)*DABS(GENTAB(SDVST)))
C
      AROLD(21)=GENTAB(CURHLT)
      AROLD(22)=GENTAB(CURHST)
      AROLD(23)=GENTAB(CVRHLT)
      AROLD(24)=GENTAB(CVRHST)
C
C  *** LET INITIAL REALIZATIONS
C  OF RANDOM COMPONENTS GIVE MEAN PLUS SQRT OF SUM OF SQUARES
C  OF SMALL AND LARGE S.D. WITH GAUSSIAN RANDOM NUMBER.***
C
C
  11  CONTINUE
C     PPRIME=DSQRT(AROLD(10)**2+AROLD(4)**2)*XKR(ISEED)
      PPRIME=0.0D0
      IF(PPRIME.LT.-1.0D0)GOTO 11
      AROLD(1)=AROLD(7)*(1+PPRIME)
C
  22  CONTINUE
C     DPRIME=DSQRT(AROLD(11)**2+AROLD(5)**2)*XKR(ISEED)
      DPRIME=0.0D0
      IF(DPRIME.LT.-1.0D0)GOTO 22
      AROLD(2)=AROLD(8)*(1+DPRIME)
C
  33  CONTINUE
C     TPRIME=DSQRT(AROLD(12)**2+AROLD(6)**2)*XKR(ISEED)
```

```
      TPRIME=0.0D0
      IF(TPRIME.LT.-1.0D0)GOTO 33
      AROLD(3)=AROLD(9)*(1+TPRIME)
C
C ASSIGN OLD PERTURBATIONS BASED ON RELATIVE SIZE OF S.D.
C
      SIGP=AROLD(4)+AROLD(10)
      SIGD=AROLD(5)+AROLD(11)
      SIGT=AROLD(6)+AROLD(12)
      AROLD(13)=(AROLD(11)/SIGD)*DPRIME*AROLD(8)
      AROLD(14)=(AROLD(12)/SIGT)*TPRIME*AROLD(9)
      AROLD(15)=(AROLD(5)/SIGD)*DPRIME*AROLD(8)
      AROLD(16)=(AROLD(6)/SIGT)*TPRIME*AROLD(9)
C
      AROLD(25)=0.0D0
      AROLD(26)=0.0D0
      AROLD(27)=0.0D0
      AROLD(28)=0.0D0
C
C SKIP TO HERE IF NOT USING PERTURBATTIONS
 200  CONTINUE
      XOLD(1)=GDLAT
      XOLD(2)=LONG
      XOLD(3)=XI(3)
      XOLD(4)=ALTITO
C
      PRES=AROLD(7)
      DENS=AROLD(8)
      ATEM=AROLD(9)
      CS=DSQRT(ATMOSK(1)*AROLD(9))
C
C   DON'T WNAT TO RE-INITIALIZE
c     IGF(1)=1
      RETURN
      END
      REAL*8 FUNCTION XKR(ISEED)
C
C  GAUSSIAN RANDOM NUMBER GENERATOR
C
C THIS ROUTINE FOLLOWS THE ARTICLE BY KINDERMAN AND RAMAGE
C "COMPUTER GENERATION OF NORMAL RANDOM VARIABLES"
C IN THE JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
C VOLUME 71, NUMBER 356, PP.893-896.
C
C CODED BY ERIC QUEEN 7/30/88
C
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 RANDOM4
      PHI(X)=DEXP(-0.5D0*X*X)/DSQRT(2.0D0*3.141592654D0)
      F(X)=PHI(X)-.180025191068563D0*(XI-DABS(X))
      XI =2.216035867166471D0
C                                         BEGIN STEP 1
      U=DBLE(RAN0(ISEED))
      IF(U.LT.0.884070402298758D0)THEN
              V=DBLE(RAN0(ISEED))
              XKR=XI*(1.131131635444180D0*U+V-1.0D0)
              RETURN
      ENDIF
C                                         BEGIN STEP 2
      IF(U.LT..973310954173898D0) THEN
```

```
C                                          BEGIN STEP 4
               IF(U.LT..958720824790463D0)THEN
C                                          BEGIN STEP 6
                       IF(U.LT..911312780288703D0)THEN
C                                          BEGIN STEP 8
  80                           V=DBLE(RAN0(ISEED))
                               W=DBLE(RAN0(ISEED))
                               Z=V-W
                               VWMIN=DMIN1(V,W)
                               VWMAX=DMAX1(V,W)
           T=.479727404222441D0-.595507138015940D0*VWMIN
           IF(VWMAX.LE.0 .805577924423817D0)GOTO 90
           TEMP1=.053377549506886D0*DABS(Z)
                               IF(F(T).GE.TEMP1)GOTO 90
                               GOTO 80
                       ELSE
C                                          BEGIN STEP 7
  70                           V=DBLE(RAN0(ISEED))
                               W=DBLE(RAN0(ISEED))
                               Z=V-W
                               VWMIN=DMIN1(V,W)
                               VWMAX=DMAX1(V,W)
                               T=.479727404222441D0+1.105473661022070D0*VWMIN
                               IF(VWMAX.LE..872834976671790D0)GOTO 90
                               TEMP2=.049264496373128D0*DABS(Z)
                               IF(F(T).GE.TEMP2)GOTO 90
                               GOTO 70
                       ENDIF
               ELSE
C                                          BEGIN STEP 5
  50                   V=DBLE(RAN0(ISEED))
                       W=DBLE(RAN0(ISEED))
                       Z=V-W
                       VWMIN=DMIN1(V,W)
                       VWMAX=DMAX1(V,W)
                       T=XI-.630834801921960D0*VWMIN
                       IF(VWMAX.LE..755591531667601D0)GOTO 90
                       TEMP3=.034240503750111D0*DABS(Z)
                       IF(F(T).GE.TEMP3)GOTO 90
                       GOTO 50
               ENDIF
       ELSE
C                                          BEGIN STEP 3
  30           V=DBLE(RAN0(ISEED))
               W=DBLE(RAN0(ISEED))
               TEMP5=0.5D0*XI*XI
               T=TEMP5-DLOG(W)
               TEMP4=V**2*T
               IF(TEMP4.GT.TEMP5)THEN
                       GOTO 30
               ELSE
                       IF(U.LT..986655477086949D0)THEN
                               XKR=DSQRT(2*T)
                               RETURN
                       ELSE
                               XKR=-DSQRT(2*T)
                               RETURN
                       ENDIF
               ENDIF
       ENDIF
```

```
C                                          BEGIN STEP 9
 90    CONTINUE
       IF(Z.LT.0.0D0)THEN
               XKR=T
               RETURN
       ELSE
               XKR=-T
               RETURN
       ENDIF
       RETURN
       END
      FUNCTION RAN0(IDUM)
      DIMENSION V(97)
      DATA IFF /0/
      IF(IDUM.LT.0.OR.IFF.EQ.0)THEN
        IFF=1
        ISEED=ABS(IDUM)
        IDUM=1
        DO 11 J=1,97
          DUM=RAN(ISEED)
11      CONTINUE
        DO 12 J=1,97
          V(J)=RAN(ISEED)
12      CONTINUE
        Y=RAN(ISEED)
      ENDIF
      J=1+INT(97.*Y)
      IF(J.GT.97.OR.J.LT.1)PAUSE
      Y=V(J)
      RAN0=Y
      V(J)=RAN(ISEED)
      RETURN
      END
```

Table 1. Coefficients of Scale Distances

Parameter	a	b
$L_{v_{\rho L}}$	20.7	-1.346×10^{-3}
$L_{v_{\rho S}}$	11.0	-2.102×10^{-4}
$L_{v_{TS}}$	3.0	5.146×10^{-4}
$L_{v_{TL}}$	7.3	0.0
$L_{v_{US}}$	6.2	3.615×10^{-4}
$L_{v_{UL}}$	31.2	-3.503×10^{-3}

Table 2. Point Statistics from 100 Sample AFE Trajectories

	Copropagated (mean, variance)	Precomputed (mean, variance)
Maximum Dynamic Pressure	1669.30	1546.74
$(kg/m.sec^2)$	10615.25	6402.82
Maximum Heating Rate	145.90	142.40
(btu/ft^2sec)	19.69	11.33
Heating Rate Integral	20187.43	19942.69
(btu/ft^2sec)	101451.30	82547.80
Final Energy (m^2/sec^2)	4232.32	4231.67
	73.97	30.89
Final Longitude (deg)	33.0	32.41
	2.4	3.4
Final Latitude (deg)	24.32	24.23
	0.16	0.24
Minimum Altitude (km)	74.25	75.07
	0.035	0.049

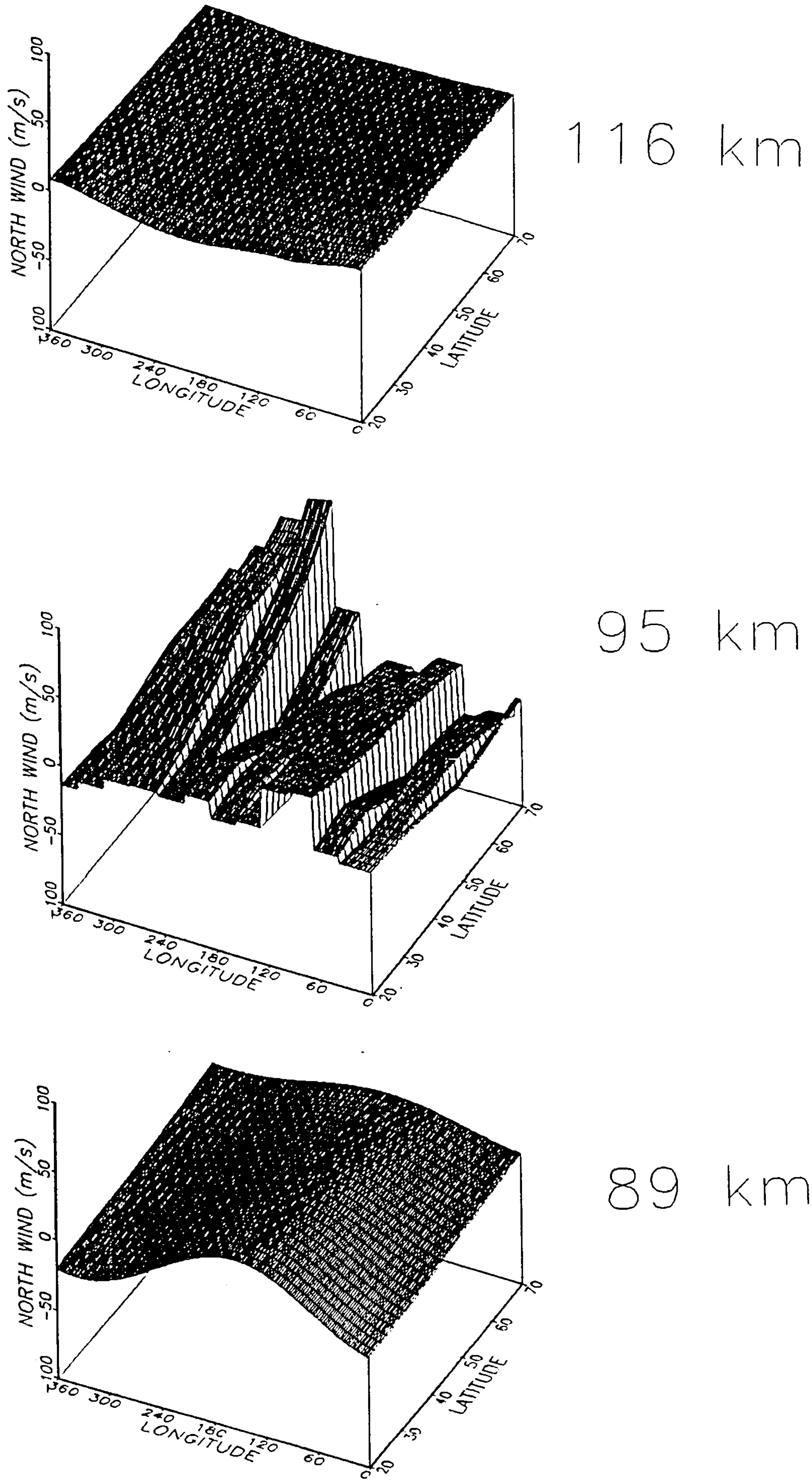

Figure 1. GRAM mean northward winds at three altitudes.

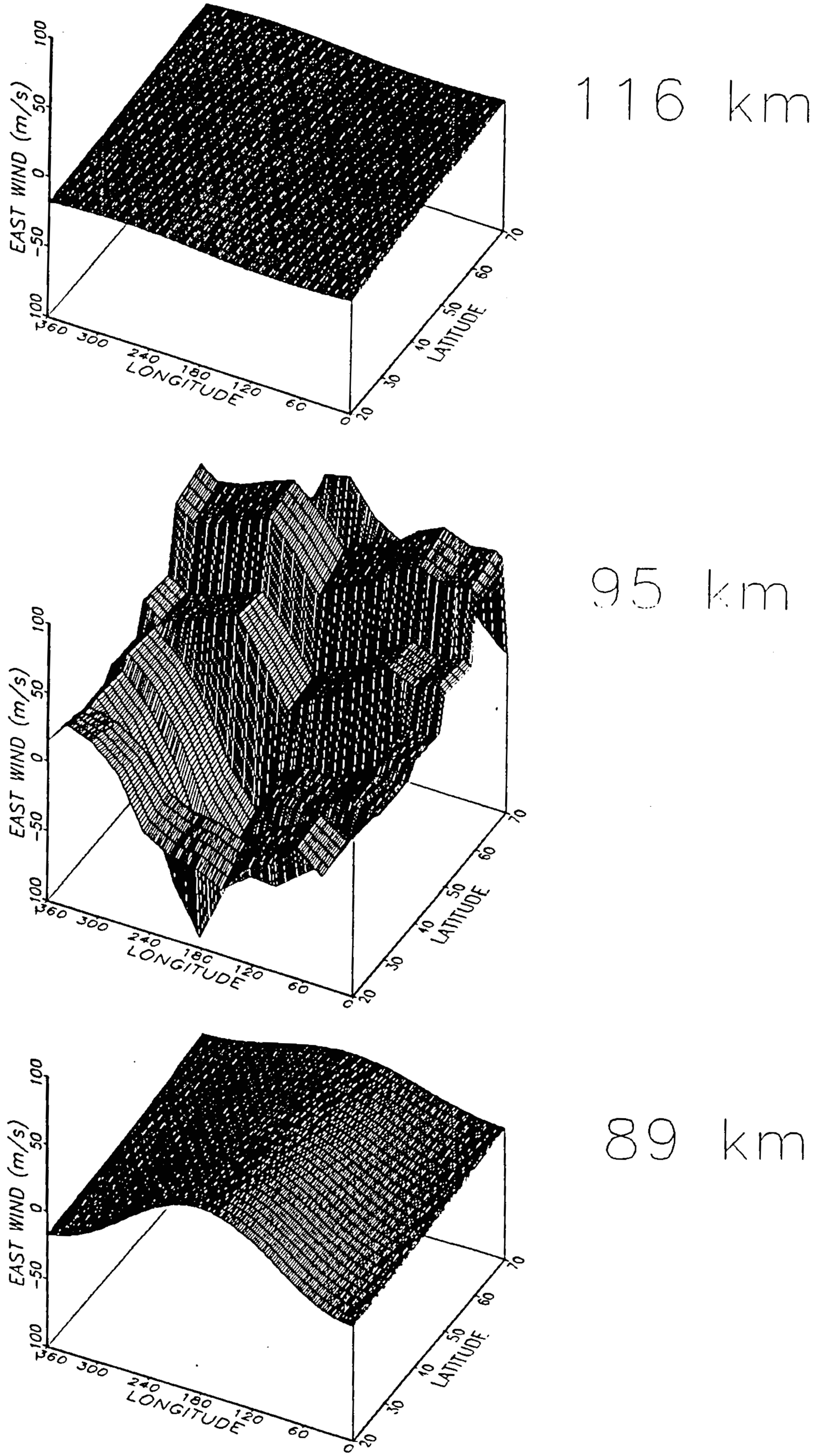

Figure 2. GRAM mean eastward winds at three altitudes.

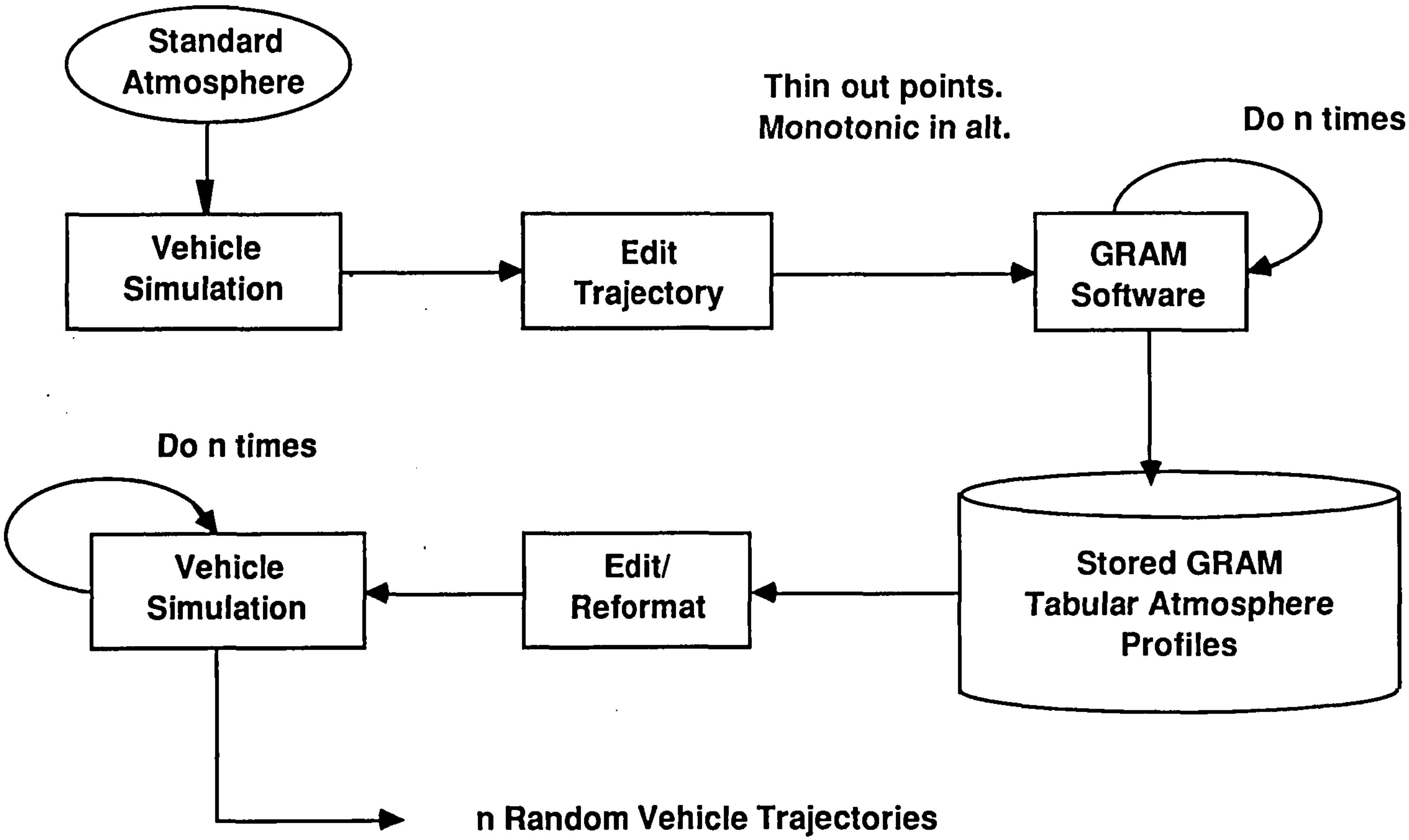

Figure 3.- Traditional, precomputed Monte Carlo Simulation Approach

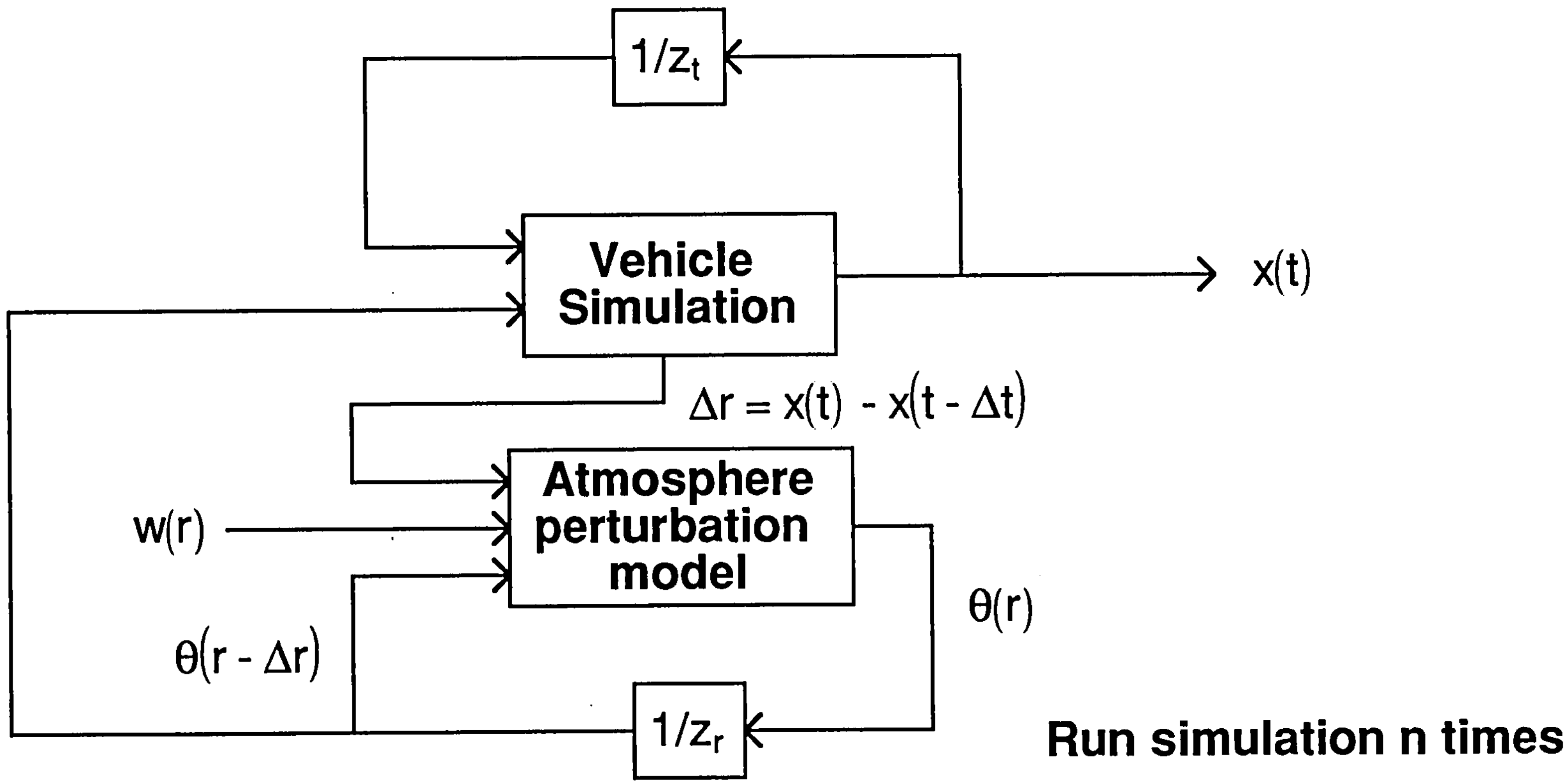

Figure 4.- Copropagated Monte Carlo simulation approach

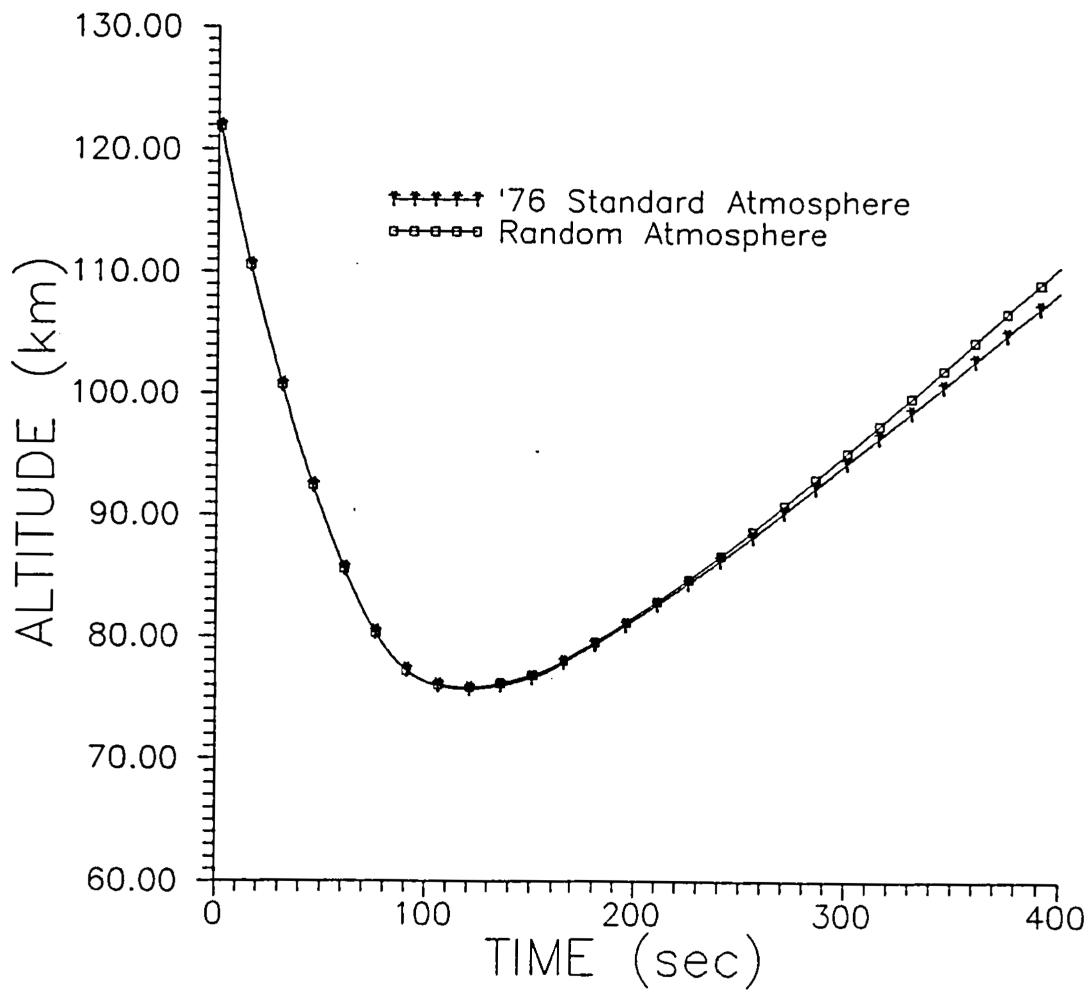

Figure 5.– Average Altitude Time histories of AFE flight through both Random and '76 Standard Atmospheres

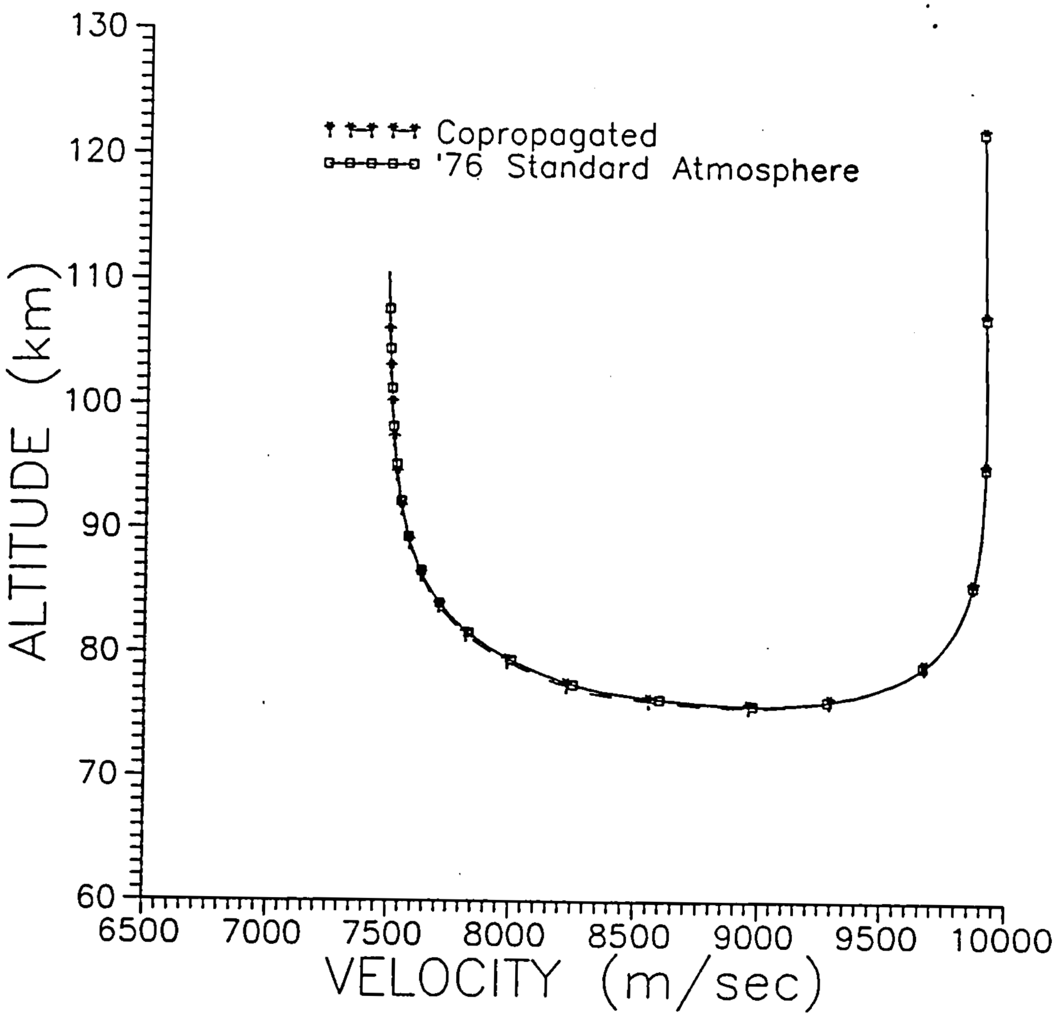

Figure 6.– Average Altitude Histories of AFE flight through both Copropagated and '76 Standard Atmospheres

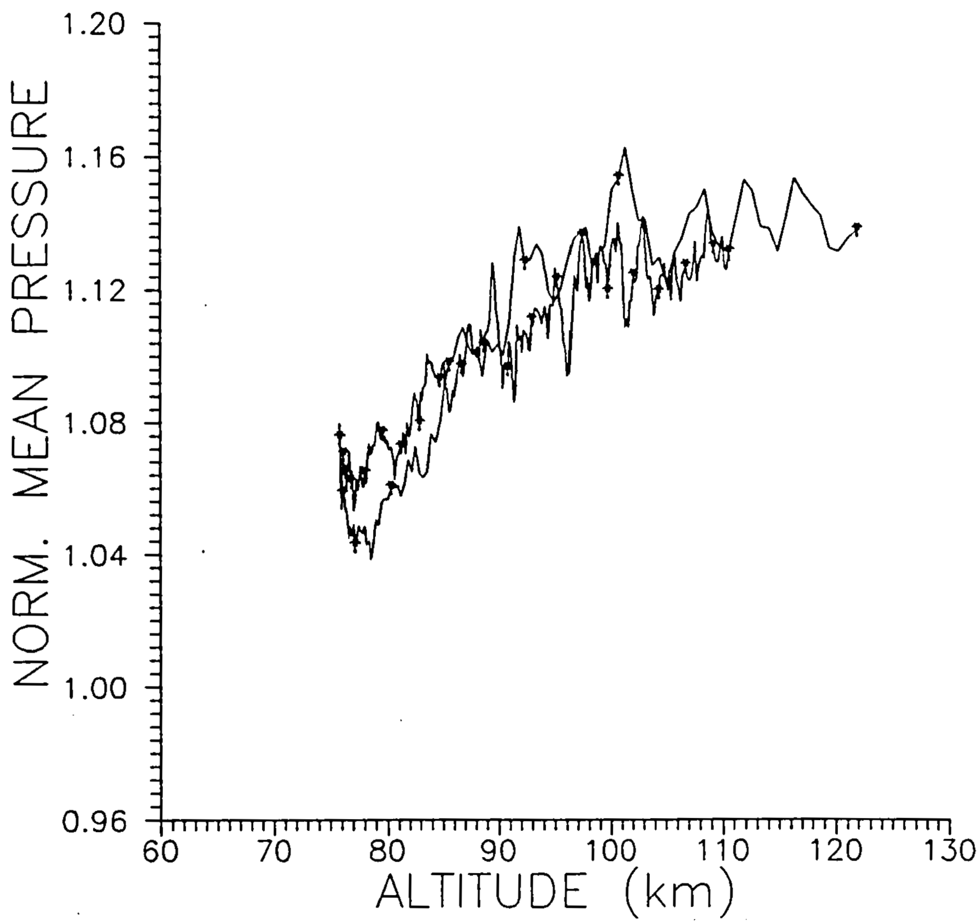

Figure 7.– Mean Pressures Normalized w.r.t. '76 Standard Atmosphere

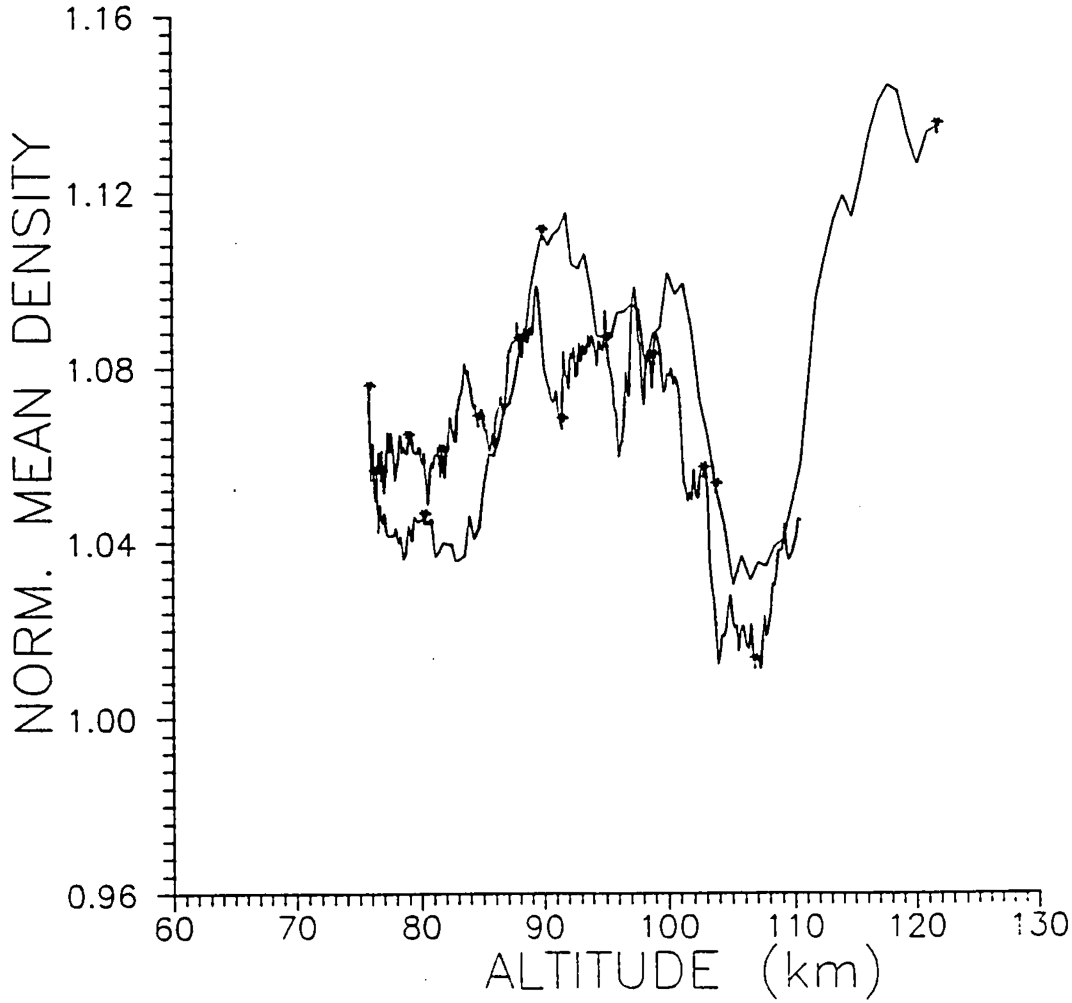

Figure 8.— Mean Densities Normalized w.r.t. '76 Standard Atmosphere

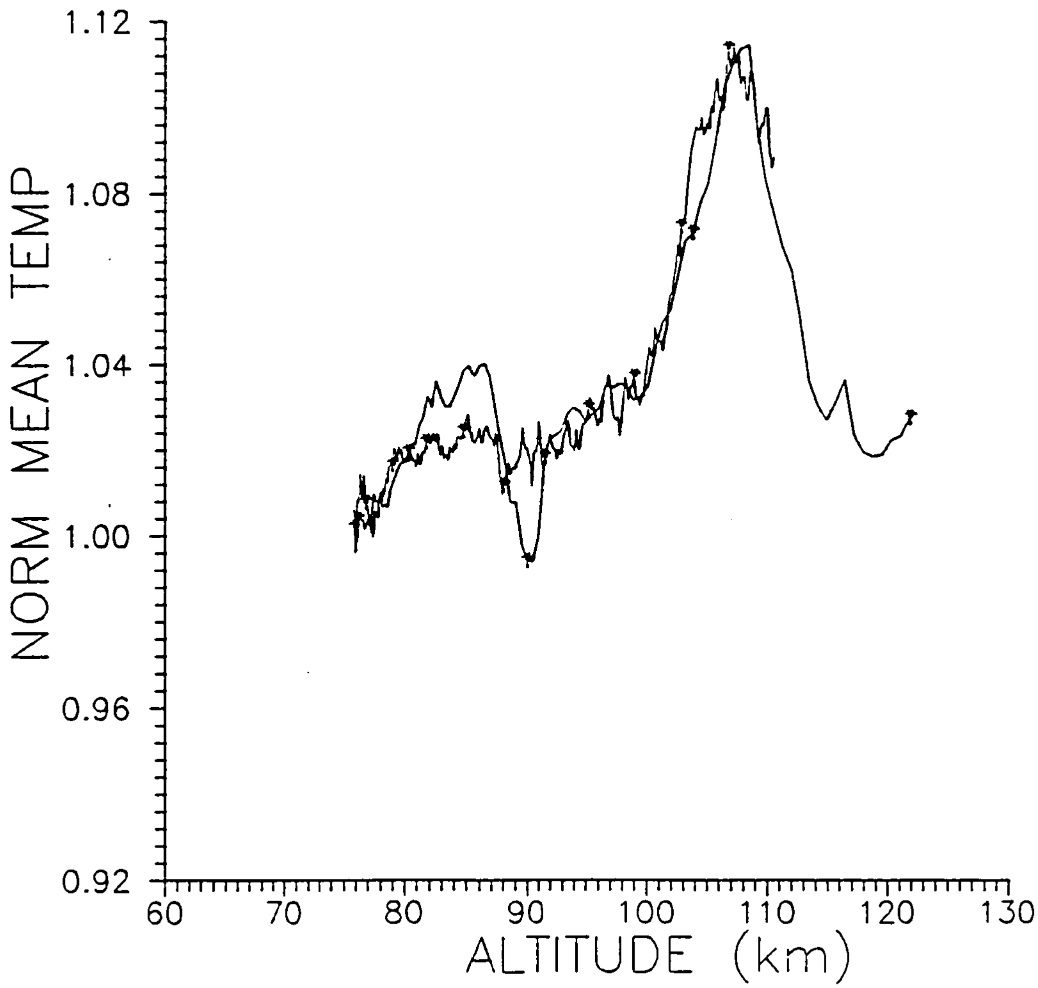

Figure 9.— Mean Temperatures Normalized w.r.t. '76 Standard Atmosphere

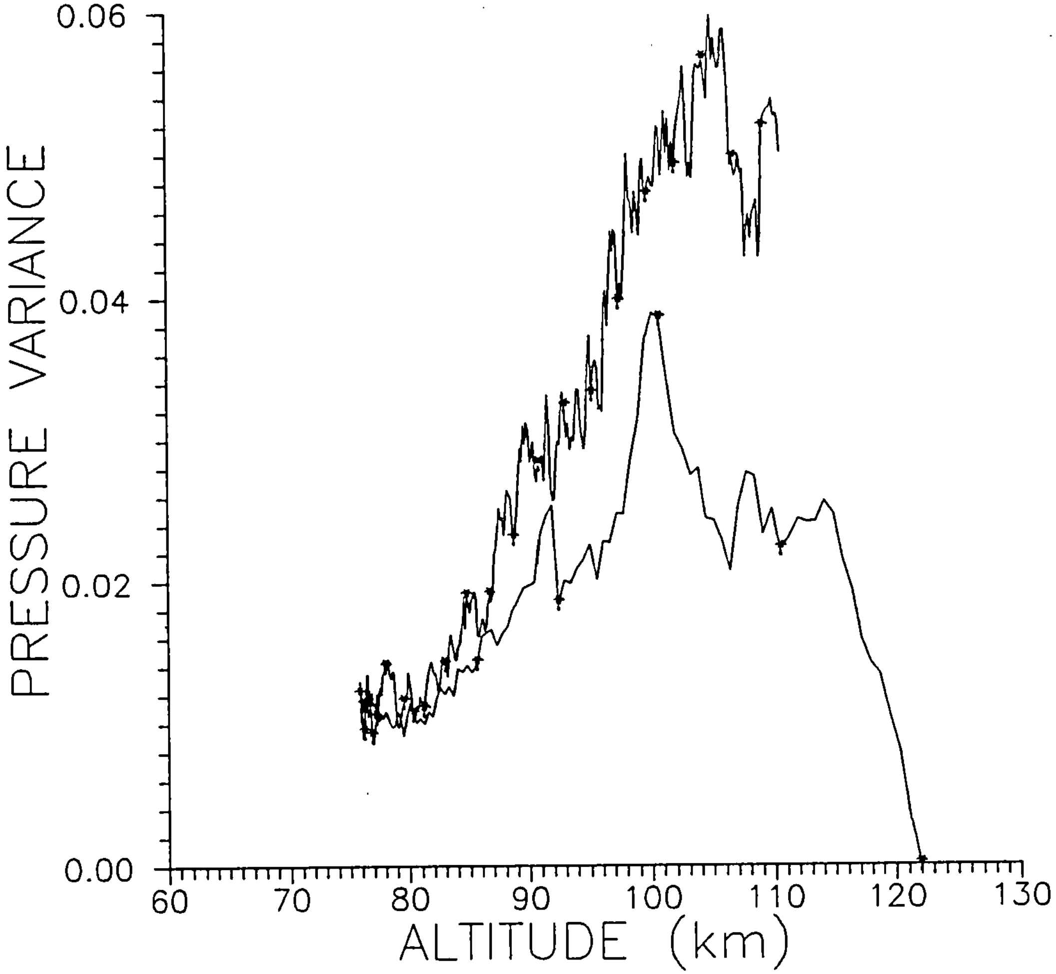

Figure 10.– Variance of Normalized Pressures

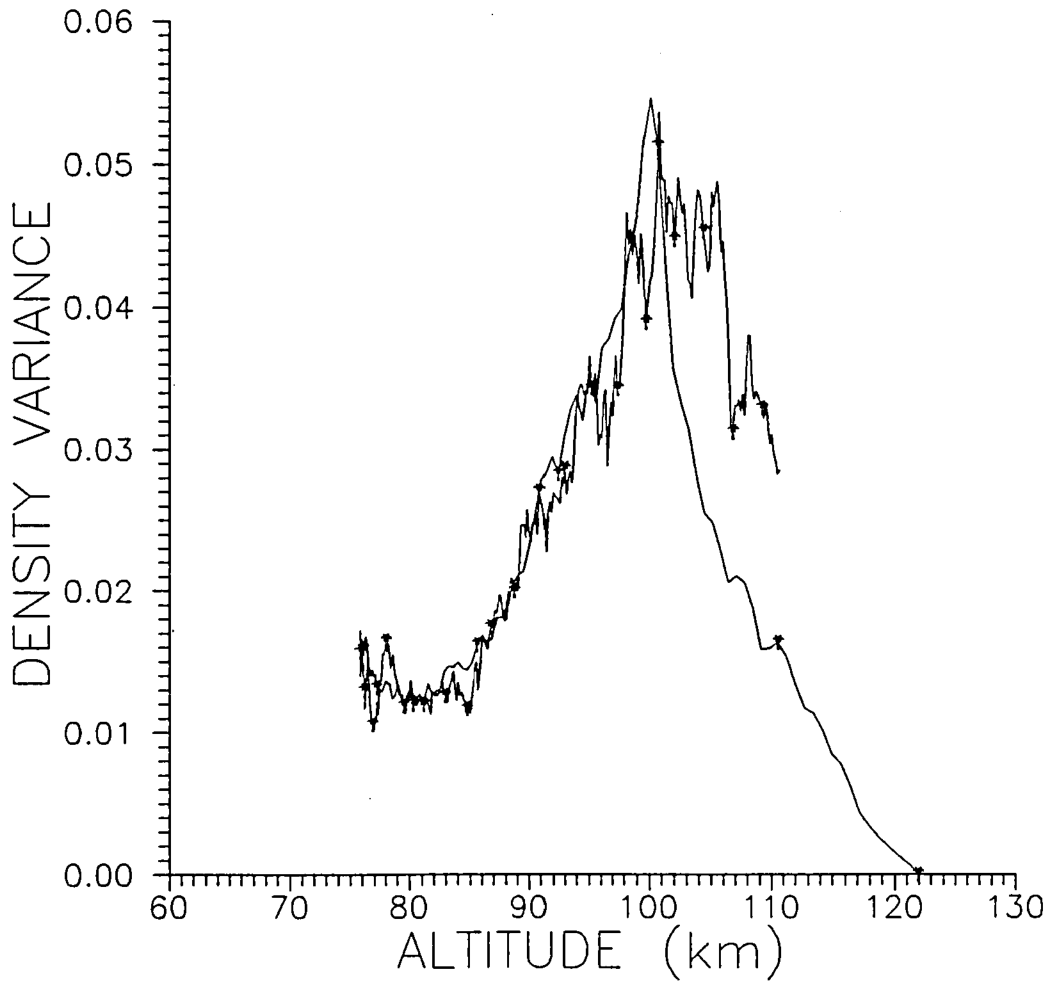

Figure 11.— Variance of Normalized Densities

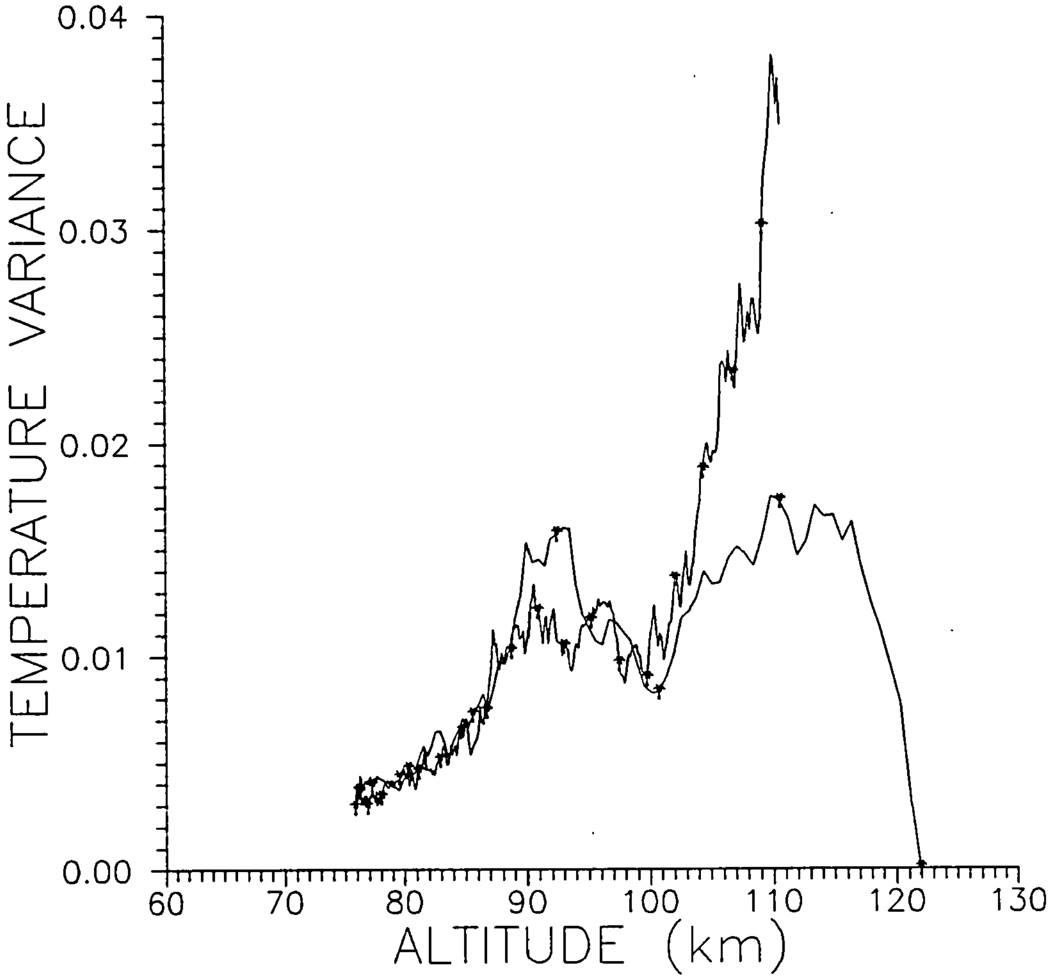

Figure 12.– Variance of Normalized Temperatures

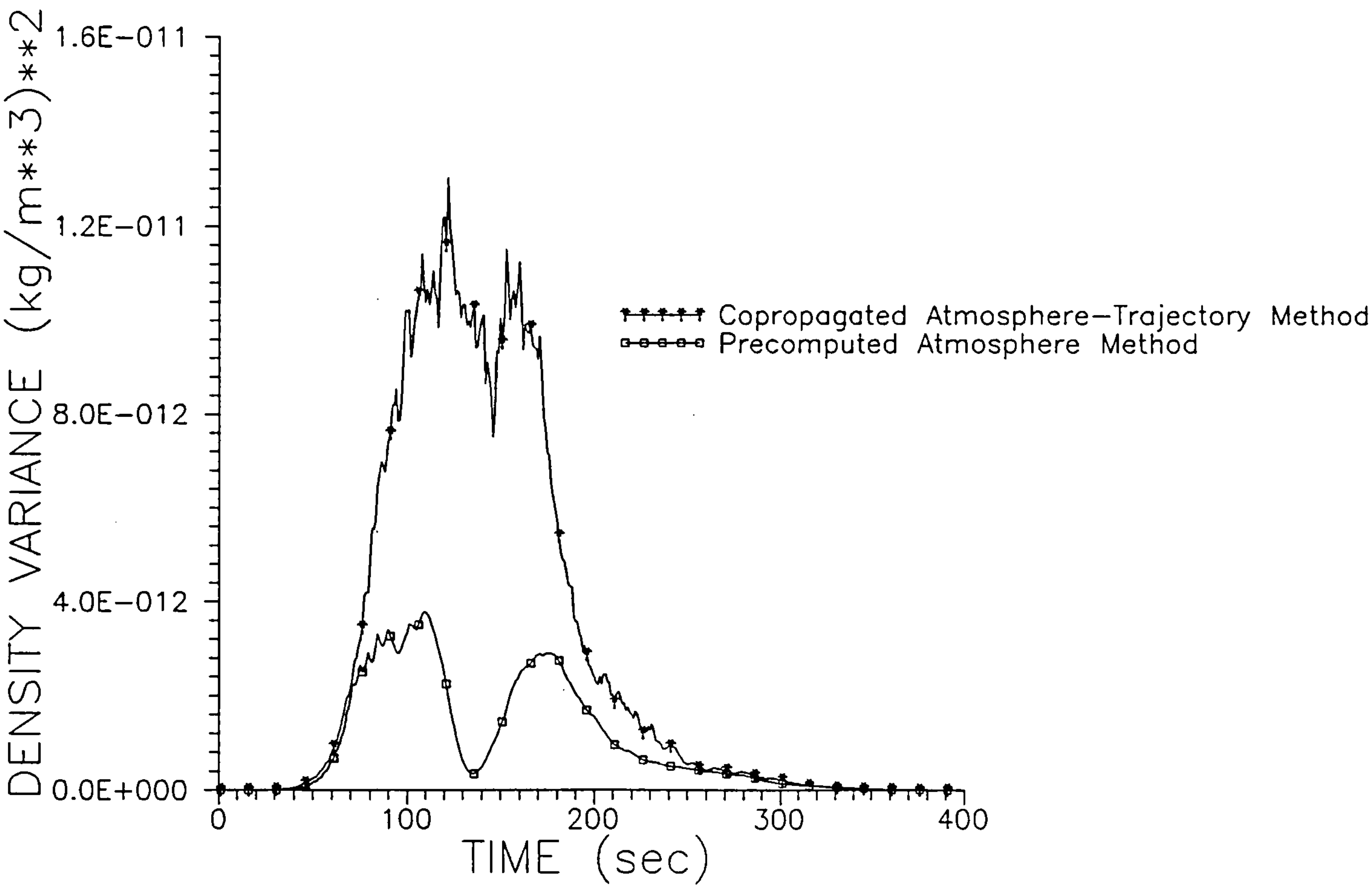

Figure 13.– Comparison of Density Variance Histories for Copropagated and Precomputed Approaches

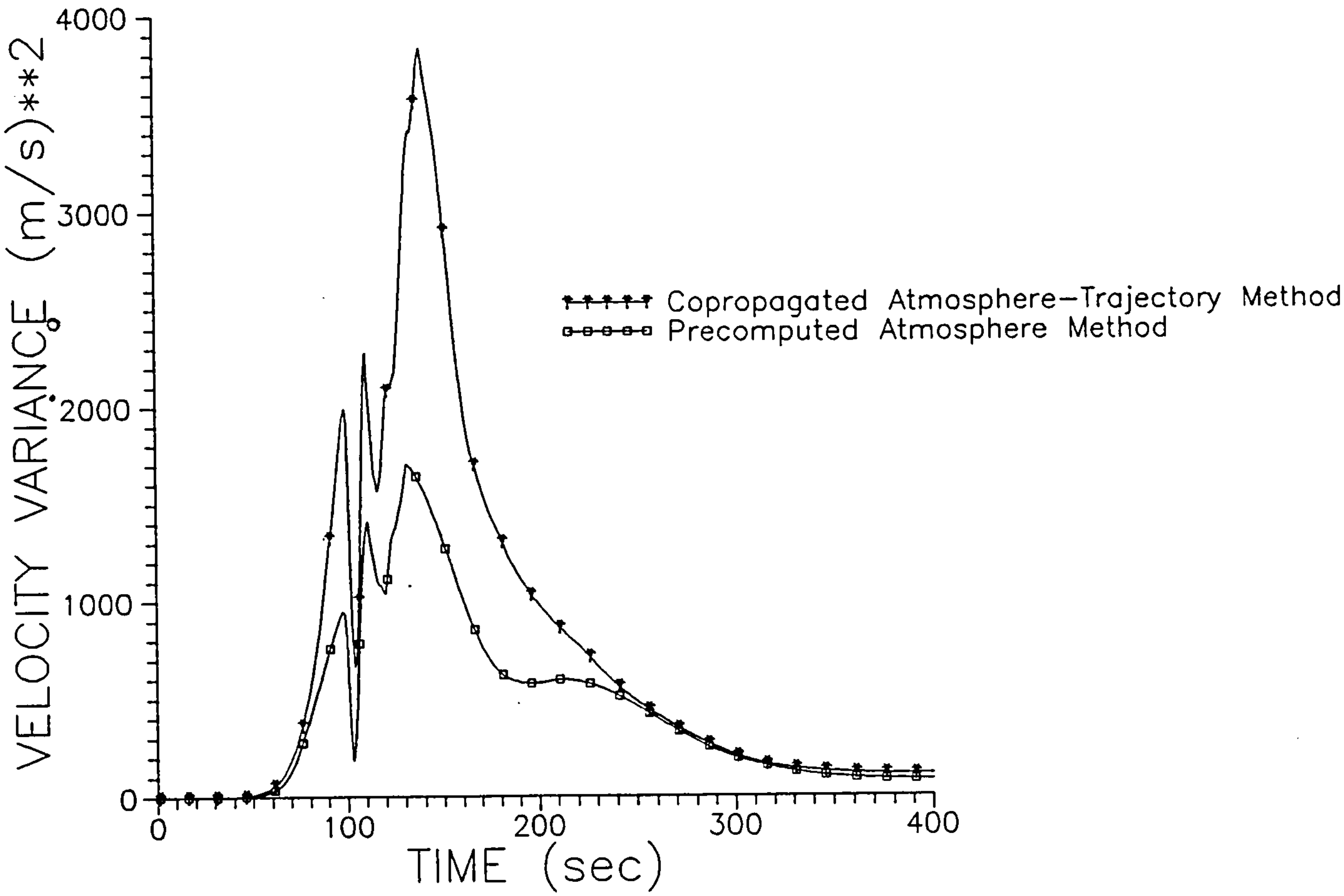

Figure 14.– Comparison of Velocity Variance Histories for Copropagated and Precomputed Approaches

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102600		

4. Title and Subtitle	5. Report Date
Comparison of Effects of Copropagated and Precomputed Atmosphere Profiles on Monte Carlo Trajectory Simulation	August 1990
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
E. M. Queen and T. M. O'Mara*	
	10. Work Unit No.
9. Performing Organization Name and Address	506-46-21-01
NASA Langley Research Center Hampton, VA 23665-5225	11. Contract or Grant No.
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Technical Memorandum
National Aeronautics and Space Administration Washington, DC 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

* Research Engineer, PRC Kentron Inc., 303 Butler Farm Road, Hampton, VA 23665

16. Abstract

A realization of a stochastic atmosphere model for use in simulations is presented. The model provides pressure, density, temperature, and wind velocity as a function of latitude, longitude, and altitude, and is implemented in a three-degree of freedom simulation package. This implementation is used in the Monte Carlo simulation of an aeroassisted orbital transfer maneuver and results are compared to those of a more traditional approach.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Random Atmosphere Atmosphere Modelling Co-propagated Atmosphere Profiles Precomputed Atmosphere Profiles	FEDD (Review for General Release August 31, 1992) Subject Category 18

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	46	

NASA FORM 1626 OCT 86

LANGLEY RESEARCH CENTER
3 1176 01343 5442