FINAL REPORT

on

An Algorithm to Generate Input Data from Meteorological and Space Shuttle Observations to Validate a CH_4-CO Model

by

Leonard K. Peters, Principal Investigator

and

John Yamanis, Co-Principal Investigator

Department of Chemical Engineering
University of Kentucky
Lexington, Kentucky 40506

October 15, 1981

NASA Research Grant No. NSG 1501

Grant Period: February 1, 1978 - October 31, 1980

AN ALGORITHM TO GENERATE INPUT DATA FROM METEOROLOGICAL AND SPACE SHUTTLE OBSERVATIONS TO VALIDATE A CH_4-CO MODEL

ABSTRACT

This research addressed objective procedures to analyze data from meteorological and Space Shuttle observations to validate a three-dimensional model which describes the transport and chemistry of carbon monoxide and methane in the troposphere. These efforts will ultimately provide the knowledge needed to understand the chemical dynamics of these species in the troposphere. There have been four aspects of this research: (1) Detailed evaluation of the variational calculus procedure, with the equation of continuity as a strong constraint, for adjustment of global tropospheric wind fields; (2) Reduction of the National Meteorological Center (NMC) Data Tapes for data input to the OSTA-1/MAPS Experiment; (3) Interpolation of the NMC Data for input to the CH_4-CO model; and (4) Temporal and spatial interpolation procedures of the CO measurements from the OSTA-1/MAPS Experiment to generate usable contours of the data. Each of these aspects is discussed in this report.

INTRODUCTION

Combined transport/chemistry models can be employed to analyze the circulation of pollutants from their sources to sinks. Depending on the time scale of the phenomena involved, these models can be broadly classified into urban, regional, or global scale. Sulfur dioxide and particulates are examples of pollutants that are important on urban scales, whereas the SO_2-sulfate conversion is representative of a problem of regional importance. Finally, carbon monoxide is a species that has significant sources and sinks on a global scale.

This report describes research to develop an algorithm to process

meteorological and CO observational data for model simulations that would be compared with CO concentrations obtained from measurements on Space Shuttle missions. In the future, data from National Meteorological Center tapes and CO observations from Space Shuttle experiments will be used to generate input and comparison data for model simulations based on the procedures discussed in this report.

OBJECTIVES OF RESEARCH

The overall objectives of this research can be summarized as follows:

a) Development of suitable spatial and temporal interpolation and extrapolation procedures to provide representative approximations of the discrete wind-field data which are necessary for the model simulations; and

b) Development of representative three-dimensional CO data from two-dimensional observations, and interpolation of meteorological data for use in tropospheric species transport models.

Most of the effort was expended on the first objective which, due to the size of the system, proved to be an unwieldy problem. Nevertheless, some suggestions relative to the second objective will be made.

DISCUSSION OF RESEARCH

The greatest emphasis during this research has dealt with the velocity field. This is most important since a mass-conservative wind field is essential to minimization of systematic numerical errors during integration of turbulent transport equations. The approaches which have been evaluated in the past to guarantee conservative wind fields will be briefly discussed. This actually involves two separate problems: (a) Establishing a conservative wind field from the observations generally available at 0:00 and 12:00 GMT; and

(b) Interpolating between these observation periods (wind fields are required at approximately 1/2-hour intervals for many models) to establish representative "best wind fields" at intermediate times that are also conservative. The thrust of our research in these areas will be discussed separately. The presentation has been divided into the following four sub-areas: (a) Spatial Adjustment of Data; (b) Temporal Interpolation of Meteorological Parameters; (c) Decoding the NMC Data Tape; and (d) Temporal and Spatial Interpolation of CO Data.

Spatial Adjustment of Data

A wind field that is free of errors and satisfies a particular set of general relations is a very important input parameter for all kinds of meteorological models, whether these models are for weather forecasting or for atmospheric transport studies. Unfortunately, such a wind field is not easily provided. The observed wind field data, as available mainly from rawinsonde measurements, contain observational errors and small scale effects, and as a result, do not satisfy general physical relations, say mass conservation or other diagnostic relations applicable to a particular atmospheric model. Such a wind field when input to model computations can lead to highly inaccurate results with initial errors multiplied many fold. In order to overcome this, it becomes an important and essential task to treat and adjust the observed wind field data in such a way that the observed data are minimally adjusted. But at the same time, the adjusted values should satisfy the physical relations applicable to a particular atmospheric model. For example, a mass consistent wind field is a must in the transport and diffusion atmospheric model. Thus, for such models the observed wind fields are adjusted so as to satisfy the mass conservation condition, i.e., the continuity equation.

Several methods of varying degrees of complexity have been investigated

to accomplish this mass conservation. The first and simplest is establishing a vertical velocity component by numerically differencing the equation of continuity*

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \qquad (1).$$

Thus, from Equation (1) the vertical velocity is immediately calculated taking $w = 0$ at $z = 0$ to initiate the calculation. This procedure has been used with completed initial simulations (Peters and Jouvanis, 1979 and Carmichael and Peters, 1981).

This procedure is very simple to use. However, due to errors in the reported observed wind field, the calculated vertical velocity can be more representative of "noisy data" than of the actual vertical motions. As a result, representativeness of the wind field may be sacrificed and not attained by this simple method. The procedures discussed next recognize that there are observational errors and make allowance for such.

A technique employed in the past (Endlich, 1967 and Goodlin, 1976) uses a point iterative method to adjust the observed wind field data so as to satisfy the desired physical constraints. Goodin et al. (1980) recently used a hybrid of the point iterative procedure with numerical satisfaction of the continuity equation.

A seemingly more general and powerful technique is that based on variational calculus principles. This technique, which was first proposed by Sasaki (1958), allows the observed wind field data to be changed by a minimum amount in a least squares sense, while at the same time, adjusted values satisfy the imposed physical conditions within numerical approximation errors. Subsidiary conditions that are to be satisfied exactly are termed strong

* All of the development presented in this section will assume constant density for simplicity. However, analysis using a varying density is an obvious and straightforward extension and is being included in the actual analysis.

constraints, while conditions being imposed approximately are weak constraints. This technique has been extended to a number of situations with various forms of constraints. Washington and Duquet (1963) used the technique with the geostrophic approximation while Sasaki (1958) and Stephens (1970) applied the balance equation as the dynamic constraint. Achtemeier (1975) extended the technique with a number of constraints such as the momentum, energy, continuity, and hydrostatic equations. Later work seems to lead to a more complex network of non-linear equations that must be linearized before solving them. Recently, Dickerson (1978) and Sherman (1978) used a variational technique to adjust the observed wind field in the San Francisco Bay area under the strong constraint of the continuity equation.

Generally, the mathematical formulations of the variational technique may be described as follows. Let u°, v°, and w° be the observed velocity components of a wind field, and u, v, and w be the corresponding modified adjusted values. The difference between the observed and adjusted field may be expressed in a least squares sense as

$$\alpha_1^2(u - u^\circ)^2 + \alpha_2^2(v - v^\circ)^2 + \alpha_3^2(w - w^\circ)^2 \qquad (2),$$

where α_1, α_2, and α_3 are weighting functions defined as the reciprocal of twice the variance of the errors of observations for the respective observed components. The simplification of $\alpha_1 = \alpha_2$ has been used in the past on the basis that the errors in the horizontal components should be comparable. In order to obtain the modified values, u, v, and w, it is required that the adjustment over space and time, subject to the satisfaction of the constraint equations, be a minimum. An adjustment functional, therefore, may be expressed as

$$I = \int_V \int_t [\alpha_1^2(u - u^\circ)^2 + \alpha_2^2(v - v^\circ)^2 + \alpha_3^2(w - w^\circ) + \gamma G_1 + \lambda G_2] \, dt dV \qquad (3),$$

where G_1 is a weak constraint which is introduced into the functional through the constant γ. The strong constraint G_2 is introduced through the Lagrangian multiplier λ. Equation (3) defines a variational problem in which a minimum value of the integrand under the given constraints G_1 and G_2 is being sought; i.e, the first variation on I must vanish, $\delta I = 0$. As a result, the associated Euler-Lagrange equations along with the so-called natural boundary conditions being inherited from the variational formulation must be satisfied. For example, a functional such as given in Equation (3) with the continuity equation as the only strong constraint and no weak constraint yields the following associated Euler-Lagrange equations along with natural boundary conditions (Sasaki, 1958).

$$u = u^\circ + \frac{1}{2\alpha_1^2}\frac{\partial \lambda}{\partial x} \tag{4}$$

$$v = v^\circ + \frac{1}{2\alpha_1^2}\frac{\partial \lambda}{\partial y} \tag{5}*$$

$$w = w^\circ + \frac{1}{2\alpha_3^2}\frac{\partial \lambda}{\partial z} \tag{6}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{7}$$

The natural boundary conditions are

$$n_x \lambda \delta u = n_y \lambda \delta v = n_z \lambda \delta w = 0 \tag{8},$$

where δu, δv, and δw are the first variation of velocity components, and n_x, n_y, and n_z are the outward positive unit normals in the subscripted directions.

From Equations (4) through (7), an expression for λ may be obtained as

---

* Note that $\alpha_1 = \alpha_2$ has been assumed.

$$\frac{\partial^2\lambda}{\partial x^2} + \frac{\partial^2\lambda}{\partial y^2} + \left(\frac{\alpha_1^2}{\alpha_3^2}\right)\frac{\partial^2\lambda}{\partial z^2} = -2\alpha_1^2\left(\frac{\partial u^\circ}{\partial x} + \frac{\partial v^\circ}{\partial y} + \frac{\partial w^\circ}{\partial z}\right) \qquad (9).$$

Equation (9) along with boundary conditions as given in Equation (8) can now be solved for $\lambda(x,y,z)$, which in turn is used in Equations (4), (5), and (6) to estimate the adjusted values of the velocity components.

The adjustment procedures under the continuity condition as a strong constraint, as described above, lead to working equations which are in principle easy to apply. These procedures with Equations (3) through (8) have been evaluated for small regions, such as the San Francisco Bay area, by Dickerson (1978) and Sherman (1978). For the global wind field adjustment problem, the validity of these procedures has yet to be completely evaluated. On the global scale, it may be required that the wind field not only satisfy the continuity relation but also retain other observed kinematic properties. One of the important kinematic properties of the wind is its vorticity. In such cases another constraint defining vorticity of the wind may be introduced into Equation (3) through a second Lagrange multiplier λ_2. Consider that it is required to retain only the vertical component of vorticity, ζ_z°, estimated from the observed velocity components. Then the second constraint may be expressed as

$$\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} = \zeta_z^\circ = \frac{\partial v^\circ}{\partial x} - \frac{\partial u^\circ}{\partial y} \quad \text{or} \quad \frac{\partial (v-v^\circ)}{\partial x} - \frac{\partial (u-u^\circ)}{\partial y} = 0 \qquad (10).$$

The functional as defined by Equation (3) with the continuity equation and vorticity, Equation (10), as strong constraints leads to the following Euler-Lagrange equations

$$u = u^\circ + \frac{1}{2\alpha_1^2}\frac{\partial\lambda_1}{\partial x} - \frac{1}{2\alpha_1^2}\frac{\partial\lambda_2}{\partial y} \qquad (11),$$

$$v = v^\circ + \frac{1}{2\alpha_1^2}\frac{\partial\lambda_1}{\partial y} + \frac{1}{2\alpha_1^2}\frac{\partial\lambda_2}{\partial x} \qquad (12),$$

$$w = w^\circ + \frac{1}{2\alpha_3^2}\frac{\partial \lambda_1}{\partial z} \tag{13},$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0 \tag{14},$$

$$\frac{\partial (v-v^\circ)}{\partial x} - \frac{\partial (u-u^\circ)}{\partial y} = 0 \tag{15},$$

with natural boundary conditions

$$n_x \lambda_1 \delta u - n_y \lambda_2 \delta u = 0 \tag{16},$$

$$n_x \lambda_2 \delta v + n_y \lambda_1 \delta v = 0 \tag{17},$$

$$n_z \lambda_1 \delta w = 0 \tag{18}.$$

λ_1 and λ_2 are the Lagrange multipliers for the continuity and vorticity constraints, respectively, and the following equations for λ_1 and λ_2 may be obtained from Equations (11) through (15).

$$\frac{\partial^2 \lambda_1}{\partial x^2} + \frac{\partial^2 \lambda_1}{\partial y^2} + \left(\frac{\alpha_1^2}{\alpha_3^2}\right)\frac{\partial^2 \lambda_1}{\partial z^2} = -2\alpha_1^2 \left(\frac{\partial u^\circ}{\partial x} + \frac{\partial v^\circ}{\partial y} + \frac{\partial w^\circ}{\partial z}\right) \tag{19},$$

$$\frac{\partial^2 \lambda_2}{\partial x^2} + \frac{\partial^2 \lambda_2}{\partial y^2} = 0 \tag{20}.$$

Equations (19) and (20) along with the boundary conditions (Equations 16 - 18) can be solved for λ_1 and λ_2 which can in turn be substituted in Equations (11) to (13) to estimate the adjusted values of the velocity components.

This variational formulation is applicable to those cases in which only the observed vertical component of the vorticity is to be retained. Similarly, cases constraining all three components of vorticity may be formulated. Such a formulation not only leads to more complicated equations which are more difficult to solve, but it also involves vorticity terms which require observed values of the vertical velocity. In practive, the vertical velocities are

rarely measured at meteorological stations. Therefore, it leads to considerable uncertainty and difficulty in calculating the observed x- and y-direction vorticities.

Any or all of the fundamental equations describing the state of the atmosphere could be used as constraints in a variational sense. However, only Achtemeier (1972) has attempted to treat the observed data so they satisfy the complete set of equations comprising the mathematical model for the atmosphere, and he did that based on the variational method. This is probably the best approach in treating the limited experimental data obtained every twelve hours, and if applied, it would yield sets of "data" which, in terms of the mathematical model used, consistently describe the state of the atmosphere at discrete points in time.

To summarize, the four procedures represented by (a) Equation (1) alone; (b) Equations (4) through (9); (c) Equations (11) through (20); and (d) the completely constrained formulation represent a comprehensive analysis of global wind field adjustment procedures. Only Procedure (b) has been studied in this research. The computer program to solve that set of equations is presented in the Appendix. The program is written in SL/1 for use on the NASA LaRC STAR computer.

Temporal Interpolation of Meteorological Parameters

The preceding analyses deal with the meteorological configuration based on observations at a specific time. Generally, observations are only available every 12 hours while data describing the state of the atmosphere may be required at the intermediate time periods. As mentioned previously, transport modeling efforts frequently require such information at 30 minute intervals. In this section, a procedure to achieve this will be described. It is an extension of the mass-consistent wind field methodology presented in the previous

section. (These temporal interpolation studies may also be applicable to the data reduction procedure for the MAPS instrument which requires the temperature (to within 2°K) and water vapor profiles to determine the CO concentration. Again, since the data are only available at 12-hour intervals, the projection of these meteorological variables at the intermediate times at locations under the Space Shuttle track are required. The procedure that is suggested could also be applied to those requirements.)

The variational formulation for spatial adjustment is applicable to the observed data available at one time level. However, a similar formulation may be extended to any time level, and interpolation of data between two time levels may be obtained in such a way that the interpolated data satisfy the given constraint(s). Consider that observed values of wind field data are given for time levels t_n and t_{n+1}. It is required to estimate interpolated values at time t where $t_n < t < t_{n+1}$. If linear interpolation between time levels is assumed, then the functional in this case may be written as

$$I^* = \int_V \int_{t_n}^{t_{n+1}} [\alpha_1^2 (U - U^*)^2 + \alpha_1^2 (V - V^*)^2 + \alpha_3^2 (W - W^*)^2 + \lambda \left(\frac{\partial U}{\partial x} + \frac{\partial V}{\partial y} + \frac{\partial W}{\partial z}\right)] \, dt dV \tag{21}$$

where

$$U = \frac{u - u_n}{t - t_n} \quad \text{and} \quad U^* = \frac{u_{n+1} - u_n}{t_{n+1} - t_n} \tag{22},$$

with similar definitions for V and W. It can be shown that a simple transformation on the velocity components can reduce Equation (21) to a functional similar to that of Equation (3) from which Euler-Lagrange equations and boundary conditions are obtained which adjust the interpolated wind field so that the continuity equation is automatically satisfied. Thus, the wind field proceeds smoothly from the configuration at t_n to that at t_{n+1}. Equation (21) and the

associated conditions apply to linear temporal interpolation, but non-linear interpolation schemes are readily handled. Finally, the additional constraint of vorticity can also be incorporated.

Decoding the NMC Data Tape

The National Meteorological Center Data tape includes considerable data that are not necessary for interpretation of the OSTA-1/MAPS experimental data. In addition, separate tapes are used for the Soutern and Northerm Hemispheres and these data must be merged. Finally, the data must be decoded for use by the MAPS Science Team. A substantial effort was expended in this phase of the project.

Weather data tapes produced by the National Meteorological Center have been coded in a binary format not easily utilized by most computer programming languages. For this reason, a preliminary procedure must be performed on the data, to transform it into a standard EBCDIC format. (See the Appendix for a description of the data on the NMC tape.) DECODE is a PL/1 program which converts the NMC binary data to their EBCDIC equivalents. The tape format is also modified so that only certain desired fields are processed and retained. A program listing of DECODE is also contained in the Appendix and has been designed for use on the IBM 370 at the University of Kentucky.

DECODE works by manipulating PL/1 data structures. Data are read into a structure of bit strings. Then, using a subroutine, the binary data are converted, first to fixed binary numbers, then to EBCDIC characters for output to tape. Conversions are done based on the equations given in Office Notes 84 and 184 from the National Meteorological Center (see the Appendix for copies of these). A selection is made so that only the desired data are processed. All included data are of type 8. Data are included for the following conditions:

1. Q = 1, height with respect to mean sea level geopotential

(at 12 levels),

2. Q = 16, atmospheric temperature in degrees Kelvin (12 levels),
3. Q = 48, u-component of wind (12 levels),
4. Q = 49, v-component of wind (12 levels), and
5. Q = 88, relative humidity (6 levels).

A sort is then done to sequence the data in the proper order for merging into longer records. The input records contain all the data for 5,365 points on a southern or northern hemispheric grid of 145 x 37. Two input records are required to store all 10,730 of the global values for each time period of a two time-period day, with 15 days making up a complete tape. Alternate data values have been omitted, leaving 2 grids of 73 x 19, which corresponds to a 5° x 5° gridding. The output records contain data for 2,774 global points in a west to east, south pole to north pole order, with the sort field values included for identification purposes. In this manner, selected data from a full time period can be contained in a single record. Data fields (and their formats) included in the output records are

DATA-TYPE	F(14)	type of surface;
C1	F(14)	value of surface levels;
Y	F(14)	year;
M	F(14)	month;
D	F(14)	day;
I	F(14)	time period;
DATA(1387)	F(9)	data values from the southern grid; and
DAT2(1387)	F(9)	data values from the northern grid.

The records are sorted by year, month, day, time period, type of surface, and value of surface levels in descending order. The type of data is sorted in the following order: 48, 49, 1, 16, 88.

DECODE has been designed for maximum ease of utility by the end-user. Only one modification must be made prior to submission of the job with a new tape. JCL cards describing the input and output datasets must be prepared for each run of the program. Input (the NMC tape) is defined as SYSUT1, illustrated by the following example:

```
//SYSUT1 DD UNIT = 3400-5, VOL = SER = tape-number, DISP = (OLD,PASS),
//       DSN = name.of.dataset.on.tape,LABEL = (file-number,SL)
```

Output records may be sent to either disk or tape. However, the JCL for tape is simpler to prepare since the output records are longer than a track and therefore would require VBS records on disk. Be sure to specify a large enough SPACE parameter when using disk. A sample JCL definition for output on tape follows.

```
//GO.TAPE DD UNIT = 3400-5, VOL = SER = tape-number, DISP = (,PASS),
//        DCB = (RECFM = F,LRECL = 25050,BLKSIZE = 25050),
//        DSN = name.of.dataset.on.tape,LABEL = (file-number,SL)
```

Based on partial tape runs done on the University of Kentucky Computing Center's IBM 370/165 Model II computer running OS/MVT, a projected cost estimate may be made with some degree of accuracy. Since a load module will be used for the production runs, compilation costs for this projection can be ignored. Processing only the 2 time periods of the first day took 55 CPU-seconds of which 3 seconds can be attributed to compile time. Using the remaining 52 seconds of execution time, the approximate cost is $7.52 for decoding the first day. This is based on a $525 per hour charge (the discounted rate for deferred priority). Multiplying by the 15 days on each tape makes each tape cost about $113 to decode. It should be understood that this is only an approximation, and the actual cost will depend on various circumstances at

the time the job is run. Other computer installations use different charging algorithms, so costs will vary even more if run on a different machine.

Interpolation of the NMC Data for OSTA-1/MAPS

The NMC data as decoded provide horizontal winds, temperature, and dew point, all at the pressure levels of 1000, 850, 700, 500, 400, 300, 250, 200, 150, 100, 70, and 50 mb. An interpolation scheme was developed to represent these data in an alternate form. This scheme provides interpolated values for a 5° Longitude x 5° Latitude grid and a non-dimensional vertical coordinate which has $\rho = 0$ at the surface and $\rho = 1$ at the tropopause. The data are provided at $\rho = 0$, 0.0333, 0.0666, 0.10, 0.20, 0.40, 0.60, 0.80, and 1.0. This gridding provides a finer structure within the planetary boundary layer.

With the data unpacked from the NMC tapes consistent with the requirements, the data can then be interpolated for a geometric gridding. The mixing layer depth will also be calculated at each grid point and will use the 1-dimensional planetary boundary layer model of Yamada and Mellor (1975). Those researchers have provided a version of their program which is being modified for the present purposes.

Temporal and Spatial Interpolation of CO Data

The CO measurements from the OSTA-1/MAPS experiment on board the second Space Shuttle flight will correspond to mid-troposphere concentrations below the orbit path. These data will be from 38°S Latitude to 38°N Latitude and represent variations in both space and time. Thus, it is necessary to interpolate the concentration-time-position data at the desired times. The following treatment would accomplish this.

1) Transform the individual 1-second signals to correspond to non-overlapping, discrete orbiter track lengths of 10.37

nautical miles.

2) The information from item (1) represents a concentration at some time t_k, and some position $\underset{\sim}{r}_j$. The position is, of course, expressible in terms of latitude and longitude.

3) In order to construct an entire concentration field at some time t_n, interpolation in both space and time is required using as much of the data as seems reasonable. Define the following.

$c_i^n \equiv$ Concentration at time t_n and position $\underset{\sim}{r}_i$

$f_{ij}^k \equiv$ Weighting factor for temporal and spatial interpolation scheme

$r_{ij} \equiv$ Distance between positions $\underset{\sim}{r}_i$ and $\underset{\sim}{r}_j$

$t_{kn} \equiv t_k - t_n$

$q \equiv$ Total number of locations which are considered to influence interpolated value

$s \equiv$ Total number of time intervals which are considered to influence the interpolated value.

For the present case, $s = q$.

Then, the interpolated concentration is simply

$$c_i^n = \sum_{k=1}^{s} \sum_{j=1}^{q} f_{ij}^k \, c_j^k \qquad (23).$$

If one considers all of the data, this could be simplified to

$$c_i^n = \sum_{j=1}^{q} f_{ij}^j \, c_j^j \qquad (24).$$

4) Now the crucial question is, What should be used for f_{ij}^k? There are a few papers on this topic for aerometric data, and one scheme used is an inverse square dependence for

spatial interpolation. Nobody has apparently reported both spatial and temporal interpolation. Nevertheless, one plan to start with is the inverse square dependence for both space and time; i.e.,

$$f_{ij}^{k} = \frac{r_{ij}^{-2}}{\sum_{j=1}^{q} r_{ij}^{-2}} \frac{t_{kn}^{-2}}{\sum_{k=1}^{s} t_{kn}^{-2}} \tag{25}$$

In this manner, f_{ij}^{k} is of course normalized, Other weighting schemes could be tried.

5) The procedure in Items (1) - (4) provides interpolated values over a 10.37 nautical mile region. Since that is too much data to handle reasonably, these values could be averaged over 1° x 1°, 2.5° x 2.5°, or 5° x 5° longitude-latitude intervals. Then these data could be presented in digitized form as well as contours. Contours will be most helpful to see large scale plumes.

This procedure is easily coded, and the weighting function could be varied to evaluate its effect. Furthermore, several times could be done at once, and one may be able to take advantage of the data format, and calculate $\sum_{j=1}^{q} r_{ij}^{-2}$ just once.

CONCLUDING REMARKS

The procedures discussed in this final report will be used to assist in the analysis and interpretation of the CO observations obtained from the OSTA-1/MAPS Experiment which will be aboard the second Space Shuttle flight scheduled for November, 1981. Specifically, DECODE will be used to provide meteorological data from the NMC data tapes in a format useable by the MAPS Science Team. This will include u, v, T, and water vapor concentration data

for the flight period in either a pressure coordinate or a dimensionless terrain/tropopause following coordinate. In addition, an interpolation procedure for the OSTA-1/MAPS CO observation data has been proposed that includes a spatial and temporal weighting function.

A considerable effort has been expended on objective analysis procedures to establish global mass-conservative wind fields from the NMC data. The variational calculus procedure with the equation of continuity as a strong constraint has been coded and is in the final stages of evaluation. Novel adjustment procedures which incorporate vorticity preservation as an additional strong constraint and temporal interpolation during adjustment have also been presented. However, these procedures have not been evaluated. Research is continuing in this overall area of wind field adjustment techniques (Kitada et al., 1981).

REFERENCES

Achtemeier, G. L. (1972) "Variational Initialization of Atmospheric Fields - A Quasi-Geostrophic Diagnostic Model", Ph.D. Dissertation, Tallahassee, Florida State University.

Achtemeier, G. L. (1975) Mon. Wea. Rev., 103, 1989, On the Initialization Problem: A Variational Adjustment Method.

Carmichael, G. R. and Peters, L. K. (1981) Application of the Sulfur Transport Eulerian Model (STEM) to a SURE Data Set, 12th Int'l. Tech. Meeting on Air Pollution Modeling and Its Application, Palo Alto, CA, August 25-28.

Dickerson, M. H. (1978) J. Appl. Meteor., 17, 241-253, MASCON - A Mass Consistent Atmospheric Flux Model for Regions with Complex Terrain.

Endlich, R. M. (1967) J. Appl. Meteor., 6, 837, An Iterative Method for Altering the Kinematic Properties of Wind Fields.

Goodin, W. R., McRae, G. V., and Seinfeld, J. H. (1980) J. Appl. Meteor., 19, 98-108, An Objective Analysis Technique for Constructing Three-Dimensional Urban-Scale Wind Fields.

Kitada, T., Kaki, A., Ueda, H., and Peters, L. K. (1981) Estimation of Vertical Air Motion from Limited Horizontal Wind Data - A Numerical Experiment, Atmos. Environ., submitted for publication.

Liu, C. Y. and Goodin, W. R. (1976) Mon. Wea. Rev., 104, 784-792, An Iterative Algorithm for Objective Wind Field Analysis.

Peters, L. K. and Jouvanis, A. A. (1979) Atmos. Environ., 13, 1443-1462, Numerical Simulation of the Transport and Chemistry of CH_4 and CO in the Troposphere.

Sasaki, Y. (1958) J. Meteor. Soc. Japan, 36, 77-88, An Objective Analysis Based on the Variational Method.

Sherman, C. A. (1978) J. Appl. Meteor., 17, 312-319, A Mass Consistent Model for Wind Fields over Complex Terrain.

Stephens, J. J. (1970) J. Appl. Meteor., 9, 732, Variational Initialization with the Balance Equation.

Washington, W. M. and Duquet, R. T. (1963) "An Objective Analysis of Stratospheric Data by Sasaki's Method", Department of Meteorology, Pennsylvania State University, 23 pages.

Yamada, T. and Mellor, G. (1975) J. Atmos. Sci., 32, 2309-2329, A Simulation of the Wangara Atmospheric Boundary Layer Data.

APPENDIX

FILE PAMLAN*

```
/set,pamlam
/copyshf(pamlam)
1     MODULE-WINDAD.IS(UNIT1=COEFMTRX,UNIT2=SOLMTRX,UNIT3=
      ADJMTRX,UNIT6=OUTPUT,UNIT5=INPUT)-BEGIN-MAIN;
   $XREF;
   $TITLE('WINDFIELD-MODULE - PL/RARIG - SEPTEMBER 2, 1981');
  /*    SOURCE = PAMLAM                                    */
  /*    DATA = LAMDATA                                     */
  /*    JCL = PAMCIL                                       */
  /*=======================================================*/
      /* THIS PROGRAM ADJUSTS OBSERVED WIND FIELD TO MAKE
      IT MASS-CONSERVATIVE WIND FIELD. ADJUSTING TECHNIQUE
      IS BASED ON VARITIONAL FORMULATION. SIMULTANEOUS
      EQUATIONS FOR LAMDA TERMS ARE SOLVED USING SUCCESSIVE
      OVER RELAXATION (S-O-R) ITERATION METHOD. FILES NAMED AS
      COEFMTRX, SOLMTRX, AND ADJMTRX HAVE NESSASARY
      ESTIMATED COEFFICIENTS. THESE FILES WILL BE READ IN THIS
      PROGRAM.  */
      /* THE FOLLOWING SECTION NEEDS TO BE CHANGED FOR DIFFERENT
      GRID SYSTEM. XTRA1 TO XTRA7 AND ISM1 TO ISM7 ARE
      PAGING PARAMETERS. AL1 IS WEIGHTING FACTOR FOR U VELOCITY
      COMPONENT.                                          */
```


* Anyone interested in this program should contact L. K. Peters at the University of Kentucky.

```
/*     MODIFICATION HISTORY                                    */
/*-------FEBRUARY-18,1981-------------------------------------*/
/*-------COMMENTING REGARDING F3 IMPLICIT I/O ----------------*/
/*------- FILE-6 IS NOW OUTPUT!-------------------------------*/
/*-----------------------------------------------------------*/
/*-----------------------------------------------------------*/
      /*--BEGINING OF THE SECTION ----------*/
      LITERALLY "5.0" DYY;
      LITERALLY "72" NUM_EW;
      LITERALLY "37" NUM_NS;
      LITERALLY "9"NUM_ALT;
      LITERALLY "10"NUM_ALTP1; /*-- NUM_ALTP1= NUM_ALT+1 */
      LITERALLY "23976"EW_NS_ALT;
      LITERALLY "33576" XTRA1;
      LITERALLY "15992" XTRA2;
      LITERALLY "17584" XTRA3;
      LITERALLY "24512" XTRA4;
      LITERALLY "51152" XTRA5;
      LITERALLY "11192" XTRA7;
      LITERALLY "768" ISM1;
      LITERALLY "640" ISM2;
      LITERALLY "128" ISM3;
      LITERALLY "256" ISM4;
      LITERALLY "256" ISM5;
      LITERALLY "256" ISM7;
/*LFE LITERALLY "0.01"ALL;  LFE*/
      LITERALLY "3.14159265358979"PI;
      /* NUM_EW=NUMBER OF GRIDS IN EAST-WEST DIRECTION.
      NUM_NS=NUMBER OF GRIDS IN NORTH-SOUTH DIRECTION.
      NUM_ALT=NUMBER OF GRIDS IN ALTITUDE-DIRECTION.
      EW_NS_ALT=TOTAL NUMBER OF GRIDS.
      TOLERANCE=TOLRANCE IN ITERATIONS.
      RLXFAC=RELAXATION FACTOR-IN SUCCESIVE-OVER RELAXTION
      METHOD. */
      /* END OF THE SECTION  */
      REAL VECTOR [15] ARRAY(EW_NS_ALT) TX;
      COMMON CT(TX,TDUMP1);
      REAL VECTOR [XTRA1] TDUMP1;
      REAL VECTOR[2644] MYALT;
      REAL VECTOR [5] ARRAY(EW_NS_ALT) ADJX;
      COMMON CA(ADJX,TDUMP3);
      REAL VECTOR [XTRA7] TDUMP3;
      REAL VECTOR [EW_NS_ALT] ARRAY(13) SSX;
      COMMON CSS(SSX,TDUMP4);
      REAL VECTOR [XTRA2] TDUMP4;
      REAL VECTOR [15] ARRAY(EW_NS_ALT) TLAMDA;
      COMMON B1(TLAMDA,TDUMP11);
      REAL VECTOR [XTRA1] TDUMP11;
      REAL VECTOR [EW_NS_ALT] S_DIV;
      COMMON B2(S_DIV,TDUMP5);
      REAL VECTOR [XTRA3] TDUMP5;
```

```
      REAL VECTOR [5] ADJ;
      REAL VECTOR [NUM_EW] ARRAY(NUM_NS,NUM_ALT) ALAMDA,BLAMDA;
      COMMON B3(BLAMDA,ALAMDA,TDUMP6);
      REAL VECTOR [XTRA3] TDUMP6;
      REAL VECTOR [NUM_EW] ARRAY(NUM_NS,NUM_ALTP1)
           U,V,W,D,ADJUSTU,ADJUSTV,
      ADJUSTW;
      COMMON B4(U,V,W,D,TDUMP7);
      REAL VECTOR [XTRA4] TDUMP7;
      COMMON B5(ADJUSTU,ADJUSTV,ADJUSTW,TDUMP8);
      REAL VECTOR [XTRA5] TDUMP8;
      REAL VECTOR [EW_NS_ALT] ARRAY(13) VEL;
      COMMON B6(VEL,TDUMP9);
      REAL VECTOR [XTRA2] TDUMP9;
      REAL VECTOR [NUM_ALTP1] Z;
      INITIAL (7=\0.0,0.0333334,0.0333334,0.0333334
      ,0.1,0.2,0.2,0.2,0.2,0.2\);
      REAL ALAMDA,T,ERROR,777,THETA,TH,CON,MAXDIV,MINDIV,TOLERANCE,RLXFAC;
      INTEGER IMINDIV,IMAXDIV;
      INTEGER NOT_CONV,I,J,K,NL,I1,I2,I3,I4,J1,J2,K2,
/*LEEK3,ITERATION,K1,IW,E1,E2,E3,E4,E5,NUM_ITER,LEE*/
      K3,ITERATION,K1,IW,E1,E2,E3,E4,E5,NUM_ITER,MAX_ITER,
      E6,E7,E8,E9,E10,MMMM,JJ,NS_INCR,ALT_INCR,EW_OUT;
      REAL VECTOR [3333] WLAMDA,WLAMDA1;
      REAL VECTOR [9] KLAMDA;
      REAL MAXLAMDA,MINLAMDA,AL1;
/*****************************************************************/
/*****************************************************************/
/*****************************************************************/
/*****************************************************************/
/*****************************************************************/
$PAGE;
      PROCEDURE MAIN;
/*LEE*/
      WRITE_UNIT(6)#'1BEGIN MAIN PROCEDURE'#;
      READ_UNIT (5)
           (RLXFAC,TOLERANCE,AL1,MAX_ITER,NS_INCR,ALT_INCR,
           EW_OUT)
      #3(F9.0/),3(I9/),I9#;
/*LEE*/
/*    ===PLR JANUARY 23 1981===*/
      IF EW_OUT > NUM_EW THEN
      WRITE_UNIT(6)(EW_OUT)#5X'USER WANTS'I5#;
      EW_OUT:=NUM_EW;
      END;
      OPENMAP('F1',TLAMDA(1)[1],ISM1,F1,'LARGE');
      OPENMAP('F2',S[1],ISM3,F2,'LARGE');
      OPENMAP('F3',BLAMDA(1,1)[1],ISM3,F3,'LARGE');
      OPENMAP('F4',U(1,1)[1],ISM4,F4,'LARGE');
      OPENMAP('F5',ADJUSTU(1,1)[1],ISM5,F5,'LARGE');
      OPENMAP('F6',VEL(1)[1],ISM2,F6,'LARGE');
```

```
        OPENMAP('COEFMTRX',TX(1)[1],ISM1,E8,'LARGE');
        OPENMAP('SOL_MTRX',SSX(1)[1],ISM2,E9,'LARGE');
        OPENMAP('AD_MTRX',AD_IX(1)[1],ISM7,F10,'LARGE');
/*LEE -WRITE UNIT(6)(E1,E2,E3,E4,E5,E6,E8,E9
        ,E10)#'OPEN', 9(3X I2)#;LEE*/
        WRITE UNIT(6)(E1,E2,E3,E4,E5,E6,E8,E9
        ,E10)#' OPEN',-9(3X I2)#;
        NUM_ITER := 0;
/*LEE- WRITE UNIT(6)
        (DYY,NUM_EW,NUM_NS,NUM_ALT,AL1,TOLERANCE,RLXFAC)#'DYY =',
        F5.2,
        'NUM_EW =',I3,'NUM_NS = ',I3,'NUM_ALT =',I3,
        'AL1 = ',F7.3,'TOLERANCE = ',F8.4,
        'RLXFACTOR = ',F6.3#;LEE*/
        WRITE UNIT(6)
        (DYY,NUM_EW,NUM_NS,NUM_ALT,AL1,TOLERANCE,RLXFAC,
        MAX_ITER)#' DYY = ',F5.2,
        'NUM_EW =',I3,'NUM_NS = ',I3,'NUM_ALT = ',I3,
        'AL1 = ',F7.3,'TOLERANCE = ',F8.4,
        'RLXFACTOR =',F6.3,' MAX_ITER =',I3#;
        CON := PI/180.0;
        K2 := NUM_ALT+1;
        /* INITIALIZING ALAMDA AND BLAMDA ARRAYS GIVING
        FIRST GUESS VALUES */
/* LAMBDA VALUES ARE FOUND ON PAGE FILE F3 */
/* WHEN THE FOLLOWING IMPLICIT I/O IS */
/* IS COMMENTED OUT, OLD VALUES FROM SAVED */
/* FILE F3 ARE NOT OVER-WRITTEN */
/*-----------------------------------------*/
        THETA:=90.0 + DYY;
        FOR J := 1 TO NUM_NS DO
        THETA := THETA-DYY;/*NOT NEEDED PLR-FER 17.31 */
/* UP THE INITIAL VALUES BY 1.0E+06 */
        FOR K := 1 TO NUM_ALT DO
        BLAMDA(J,K)[1:NUM_EW] := 2.0E+06;
        ALAMDA(J,K)[1:NUM_EW] := 2.0E+06;
        ENDF;
        ENDF;
        FOR I := 1 TO NUM_NS DO
        FOR K := 1 TO NUM_ALT DO
        FOR I := 1 BY 2 TO NUM_EW DO
        BLAMDA(J,K)[I] := 1.0E+06;
        ALAMDA(J,K)[I] := 1.0E+06;
        ENDF;
        ENDF;
        ENDF;
/* === END OF I/O === INSERT COMMENT HERE ==*/
/* === FLOAT APRIL 7TH === */
/* BLAMDA(5,5)[36] := 1.5E+06;
        ALAMDA(5,5)[36] := 1.5E+06; */
        FOR J := 1 TO NUM_NS DO
```

```
        WLAMDA[J] := BLAMDA(J,5)[36];
    ENDF;
/*LEE*/WRITE UNIT(6)#'OWLAMDA ARE:'#;/*LEE*/
    WRITE UNIT(6)(WLAMDA[1:NUM_NS])#5E14.7#;
    FOR K := 1 TO NUM_ALT DO
        KLAMDA[K] := BLAMDA(5,K)[36];
    ENDF;
/*LEE WRITE UNIT(6)(KLAMDA[1:NUM_ALT])#'KLAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[30:40])#'LAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[1],BLAMDA(5,5)[NUM_EW])
    #'LAMDA = ',2E14.7#;LEE*/
/*LEE*/WRITE UNIT(6)(KLAMDA[1:NUM_ALT])#' KLAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[30:40])#'LAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[1],BLAMDA(5,5)[NUM_EW])
    #'LAMDA = ',2E14.7#;/*LEE*/
    FOR I := 1 TO 13 DO
        VEL(I)[1:EW,NS_ALT] := 0.0;
    ENDF;
    /*_INITIALIZING WIND FIELD U,V,W AND DENSITY D ARRAYS */
    THETA := -90.0-DYY;
    FOR J := 1 TO NUM_NS DO
        THETA := THETA+DYY;
        TH := THETA*CON;
        FOR K := 1 TO K2 DO
            U(J,K)[1:NUM_EW] := -10.0;
            V(J,K)[1:NUM_EW] := -3.0;
            W(J,K)[1:NUM_EW] := 0.0;
            D(J,K)[1:NUM_EW] := 1.0;
        ENDF;/*LOOP ON K LEE*/
    ENDF;/*LOOP ON J LEE*/
    CALL DIVRGNCE;
    CALL SOR;
/*  AUG 27TH 1981 */
/*_PRINT OUT 10 LARGEST AND */
/*_SMALLEST DIVS PER LAYER...SELED RETURNS */
/*_#_OF UNSUCCESSFUL COMPARISONS! */
/*                                */
    FOR K := 1 TO NUM_ALT DO
        NL := 0;
        WRITE UNIT(6)(K)#1H1,///,40X,'ALTITUDE =',I3,///#;
        FOR J := 1 TO NUM_NS DO
            FOR I := 1 TO NUM_EW DO
                NL := NL+1;
                MYALT[NL] := DIV[(K-1)*72+(J-1)*648+I];

        ENDF;
    ENDF;
    MYALT[1:2664] := ABS(MYALT[1:2664]);
    FOR JJ := 1 TO 20 DO
        MAXDIV := MAX(MYALT);
        MINDIV := MIN(MYALT);
```

```
      IMAXDIV:=SELEQ(MYALT,MAXDIV)+1;
      IMINDIV:=SELEQ(MYALT,MINDIV)+1;
      J1:=IMAXDIV.DIV.72+1;
      I1:=IMAXDIV-((J1-1)*72);
      J2:=IMINDIV.DIV.72+1;
      I2:=IMINDIV-((J2-1)*72);
      WRITE UNIT(6)(K,MAXDIV,IMAXDIV,I1,J1,MINDIV,IMINDIV,I2,J2)
   #5X,I5,E15.5,3I5,E15.5,3I5#;
      MYALT[IMAXDIV]:=0.0;
      MYALT[IMINDIV]:=0.0;
   END F;
END F;
   CLOSE('F1',E1);
   CLOSE('F2',E2);
   CLOSE('F3',E3);
   CLOSE('F4',E4);
   CLOSE('F5',E5);
   CLOSE('F6',E6);
   CLOSE('COEFMTRX',E8);
   CLOSE('SOLMTRX',E9);
   CLOSE('ADJMTRX',E10);
/*LEE   WRITE UNIT(6)(E1,E2,E3,E4,E5,E6,E8
      ,E9,E10)#'CLOSE', 9(3X)2)#;LEE*/
   WRITE UNIT(6)(E1,E2,E3,E4,E5,E6,E8
      ,E9,E10)#' CLOSE', -9(3X)2)#;
/*LEE*/ WRITE UNIT(6)#'1LEAVE MAIN PROCEDURE'#;/*LEE*/;
   END;/*MAIN -LEE*/
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
$PAGE;
   PROCEDURE DIVRGNCE;
/*  REVISED -JUNE 8,1981-  PL/RARIG                              */
/*  TREAT EDGE NODES FOR VEL(4,8,10)                             */
/*  K=1    GROUND EDGE                                           */
/*  K=9    TROPOPAUSE EDGE                                       */
/*-------------------------------------------------------------*/
/*LEE*/ WRITE UNIT(6)#'1BEGIN PROCEDURE DIVRGNCE'#;/*LEE*/
   /* THIS PROCEDURE ESTIMATES THE VALUE OF RIGHT HAND SIDE
   OF EQUATIONS. AT THE END IT CALCULATES THE DIVERGENCE
   OF OBSERVED OR ADJUSTED WIND FIELD. */
   NL:=0;
   FOR L:=1 TO NUM_NS DO
      FOR K:=1 TO NUM_ALT DO
         FOR I:=1 TO NUM_EW DO
            NL:=NL+1;
            I1:=I+1;
            I2:=I-1;
            IF I1=NUM_EW+1 THEN I1:=1;END;
```

```
            IF I2=0     THEN I2 := NUM_EW;END;
        VEL(1)[NL] := U(J,K)[I]*D(J,K)[I];
        VEL(2)[NL] := U(J,K)[I2]*D(J,K)[I2];
        VEL(3)[NL] := U(J,K+1)[I]*D(J,K+1)[I];
        IF K=1 THEN
            VEL(4)[NL] := (2*U(J,1)[I]-U(J,2)[I])*(2*D(J,1)[I]
            -D(J,2)[I]);
        END;
        IF K=9 THEN
            VEL(4)[NL] := (2*U(J,8)[I]-U(J,9)[I])*(2*D(J,8)[I]
            -D(J,9)[I]);
    END;
        IF((K>1).AND.(K<9))THEN
            VEL(4)[NL] := U(J,K-1)[I]*D(J,K-1)[I];
        END;
    END;/*LOOP ON I--LEE*/
  END;/*LOOP ON K  LEE*/
END;/*LOOP ON J--LEE*/
NL := 0;
FOR J := 1 TO NUM_NS DO
    FOR K := 1 TO NUM_ALT DO
        FOR I := 1 TO NUM_EW DO
            NL := NL+1;
            J1 := J+1;
            J2 := J-1;
            IF J1=NUM_NS+1 THEN J1 := NUM_NS-1;END;
            IF J2=0 THEN J2 := 2;END;
            VEL(5)[NL] := V(J1,K)[I]*D(J1,K)[I];
            VEL(6)[NL] := V(J2,K)[I]*D(J2,K)[I];
            VEL(7)[NL] := V(J,K+1)[I]*D(J,K+1)[I];
            IF K=1
            THEN VEL(8)[NL] := (2*V(J,1)[I]-V(J,2)[I])*(2*D(J,1)[I]
            -D(J,2)[I]);
            END;
            IF K=9
            THEN VEL(8)[NL] := (2*V(J,8)[I]-V(J,9)[I])*(2*D(J,8)[I]
            -D(J,9)[I]);
            END;
    IF((K>1).AND.(K<9))THEN
            VEL(8)[NL] := V(J,K-1)[I]*D(J,K-1)[I];
        END;
    END;/*LOOP ON I--LEE*/
  END;/*LOOP ON K--LEE*/
END;/*LOOP ON J--LEE*/
NL := 0;
FOR J := 1 TO NUM_NS DO
    FOR K := 1 TO NUM_ALT DO
        FOR I := 1 TO NUM_EW DO
            NL := NL+1;
            VEL(9)[NL] := W(J,K+1)[I]*D(J,K+1)[I];
            IF K=1 THEN
```

```
            VEL(10)[NL] := (2*W(J,1)[I]-W(J,2)[I])*(2*D(J,1)[I]
      -D(J,2)[I]);
      ENDI;
      IF K=9 THEN
      VEL(10)[NL] := (2*W(J,8)[I]-W(J,9)[I])*(2*D(J,8)[I]
      -D(J,9)[I]);
   ENDI;
      IF((K>1).AND.(K<9))THEN
      VEL(10)[NL]:=W(J,K-1)[I]*D(J,K-1)[I];
      ENDI;
      VEL(11)[NL] := U(J,K)[I]*D(J,K)[I];
      VEL(12)[NL] := V(J,K)[I]*D(J,K)[I];
      VEL(13)[NL] := W(J,K)[I]*D(J,K)[I];
      ENDF;
   ENDF;
ENDF;
   /*CALCULATING RIGHT-HAND SIDE OF EQUATIONS*/
   S[1:EW_NS_ALT] := 0.0;
/*----BEGIN-COMMENT-HERE------*/
      FOR I := 1 TO 13 DO
   S[1:EW_NS_ALT] := S[1:EW_NS_ALT]+
      SSX(I)[1:EW_NS_ALT]*VEL(I)[1:EW_NS_ALT]
;
   ENDF;
/*----END-COMMENT HERE----*/
   /*CALCULATING DIVERGENCE OF WIND FIELD*/
   AL3 := -2.0*AL1*AL1;
   DIV[1:EW_NS_ALT] := S[1:EW_NS_ALT]/AL3;
   MAXDIV := MAX(DIV);
   MINDIV := MIN(DIV);
/*LEE   WRITE UNIT(6)(MAXDIV,MINDIV)#'MAXDIV = ',E14.7,
   'MINDIV = ',E14.7#;LEE*/
   WRITE UNIT(6)(MAXDIV,MINDIV)#'MAXDIV = ',E14.7,
   ' MINDIV = ',E14.7#;
BBB: WRITE UNIT(6)#'LEAVE PROCEDURE DIVRGNCE'#
   ENDP;/*DIVRGNCE LEE*/
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
/*************************************************************************/
$PAGE;
   PROCEDURE SOR;
      INTEGER JMOD,KMOD,IMOD;
/*LEE*/ WRITE UNIT(6)#'BEGIN PROCEDURE SOR'#;/*LEE*/
   /* THIS PROCEDURE IS A PROGRAM FOR A SUCCESSIVE OVER
   RELAXATION (S-O-R) ITERATION METHOD TO SOLVE SIMULTANEOUS
   EQUATIONS.FIRST VELOCITIES AND LAMDA TERMS ARE ARRANGED
   IN VECTOR FORMS,AND THEN S-O-R EQUATION IN GENERAL
   FORM IS USED FOR ITERATIONS. */
/*-----------------------------------------*/
```

```
        NUM_ITER := 0;
LOOP:---NL := 0;
--------/* INITIALIZING TLAMDA ARRAYS */
--------FOR I := 1 TO EW_NS_ALT DO
--------TLAMDA(I)[1:15] := 0.0;
--------ENDF;
-------/*LEE ----------NOT_CONV := 1;*/
--------NOT_CONV := 0;
--------FOR J := 1 TO NUM_NS DO
----------FOR K := 1 TO NUM_ALT DO
------------ FOR I := 1 TO NUM_EW DO
--------------NL := NL+1;
--------------I1 := I+1;
--------------I2 := I-1;
--------------I3 := I+2;
--------------I4 := I-2;
--------------J1 := J+1;
--------------J2 := J-1;
--------------K1 := K+1;
--------------K3 := K-1;
--------------IF I1=NUM_EW+1 THEN I1 := 1;ENDI;
--------------IF I2=0 ---THEN I2 := NUM_EW;ENDI;
--------------IF J1=NUM_NS+1 THEN J1 := NUM_NS-1;ENDI;
--------------IF J2=0 ---THEN J2 := 2;ENDI;
--------------IF K1=NUM_ALT+1 --THEN K1 := NUM_ALT;ENDI;
--------------IF K3=0 THEN K3 := 1;ENDI;
--------------IF I3=NUM_EW+2 THEN I3 := 2;ENDI;
--------------IF I3=NUM_EW+1 THEN I3 := 1;ENDI;
--------------IF I4=0 ---THEN I4 := NUM_EW;ENDI;
--------------IF I4=-1 ---THEN I4 := NUM_EW-1;ENDI;
--------------TLAMDA(NL)[1] := BLAMDA(J,K)[I];
--------------TLAMDA(NL)[2] := BLAMDA(J,K)[I1];
--------------TLAMDA(NL)[3] := BLAMDA(J,K)[I2];
--------------TLAMDA(NL)[4] := BLAMDA(J1,K)[I];
--------------TLAMDA(NL)[5] := BLAMDA(J2,K)[I];
--------------TLAMDA(NL)[6] := BLAMDA(J,K1)[I];
--------------TLAMDA(NL)[7] := BLAMDA(J,K3)[I];
--------------TLAMDA(NL)[8] := BLAMDA(J,K1)[I1];
--------------TLAMDA(NL)[9] := BLAMDA(J,K3)[I1];
--------------TLAMDA(NL)[10] := BLAMDA(J,K1)[I2];
--------------TLAMDA(NL)[11] := BLAMDA(J,K3)[I2];
--------------TLAMDA(NL)[12] := BLAMDA(J1,K1)[I];
--------------TLAMDA(NL)[13] := BLAMDA(J1,K3)[I];
--------------TLAMDA(NL)[14] := BLAMDA(J2,K1)[I];
--------------TLAMDA(NL)[15] := BLAMDA(J2,K3)[I];
--------------/* ENDING OF VECTORIZATION */
--------------/*BEGINING OF S-O-R METHOD */
--------------T := TLAMDA(NL).DOT.TX(NL);
--------------BLAMDA(J,K)[I] := ALAMDA(J,K)[I]+(RLXFAC*(S[NL]-T)/
------------TX(NL)[1]);
  /*  DIAGNOSTIC EVERY 10TH GUY    */
```

```
/*  MARCH 27,1981     PLR          */
      JMOD:=J.MOD.10;
      IMOD:=I.MOD.10;
      KMOD:=K.MOD.10;
    IF((JMOD=1).AND.(KMOD=1).AND.(IMOD=1)) THEN
      WRITE UNIT(6)
      (I,J,K,ALAMDA(J,K)[I],BLAMDA(J,K)[I],S[NL],
      T,TX(NL)[1])#3I5,5E15.5#;
    ENDI;
          IF((J=1).OR.(J=NUM_NS)) THEN
          IF(J=1) THEN JJ := 2;ENDI;
          IF(J=NUM_NS) THEN JJ := NUM_NS-1;ENDI;
          BLAMDA(J,K)[I] := SUM(BLAMDA(JJ,K)[1:NUM_EW])/NUM_EW;
          ENDI;
  /*  BUMP UP BY 1.0E+06 MAR 31     */
  /*  TRY A"FLOAT"  */
  /*      BLAMDA(5,5)[36] := 1.5E+06;  */
          IF(ALAMDA(J,K)[I]=0.0) THEN ALAMDA(J,K)[I] := 0.0001;ENDI;
          ERROR := 1.0-(BLAMDA(J,K)[I]/ALAMDA(J,K)[I]);
          IF ABS(ERROR)>TOLERANCE THEN NOT_CONV := NOT_CONV+1;ENDI;
        ENDF;
      ENDF;
    ENDF;
  /*---------TEMPORARY RHS PRINTOUT ----------
    ---PLR MARCH 12 1981-------*/
  /*  FOR J FROM 1 BY 100 TO FW_NS_ALT DO
      WRITE UNIT(6)(J,S[J])#5X,I5,5X,E15.5#;
    ENDF;  */
    FOR J := 1 TO NUM_NS DO
      WLAMDA[J] := BLAMDA(J,5)[36];
    ENDF;
  /*LEE*/WRITE UNIT(6)#'OWLAMDA ARE:'#;/*LEE*/
    WRITE UNIT(6)(WLAMDA[1:NUM_NS])#5E14.7#;
    FOR K := 1 TO NUM_ALT DO
      KLAMDA[K] := BLAMDA(5,K)[36];
    ENDF;
  /*LEE WRITE UNIT(6)(KLAMDA[1:NUM_ALT])#'KLAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[30:40])#'LAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[1],BLAMDA(5,5)[NUM_EW])
    #'LAMDA = ',2E14.7#;LEE*/
    WRITE UNIT(6)(KLAMDA[1:NUM_ALT])#' KLAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[30:40])#' LAMDA = ',5E14.7#;
    WRITE UNIT(6)(BLAMDA(5,5)[1], BLAMDA(5,5)[NUM_EW])
    #' LAMDA = ',2E14.7#;
    MMMM:=0;
    NL := 0;
    FOR J:=1TO NUM_NS DO
      FOR K:=1 TO NUM_ALT DO
        NL := NL+1;
      WLAMDA[NL] := MAX(BLAMDA(J,K));
        WLAMDA1[NL] := MIN(BLAMDA(J,K));
```

```
            IF (MMMM<3) THEN
            IF (WLAMDA1[NL]<0.0) THEN
/*LEE  WRITE UNIT(6)(J,K,WLAMDA1[NL])#'J = ',I3,'K = ',I3,
      'WLAMDA1 = ',E14.7#;LEE*/
            WRITE UNIT(6)(J,K,WLAMDA1[NL])#' J = ',I3,' K = ',I3,
            ' WLAMDA1 = ',E14.7#;
            MMMM := MMMM+1;
            ENDI;
            ENDI;
            IF((J=37).AND.(K=1)) THEN
/*LEE  WRITE UNIT(6)(J,K,WLAMDA1[NL])#'J = ',I3,'K = ',I3,
      'WLAMDA1 = ',E14.7#;LEE*/
            WRITE UNIT(6)(J,K,WLAMDA1[NL])#' J = ',I3,' K = ',I3,
            ' WLAMDA1 = ',E14.7#;
            ENDI;
            ALAMDA(J,K)[1:NUM_EW] := BLAMDA(J,K)[1:NUM_EW];
            ENDF;
            ENDF;
            MAXLAMDA := MAX(WLAMDA);
            MINLAMDA := MIN(WLAMDA1);
/*LEEWRITE UNIT(6)(MAXLAMDA,MINLAMDA)#'LAMDA = ','MAX = ',E14.7,
      'MIN = ',E14.7#;LEE*/
            WRITE UNIT(6)(MAXLAMDA,MINLAMDA)#'LAMDA = ','MAX = ',E14.7,
            ' MIN = ',E14.7#;
            NUM_ITER := NUM_ITER + 1;
/*LEE  WRITE UNIT(6)(NOT_CONV)#'NUMBER OF ELEMENTS NOT CONVED = '
      ,I8#;LEE*/
            WRITE UNIT(6)(NOT_CONV,NUM_ITER)
            #',I8,' ELEMENTS HAVE NOT CONVERGED AFTER',
            I8,' ITERATIONS.'#;
/*LEE  WRITE UNIT(6)(NUM_ITER)#'COUNTING OF ITERATIONS = ',I8#;LEE*/
/*LEE IF(NUM_ITER>MAX_ITER)THEN GO TO BBB;ENDI;LEE*/
            IF(NUM_ITER>MAX_ITER)THEN GO TO BBB;ENDI;
/*LEE  IF NOT CONV>1 THEN GO TO LC;ENDI;*/
            IF NOT CONV >0 THEN GO TO LC; ENDI;
/*LEE  WRITE UNIT(6)(ITERATION)#10,'NUMBER OF ITERATIONS = '
      ,I3#;LEE*/
            WRITE UNIT(6)(NUM_ITER)#'0',I8,' ITERATIONS BEFORE TERMINATION',
            'OF S-O-R SOLUTION'#;
            /* ENDING OF S-O-R METHOD */
            /* PRINTING OF LAMDA TERMS */
            /*      FOR J := 1 TO NUM_NS DO   */
/*LEE*/WRITE UNIT(6)#'OFINAL LAMBDA ARE:'#;/*LEE*/
            FOR J := 1 BY NS_INCR TO NUM_NS DO
            FOR K := 1 BY ALT_INCR TO NUM_ALT DO
            WRITE UNIT(6)(J,K)#10,'J = ',I3,'K = ',I3#;
            WRITE UNIT(6)(BLAMDA(J,K)[1:10])#5E14.7#;
            ENDF;
            ENDF;
            /* ADJUSTMENT OF OBSERVED WIND FIELD TO MAKE IT MASS-
               CONSERVATIVE  */
```

```
        NL := 0;
- ------FOR-I-:=-1-TO-NUM_NS-DO-----------------------------------------------------------------------
----------FOR K :=-1 -TO NUM_ALT DO-----------------------------------------------------------------
 ----------FOR J :=-1 -TO NUM_EW DO-----------------------------------------------------------------
----------NL:=-NL+1;-----------------------------------------------------------------------------
----------ADJ[1:5] :=-ADJX(NL)[1:5];-------------------------------------------------------------
----------I1 :=-I+1;-----------------------------------------------------------------------------
----------I2 :=-I-1;-----------------------------------------------------------------------------
----------J1 :=-J+1;-----------------------------------------------------------------------------
----------J2 := J-1;-----------------------------------------------------------------------------
----------K1 :=-K+1;-----------------------------------------------------------------------------
----------K3 := K-1;-----------------------------------------------------------------------------
----------IF (I1=NUM_EW+1) THEN I1 :=-1;ENDI;-----------------------------------------------------
----------IF (I2=0 --) THEN I2 := NUM_EW;ENDI;----------------------------------------------------
----------IF (J1=NUM_NS+1)-THEN J1 := NUM_NS-1;J2 := NUM_NS;ENDI;----------------------------------
----------IF (J2=0 --) THEN J2 := 1;ENDI;---------------------------------------------------------
----------IF (K1=NUM_ALT+1) THEN K1 := NUM_ALT;ENDI;-----------------------------------------------
----------IF (K3=0 --) THEN K3 := 1;ENDI;----------------------------------------------------------
----------ADJUSTU(J,K)[I] := U(J,K)[I]+ADJ[1]*(BLAMDA(J,K)[I1]-----------------------------------
----------BLAMDA(J,K)[I2])-ADJ[2]*(BLAMDA(J,K1)[I]-------------------------------------------------
----------BLAMDA(J,K3)[I]);-------------------------------------------------------------------------
----------ADJUSTV(J,K)[I] := V(J,K)[I]+ADJ[3]*(BLAMDA(J1,K)[I]--------------------------------------
----------BLAMDA(J2,K)[I])-ADJ[4]*(BLAMDA(J,K1)[I]--------------------------------------------------
----------BLAMDA(J,K3)[I]);-------------------------------------------------------------------------
----------ADJUSTW(J,K)[I] := W(J,K)[I]+ADJ[5]*(BLAMDA(J,K1)[I]---------------------------------------
----------BLAMDA(J,K3)[I]);-------------------------------------------------------------------------
--------ENDF;-------------------------------------------------------------------------------------
--------ENDF;-------------------------------------------------------------------------------------
------ENDF;-------------------------------------------------------------------------------------
--------/* PRINTING OF ADJUSTED WIND FIELD */-------------------------------------------------------
--------/*--------FOR J := 1 NUM_NS DO --*/----------------------------------------------------------
--------FOR J := 1 BY NS_INCR TO NUM_NS DO-------------------------------------------------------
----------FOR K := 1 BY ALT_INCR TO NUM_ALTP1 DO---------------------------------------------------
------/*IFF-------WRITE UNIT(6)(J,K)#1H0,'J = ',I3,'K = ',I3#;-----------------------------------------
----------WRITE UNIT(6)(U(J,K)[1:10],ADJUSTU(J,K)[1:10],-------------------------------------------
----------V(J,K)[1:10],ADJUSTV(J,K)[1:10],---------------------------------------------------------
----------W(J,K)[1:10],ADJUSTW(J,K)[1:10])#5E14.7#;IFF*/-------------------------------------------
----------WRITE UNIT(6)(J,K)#'0J = ',I3,'K = ',I3#;-------------------------------------------------
--------WRITE UNIT(6)(U(J,K)[1:EW_OUT])#' UNADJUSTED U ARE',5E14.7#;-----------------------------
--------WRITE UNIT(6)(ADJUSTU(J,K)[1:EW_OUT])#' ADJUSTED U ARE',5E14.7#;-------------------------
--------WRITE UNIT(6)(V(J,K)[1:EW_OUT])#' UNADJUSTED V ARE',5E14.7#;-----------------------------
--------WRITE UNIT(6)(ADJUSTV(J,K)[1:EW_OUT])#' ADJUSTED V ARE',5E14.7#;-------------------------
--------WRITE UNIT(6)(W(J,K)[1:EW_OUT])#' UNADJUSTED W ARE',5E14.7#;-----------------------------
--------WRITE UNIT(6)(ADJUSTW(J,K)[1:EW_OUT])#' ADJUSTED W ARE',5E14.7#;-------------------------
----------ENDF;-------------------------------------------------------------------------------------
--------ENDF;-------------------------------------------------------------------------------------
--------/* CALCULATING DIVERGENCE OF ADJUSTED WIND FIELD */-------------------------------------
--------FOR J := 1 TO NUM_NS DO-------------------------------------------------------------------
----------FOR K := 1 TO NUM_ALT DO-----------------------------------------------------------------
----------U(J,K)[1:NUM_EW1] := ADJUSTU(J,K)[1:NUM_EW1];--------------------------------------------
------------V(J,K)[1:NUM_EW1 := ADJUSTV(J,K)[1:NUM_EW1];
```

```
                W(J,K)[1:NUM_EW] := ADJUSTW(J,K)[1:NUM_EW];
          ENDF;
        ENDF;
        CALL DIVRGNCE;
    /*LEE BBB:      ENDP:LEE*/
    /*LEE*/
  BBB:WRITE UNIT(6)#'LEAVE PROCEDURE-SOR'#;
        ENDP:/*SOR LEE*/
  /*LEE*/
        ENDM:
 EOT-ENCOUNTERED.
 /
```

FILE WINDP

ave.windp
awind.windp
EWIND.WINDP
oeyshf.windp

```
MODULE WIND(UNIT1=GLBTROP,UNIT4=COEFMTRX,UNIT7=SOLMTRX
            ,UNIT8=ADJMTRX,UNIT6=RESULT,UNIT9=SOLMTR)
BEGIN MAIN;
/*CC---PROGRAMME WIND LEADS TO AN ADJUSTED MASS CONSERVATIVE ----CCCC-- 0002
C------WIND FIELD FROM REAL-TIME OBSERVED WIND FIELD DATA. THIS ------C-- 0003
C------PROGRAMME INVOLVES NUMERICAL SOLUTIONS OF THE PARTIAL ---------C-- 0004
C------DIFFERENTIAL EQUATIONS RESULTING FROM A CONSTRAINED ------------C-- 0005
C------VARIATIONAL FORMULATION (EULER-LAGRANGE TYPE). THIS PROGRAMME ---C-- 0006
CCCC---IS WRITTEN IN STAR LANGUAGE-1(SL1) OF CYBER COMPUTER. ----------CCCC-- 0007
CCCC---DEFINITION OF TERMS ---------------------------------------------CCCC-- 0008
C------X:=EASTWARD DIRECTION ----------------------------------------------C-- 0009
C------Y:=NORTHWARD DIRECTION ---------------------------------------------C-- 0010
C------Z:=VERTICAL DIRECTION ----------------------------------------------C-- 0011
C------XX:=GRID SIZE IN X-DIRECTION,DEGREES & RADIANS RESPECTIVELY -------C-- 0012
C------YY:=GRID SIZE IN Y-DIRECTION,DEGREES & RADIANS RESPECTIVELY -------C-- 0013
C------ZZ:=GRID SIZE IN Z-DIRECTION ---------------------------------------C-- 0014
C------IL:=NUMBERS OF GRIDS IN X- DIRECTION -------------------------------C-- 0015
C------JL:=NUMRERS OF GRIDS IN Y--DIRECTION -------------------------------C-- 0016
C------KL:=NUMBERS OF GRIDS IN Z--DIRECTION -------------------------------C-- 0017
C------THETA-&-TH :=ANGLE IN DEGREES & RADIANS RESPECTIVELY FROM SOUTH ----C-- 0018
C----------------POLE(-90.0) TO NORTH POLE(+90.0) ------------------------C-- 0019
C------NLAMDA:=DUMMY LAGRANGE MULTIPLIER FORCOUNTING PURPOSE ---------------C-- 0020
C------TALPHA:=THREE DIMENSIONAL COEFFICIENT MATRIX ------------------------
CCCC---SS:=ONE DIMENSIONAL SOLUTION MATRIX ----------------------------CC*/ -- 0023
/*---THE FOLLOWING SECTION NEEDS TO BE CHANGED FOR ------------------------
-----DIFFERENT GRID SYSTEM. XTRA1- TO ISM3 ARE PAGING ---------------------
-----PARAMETRES. THIER VALUES SHOULD BE ESTIMATED -------------------------
-----ACCORDINGLY. AL1,AL3 AND AL2 ARE WEIGHTING FACTORS -------------------
----FOR U,V AND W VELOCITY COMPONENETS RESPECTIVELY.--*/ -------------------
/*---REGINING OF THE SECTION ----------------------------------*/ ----------
LITERALLY "5.0" DXX; ------------------------------------------------------
LITERALLY "5.0" DYY; ------------------------------------------------------
LITERALLY "72" IL; --------------------------------------------------------
LITERALLY "37" JL; --------------------------------------------------------
LITERALLY "9" KL; ---------------------------------------------------------
LITERALLY "10"KKLL; /*KKLL=KL+1*/ ----------------------------------------
LITERALLY "23976" LIK; ----------------------------------------------------
LITERALLY "0.01" AL1; -----------------------------------------------------
LITERALLY "0.01" AL3; -----------------------------------------------------
LITERALLY "1.0" AL2; ------------------------------------------------------
LITERALLY "33576" XTRA1; --------------------------------------------------
LITERALLY "15992" XTRA2; --------------------------------------------------
LITERALLY "11192" XTRA3; --------------------------------------------------
LITERALLY "768" ISM1; -----------------------------------------------------
LITERALLY "640" ISM2; -----------------------------------------------------
LITERALLY "256" ISM3; -----------------------------------------------------
/*--END OF THE SECTION.----------*/ ----------------------------------------
REAL VECTOR (KKLL) Z; -----------------------------------------------------
---INITIAL(Z=\0.0333334,0.0333334,0.0333334,0.0333334 ---------------------
----------,0.1,0.2,0.2,0.2,0.2,0.2,0.2\); ------------------------------------
LITERALLY "ZVECTOR(K)" ZZ;
```

```
.ITERALLY "ZZ1[K]" Z1;
-ITERALLY-"ZZ2[K]"-Z2;
-ITERALLY-"ZZ3[K]"-Z3;
REAL-VECTOR-[KL]-Z1,Z2,Z3,ZVECTOR;
-ITERALLY-3.14159265358979793"-PI;
-ITERALLY-"WRITE"-PRINT;
-ITERALLY-"RTRS(1)[I,JL]"-TS1;
-ITERALLY-"RTRS(2)[I,JL]"-TS2;
-ITERALLY-"RTRS(3)[I,JL]"-TS3;                                        00
-ITERALLY-"RTRS(4)[I,JL]"-TS4;                                        00
-ITERALLY-"RTRS(5)[I,JL]"-TS5;                                        39
-ITERALLY-"R(J,K)[I,JL]"RO;                                          0040
/*-DEFINITIONS-----------
--T1=LAMDA(J,K)[I],T2=LAMDA(J,K)[I+1],T3=LAMDA(J,K)[I-1]
--T4=LAMDA(J+1,K)[I],T5=LAMDA(J-1,K)[I],T6=LAMDA(J,K+1)[I]
--T7=LAMDA(J,K-1)[I],T8=LAMDA(J,K+1)[I+1],T9=LAMDA(J,K-1)[I+1]
--T10=LAMDA(J,K+1)[I-1],T11=LAMDA(J,K-1)[I-1]
--T12=LAMDA(J+1,K+1)[I],T13=LAMDA(J+1,K-1)[I],
--T14=LAMDA(J-1,K+1)[I],T15=LAMDA(J-1,K-1)[I]
--RTRS=DISTANCE-BETWEEN-SURFACE-AND-TROPOPAUSE-AT-ANY-GRID.
--GAMAU,-GAMAV-AND-GAMAW-ARE-COEFFICIENTS-FOR-THE
--AD.LISTING EQUATIONS.-------------------------------------*/
----------LITERALLY-"TALPHA[1]"-T1;
----------LITERALLY-"TALPHA[2]"-T2;
----------LITERALLY-"TALPHA[3]"-T3;
----------LITERALLY-"TALPHA[4]"-T4;
----------LITERALLY-"TALPHA[5]"-T5;
----------LITERALLY-"TALPHA[6]"-T6;
----------LITERALLY-"TALPHA[7]"-T7;
----------LITERALLY-"TALPHA[8]"-T8;
----------LITERALLY-"TALPHA[9]"-T9;
----------LITERALLY-"TALPHA[10]"-T10;
----------LITERALLY-"TALPHA[11]"-T11;
----------LITERALLY-"TALPHA[12]"-T12;
----------LITERALLY-"TALPHA[13]"-T13;
----------LITERALLY-"TALPHA[14]"-T14;
----------LITERALLY-"TALPHA[15]"-T15;
----------LITERALLY-"GAMAU(1)[I,JL]"-GU1;
----------LITERALLY-"GAMAU(2)[I,JL]"-GU2;
----------LITERALLY-"GAMAV(1)[I,JL]"-GV1;
----------LITERALLY-"GAMAV(2)[I,JL]"-GV2;
----------LITERALLY-"GAMAW(1)[I,JL]"-GW1;
----------LITERALLY-"GAMAU(1)[I]"GGU1;
----------LITERALLY-"GAMAU(2)[I]"GGU2;
----------LITERALLY-"GAMAV(1)[I]"GGV1;
----------LITERALLY-"GAMAV(2)[I]"GGV2;
----------LITERALLY-"GAMAW(1)[I]"GGW1;
----------EXTERNAL-PROCEDURE(INTEGER,BIT_VECTOR[64],REAL--,
----------------------INTEGER,INTEGER)-Q3OPNMAP;
--------EXTERNAL-PROCEDURE(INTEGER,BIT_VECTOR[64])-Q3CLOSE;
    REAL VECTOR [15] ARRAY(IJK) TX;
```

```
        COMMON CT(TX,TDUMP1);
        REAL VECTOR [XTRA1] TDUMP1;
        REAL VECTOR [13] ARRAY(IJK) SX;
        COMMON CS(SX,TDUMP2);
        REAL VECTOR [XTRA2] TDUMP2;
        REAL VECTOR [5] ARRAY(IJK) ADJX;
        COMMON CA(ADJX,TDUMP3);
        REAL VECTOR [XTRA3] TDUMP3;
        REAL VECTOR [IJK] ARRAY(13) SSX;
        COMMON CSS(SSX,TDUMP4);
    REAL VECTOR [XTRA2] TDUMP4;
        REAL VECTOR [IL] ARRAY(2) DAMAU,DAMAV,DAMAW;
        REAL VECTOR [5] ADJ;
        INTEGER IER1,IER2,IER3,IER4;
INTEGER I,J,K,NL,I1,I2,I3,I4,J1,J2,K3,KK,JJ,K2,ITL,COUNT;
REAL VECTOR[6] RRO;
REAL AL4,CON,ZZZ,XX, THETA,TH,YY;
REAL VECTOR [IL] ARRAY(8) ALPHA,BETA;                                   0043
REAL VECTOR [IL] ARRAY(5)RTRS;
REAL VECTOR [15] TALPHA;
REAL VECTOR [13] SS;
REAL VECTOR [IL] A,B,C,E;
REAL VECTOR [IL] ARRAY(JL,KL)R;
REAL VECTOR [JL] TROP;
INITIAL (TROP=\9.2,9.3,9.4,9.5,9.7,9.9,10.1,10.3,10.5,10.7,
        10.9,11.1,11.8,16.8,17.8,18.0,18.0,18.0,18.0,
        18.0,18.0,18.0,17.8,16.8,11.8,11.1,10.9,10.7,10.5,
        10.3,10.1,9.9,9.7,9.5,9.4,9.3,9.2\);
    REAL VECTOR[IL]RR1,RR2,RR3,RR4,RR5,RR6;
PROCEDURE MAIN;                                                         0051
    WRITE UNIT(6)('DXX,DYY,KL,IJK,AL1,AL2,AL3)#'DXX=',F5.2,
        'DYY=',F5.2,'KL=',I2,'IJK=',I7,'AL1=',F7.3,
        'AL2=',F7.3,'AL3=',F7.3#;
    ZVECTOR[1:KL]:=(Z[2:KKL1]+Z[1:KL])/2.0;
    ZZ1[1:KL]:=1.0/Z[2:KKL];
    ZZ3[1:KL]:=1.0/Z[1:KL];
    ZZ2[1:KL]:=ZZ1[1:KL]+ZZ3[1:KL];
    CALL OBOPNMAP(IER1,X\434F464D54525820\,TX(1)[1],ISM1,1);
    CALL OBOPNMAP(IER2,X\534F4C4D54522020\,SX(1)[1],ISM2,1);
    CALL OBOPNMAP(IER3,X\41444A4D54525820\,ADJX(1)[1],ISM3,1);
    WRITE UNIT(6),(IER1,IER2,IER3)#'IER1=',Z16,4X,
        'IER2=',Z16,4X,'IER3=',Z16#;
    COUNT:=0;
    CON:=PI/180.0;
        /* INITIALIZING TALPHA,ADJ,SS ARRAYS */
        TALPHA[1:15]:=0.0;
        ADJ[1:5]:=0.0;
        SS[1:13]:=0.0;
    XX:=DXX*CON;
    YY:=DYY*CON;
        K2:=KL+1;
```

```
      AL4:=(AL1/AL3)**2;
      IL:=II;
/*CC--INITIALIZING ELEMENTS OF R(IL,KL)[IL]                    --CC*/
   FOR JJ:=1 TO JL DO
   READ UNIT(1)(R(JJ,1)[1:IL])#6E13.6#;
   R(JJ,1)[1:IL]:=6371000.0+R(JJ,1)[1:IL];
   R(JJ,KL)[1:IL]:=6371000.0 ++TROP[JJ]*1000.0;
   ZZZ:=0.0;
   FOR KK:=2 TO KL-1 DO
   ZZZ:=ZZZ+Z[KK];
   TS1:=R(JJ,KL)[1:IL]-R(JJ,1)[1:IL];
   RO:=R(JJ,1)[1:IL]+ZZZ*TS1;
   ENDF;
   ENDF;
/*CC---THE FOLLOWING DO-LOOP DEFINES THE ELEMENTS OF COEFFICIENT----CCCC--  0069
C      MATRIX "TALPHA" AND SOLUTION MATRIX "SS". IT HAS FIVE          C    0070
C      PROCEDURES WHICH DEAL WITH EQUATIONS SUITABLE FOR PARTICULAR   C    0071
C      CONDITIONS. PROCEDURE "SNPOLS" STANDS FOR SOUTH NORTH POLAR    C    0072
C      SURFACE. SIMILIARLY "SNPOLI" STANDS FOR SOUTH NORTH POLAR      C    0073
C      INTERIOR; SNPOLT FOR SOUTH NORTH POLAR TROPOPAUSE;             C    0074
C      "INTERS" FOR INTERIOR POINTS AND SURFACE AND "TROPOI" FOR      C    0075
CCCC---TROPOPAUSE- INTERIOR.                                  ---CC*/    0076
   THETA:=-90.0-DYY;
   FOR JJ:=1 TO JL DO                                                    0078
   THETA:=THETA+DYY;                                                     0079
   TH:=THETA-*CON;                                                       0080
   FOR KK:=1 TO KL DO                                                    0081
   TS1:=R(JJ,KL)[1:IL]-R(JJ,1)[1:IL];
   IF ((JJ=1).OR.(JJ=JL)).AND.(KK=1) THEN CALL SNPOLS; ENDI;
   IF ((JJ=1).OR.(JJ=JL)).AND.((KK>1).AND.(KK<KL)) THEN CALL SNPOLI;
   ENDI;
   IF ((JJ=1).OR.(JJ=JL)).AND.(KK=KL) THEN CALL SNPOLT; ENDI;
   IF ((JJ>1).AND.(JJ<JL)).AND.(KK<KL) THEN CALL INTERS; ENDI;
   IF ((JJ>1).AND.(JJ<JL)).AND.(KK=KL) THEN CALL TROPOI; ENDI;
   ENDF;                                                                 0092
   ENDF;                                                                 0093
   CALL D30CLOSE(IER1,X\434F464D54525820\);
   CALL D30CLOSE(IER3,X\41444A4D54525820\);
   CALL D30PNMAP(IER4,X\534F4C4D54525820\,SSX(1)[1],ISM2+1);
   WRITE UNIT(6)(IER4)#'IER4=',Z16A#;
   FOR I:=1 TO 13 DO
   JJ:=1;
LS: SSX(I)[JJ]:=SX(JJ)[I];
   JJ:=JJ+1;
   IF (JJ<=IJK) THEN GO TO LS; ENDI;
   ENDF;
   CALL D30CLOSE(IER2,X\534F4C4D54522020\);
   CALL D30CLOSE(IER4,X\534F4C4D54525820\);
   WRITE UNIT(6)#'CLOSE'#;
   WRITE UNIT(6)(IER1,IER2,IER3,IER4)#'IER1=',Z16.4X,
                'IER2=',Z16.4X,'IER3=',Z16.4X,'IER4=',Z16A#;
```

```
      ENDP;                                                              0143
      PROCEDURE SNPOLS;                                                  0144
/*CC   PROCEDURE - "SNPOLS" DEALS WITH THOSE EQUATIONS WHICH ARE  CCCC   0145
C     SUITABLE FOR SOUTH AND NORTH POLES SURFACES. IT ESTIMATES     C      0146
C     COEFFICIENTS OF EACH CORRESPONDING "LAMDA" TERMS AND TRANSFERS C     0147
CCCC  THEM TO THE WORKING MATRICES-"TALPHA","SS".              CC*/       0148
      IF J=1 THEN JJ:=1; ENDI;                                           0150
      IF J=JL THEN JJ:=JL-2; ENDI;                                       0151
      RR1[1:IL]:=R(JJ+1,1)[1:IL]-R(J,1)[1:IL];                           0152
      ALPHA(2)[1:IL]:=((AL1*AL1/(1.0*AL2*AL2*RTRS(1)[1:IL]))+(((RR1
      [1:IL])/(R(J,1)[1:IL]*YY))**2)*AL4/(1.0*RTRS(1)[1:IL]))*71;
      ALPHA(3)[1:IL]:=(AL4*RR1[1:IL])/(R(J,1)[1:IL]*R(J,1)[1:IL]*
                      YY*YY);
      BETA(2)[1:IL]:=2.0*AL1*AL1*(RR1[1:IL])/(R(J,1)[1:IL]*YY);          0159
      BETA(3)[1:IL]:=2.0*AL1*AL1;                                        0160
      GV1:=1.0/(2.0*AL3*AL3*R0*YY);
      GV2:=RR1[1:IL]/(2.0*AL3*AL3*TS1*R0*YY*(1.0/71));
      GW1:=1.0/(2.0*AL2*AL2*TS1*(1.0/71));
  FOR I:=1 TO IL DO                                                      0163
      T1:=ALPHA(2)[I]+ABS(ALPHA(3)[I]);
      IF(J=1) THEN T4:=-ABS(ALPHA(3)[I]);ENDI;
      IF(J=JL) THEN T5:=-ABS(ALPHA(3)[I]);ENDI;
      T6:=-ALPHA(2)[I];
      SS[2]:=-BETA(2)[I];
      SS[3]:=BETA(3)[I];
      AD.JL3]:=GAMAV(1)[I];
      AD.JL4]:=GAMAV(2)[I];
      AD.J[5]:=GAMAW(1)[I];
      CALL TRANSFER;
  ENDF;                                                                  0170
      ENDP;                                                              0171
      PROCEDURE SNPOLI;                                                  0172
/*CC   PROCEDURE - "SNPOLI" DEALS WITH THOSE EQUATIONS WHICH ARE  CCCC   0173
C     SUITABLE FOR SOUTH AND NORTH POLES INTERIOR POINTS. IT        C      0174
C     ESTIMATES COEFFICIENTS OF EACH CORRESPONDING "LAMDA" TERMS AND C     0175
CCCC  TRANSFERS THEM TO THE WORKING MATRICES-"TALPHA","SS".    CC*/       0176
      IF JJ=1 THEN JJ:=1 ENDI;                                           0178
      IF JJ=JL THEN JJ:=JL-2 ENDI;                                       0179
      RR2[1:IL]:=R(JJ+1,K)[1:IL]-R(J,K)[1:IL];                           0180
      ALPHA(1)[1:IL]:=(AL4*RR2[1:IL]*71)/(1.0*RTRS(1)[1:IL]*R0*R0*YY);
      ALPHA(2)[1:IL]:=AL4/(R0*R0*YY);
      ALPHA(3)[1:IL]:=ABS(ALPHA(1)[1:IL])+ALPHA(2)[1:IL];
      BETA(1)[1:IL]:=-2.0*AL1*AL1/R0;
      GV1:=1.0/(2.0*AL3*AL3*R0*YY);
      GV2:=RR2[1:IL]/(4.0*AL3*AL3*R0*TS1*YY*ZZ);
      GW1:=1.0/(4.0*AL2*AL2*TS1*ZZ);
  FOR I:=1 TO IL DO                                                      0182
      T1:=+ALPHA(3)[I];
      T6:=-ABS(ALPHA(1)[I]);
      IF(J=1) THEN T4:=-ALPHA(2)[I];ENDI;
```

```
      IF(J=JL) THEN T5:=-ALPHA(2)[I]:ENDI;
    IF(R(JJ+1,K)[I]<R(JJ,K)[I]) THEN
    SS[12]:=-BETA(1)[I];
  ELSE
    SS[12]:=BETA(1)[I];
  ENDI;
      ADJ[3]:=GAMAV(1)[I];
      ADJ[4]:=GAMAV(2)[I];
      ADJ[5]:=GAMAW(1)[I];
      CALL TRANSFER;
ENDF;                                                              0192
    ENDP;                                                          0193
    PROCEDURE SNPOLT;                                              0194
/*CC---PROCEDURE- "SNPOLT"-DEALS WITH THOSE EQUATIONS WHICH ARE -----CCCC---0195
C-------SUITABLE FOR SOUTH AND NORTH POLES- TROPOPAUSES, IT ESTIMATES------C---0196
C-------COEFFICIENTS OF EACH CORRESPONDING "LAMDA"TERMS AND TRANSFERS-----C---0197
CCCC---THEM TO THE WORKING MATRICES-"TALPHA","SS"-------------------CC*/---0198
      IF J=1 THEN JJ:=1 ENDI;                                      0199
      IF J=JL THEN JJ:=JL-2 ENDI;                                  0200
    ALPHA(1)[I:IL]:=AL4/(R0*R0*YY);
    BETA(1)[I:IL]:=2.0*AL1*AL1/R0;
      GV1:=1.0/(2.0*AL3*AL3*R0*YY);
FOR I:=1 TO IL DO                                                  0201
    T1:=+ALPHA(1)[I];
    IF(J=1) THEN T4:=-ALPHA(4)[I];ENDI;
    IF(J=JL) THEN T5:=-ALPHA(1)[I];ENDI;
    SS[12]:=+BETA(1)[I];
      ADJ[3]:=GAMAV(1)[I];
      CALL TRANSFER;
ENDF;                                                              0207
    ENDP;                                                          0208
    PROCEDURE INTEPS;                                              0209
/*CC---PROCEDURE-"INTEPS"-DEALS WITH THOSE EQUATIONS WHICH ARE -----CCCC---0210
C-------SUITABLE FOR ALL INTERIOR POINTS AND SURFACE POINTS EXCEPT-------C---0211
C-------SOUTH AND NORTH POLES SURFACES, IT ESTIMATES COEFFICIENTS OF------C---0212
C-------EACH CORRESPONDING "LAMDA"-TERMS AND TRANSFERS-THEM TO THE------C---0213
C-------WORKING MATRICES="TALPHA","SS", IN CASE OF SURFACE POINTS, THE---C---0214
CCCC---FIRST "IF"-STATEMENT ACCESS TO ANOTHER PROCEDURE-"INSUR".---CC*/---0215
    IS2:=R(J+1,KL)[1:IL]-R(J+1,1)[1:IL];                           0217
    RTRS(3)[1:IL]:=R(J-1,KL)[1:IL]-R(J-1,1)[1:IL];                 0218
    RTRS(4)[1:IL-1]:=RTRS(1)[2:IL];                                0219
    RTRS(4)[IL]:=RTRS(1)[1];                                       0220
    RTRS(5)[2:IL]:=RTRS(1)[1:IL-1];                                0221
    RTRS(5)[1]:=RTRS(1)[IL];                                       0222
    RR3[1]:=R(J,K)[2]-R(J,K)[IL];                                  0223
    RR3[IL]:=R(J,K)[1]-R(J,K)[IL-1];                               0224
    RR3[2:IL-1]:=R(J,K)[3:IL]-R(J,K)[1:IL-2];                      0225
    RR4[1:IL]:=R(J+1,K)[1:IL]-R(J-1,K)[1:IL];                      0226
    RR5[1]:=R(J,K)[2]-2.0*R(J,K)[1]+R(J,K)[IL];                    0227
    RR5[IL]:=R(J,K)[1]-2.0*R(J,K)[IL]+R(J,K)[IL-1];                0228
    RR5[2:IL-1]:=R(J,K)[3:IL]-2.0*R(J,K)[2:IL-1]+R(J,K)[1:IL-2];   0229
```

```
      RR6[1:IL]:=R(J+1,K)[1:IL]-2.0*R(J,K)[1:IL]+R(J-1,K)[1:IL];
A[1:IL-1]:=((AL1/(AL2*TS1))**2)+(((RR3[1:IL])/(2.0*XX*                                  0231
R(J,K)[1:IL-1]*TS1*COS(TH)))**2)+(((RR4[1:IL])/(2.0*YY*
R(J,K)[1:IL]*TS1))**2)*AL4);
      IF (JCJL).AND.(K=-1) THEN CALL INSUR; END;
      IF (JCJL).AND.(K=1) THEN GO TO LA; END;
R[1:IL]:=((AL1/AL2)**2)*2.0/(R(J,K)[1:IL-1]*TS1)+((RR3[1:IL]                            0235
*(TS4-TS5))/(2.0*((R(J,K)[1:IL]*TS1                                                     0236
*COS(TH)*XX)**2)))-(RR5[1:IL]/(TS1*((R0*XX*COS(TH))**2                                  0
)))+(((AL4*RR4[1:IL])*(TS2-TS3))/(2.0*((R0*TS1                                          0
*YY)**2)))-((AL4*RR6[1:IL])/(TS1*((                                                     0
R0*YY)**2)))+((TAN(TH)*(AL4*RR4[1:IL]))/(2.0*YY*TS1*R0*R0));                           0
C[1:IL]:=-(RR3[1:IL])/(XX*TS1*((R0*COS(TH))**2));                                       0240
E[1:IL]:=-(AL4*RR4[1:IL])/(YY*TS1*R0*R0);                                               0
ALPHA(1)[1:IL]:=2.0*((1.0/((R0*COS(TH)*XX)**2))+(AL4/((R0
*YY)**2))+(Z2*A[1:IL]/(2.0*ZZ)));
ALPHA(2)[1:IL]:=1.0/((R0*COS(TH)*XX)**2);                                               0244
ALPHA(3)[1:IL]:=((1.0/(R0*R0*YY*YY))-(TAN(TH)/(2.0*R0*R0*YY))
)*AL4;
ALPHA(4)[1:IL]:=((1.0/(R0*R0*YY*YY))+(TAN(TH)/(2.0*R0*R0*YY))
)*AL4;
ALPHA(5)[1:IL]:=(Z1*A[1:IL]/(ZZ))+(R[1:IL]/(2.0*ZZ));
ALPHA(6)[1:IL]:=(Z3*A[1:IL]/(ZZ))-(B[1:IL]/(2.0*ZZ));
ALPHA(7)[1:IL]:=C[1:IL]/(4.0*ZZ*XX);                                                    0249
ALPHA(8)[1:IL]:=E[1:IL]/(4.0*ZZ*YY);                                                    0250
BETA(1)[1:IL]:=AL1*AL1/(R0*COS(TH)*XX);                                                 0251
BETA(2)[1:IL]:=AL1*AL1/(R0*YY);                                                         0252
BETA(3)[1:IL]:=AL1*AL1/(TS1*ZZ);                                                        0253
BETA(4)[1:IL]:=AL1*AL1*(RR3[1:IL])/(2.0*XX*ZZ*TS1*R0*COS(TH));                          0254
BETA(5)[1:IL]:=AL1*AL1*(RR4[1:IL])/(2.0*YY*ZZ*R0*TS1);                                  0255
BETA(6)[1:IL]:=2.0*AL1*AL1*TAN(TH)/R0;                                                  0256
BETA(7)[1:IL]:=4.0*AL1*AL1/R0;                                                          0257
GU1:=1.0/(4.0*AL1*AL1*R0*COS(TH)*XX);
GU2:=RR3[1:IL]/(8.0*AL1*AL1*R0*TS1*COS(TH)*XX*ZZ);
GV1:=1.0/(4.0*AL2*AL2*R0*YY);
GV2:=RR4[1:IL]/(8.0*AL2*AL2*R0*TS1*YY*ZZ);
GW1:=1.0/(4.0*AL2*AL2*TS1*ZZ);
FOR I:=1 TO IL DO;                                                                      0260
I1:=I+1;                                                                                0261
I2:=I-1;                                                                                0262
IF I1=IL+1 THEN I1:=1;END;                                                              026
IF I2=0 THEN I2:=IL;END;                                                                0264
T1:=+ALPHA(1)[I];                                                                       0266
T2:=-ALPHA(2)[I];
T3:=-ALPHA(2)[I];                                                                       0270
T4:=-ALPHA(3)[I];                                                                       0272
T5:=-ALPHA(4)[I];                                                                       0274
T6:=-ALPHA(5)[I];                                                                       0276
T7:=-ALPHA(6)[I];                                                                       0278
T8:=-ALPHA(7)[I];                                                                       0280
T9:=+ALPHA(7)[I];                                                                       0282
```

```
      T10:=+ALPHA(7)[I];                                                    0284
------T11:=-ALPHA(7)[I];-------------------------------------------------------02
------T12:=-ALPHA(8)[I];-------------------------------------------------------02
------T13:=+ALPHA(8)[I];-------------------------------------------------------0290
------T14:=+ALPHA(8)[I];-------------------------------------------------------0292
------T15:=-ALPHA(8)[I];-------------------------------------------------------02
------SS[1]:=+BETA(1)[I];
------SS[2]:=-BETA(1)[I];
------SS[5]:=+BETA(2)[I];
------SS[6]:=-BETA(2)[I];
------SS[9]:=+BETA(3)[I];
------SS[10]:=-BETA(3)[I];
------SS[3]:=-BETA(4)[I];
------SS[4]:=+BETA(4)[I];
------SS[7]:=-BETA(5)[I];
------SS[8]:=+BETA(5)[I];
------SS[12]:=-BETA(6)[I];
------SS[13]:=+BETA(7)[I];
----------ADJ[1]:=GAMAU(1)[I];
----------ADJ[2]:=GAMAU(2)[I];
----------ADJ[3]:=GAMAV(1)[I];
----------ADJ[4]:=GAMAV(2)[I];
----------ADJ[5]:=GAMAW(1)[I];
----------CALL TRANSFER;
----END;-------------------------------------------------------------------0307
---LA:ENDP;-----------------------------------------------------------------0308
-------PROCEDURE INSUR;-----------------------------------------------------0309
/*CC---PROCEDURE "INSUR" DEALS WITH THOSE EQUATIONS WHICH ARE ----CCCC----0310
C       SUITABLE FOR ALL SURFACE POINTS EXCLUDING SOUTH AND NORTH ------C---0311
C       POLES ONES. IT ESTIMATES COEFFICIENTS OF EACH CORRESPONDING ----C---0312
C       "LAMDA" TERMS AND TRANSFERS THEM TO THE WORKING MATRICES -------C---0313
C       ="ALPHA","SS".                                                  */
----------ALPHA(1)[1:IL]:=ABS(RR3[1:IL]/(2.0*XX*XX*R0*R0*-------------------
-------------------------COS(TH)*COS(TH)));-------------------------------------
----------ALPHA(2)[1:IL]:=ABS(RR4[1:IL]*AL4/(2.0*R0*R0*YY*YY));--------------
----------ALPHA(3)[1:IL]:=A[1:IL]*TS1*Z1;-------------------------------------
----------BETA(1)[1:IL]:=AL1*AL1*RR3[1:IL]/(R0*COS(TH)*XX);-------------------
----------BETA(2)[1:IL]:=AL1*AL1*RR4[1:IL]/(R0*YY);---------------------------
----------BETA(3)[1:IL]:=2.0*AL1*AL1;-----------------------------------------
----------GU1:=1.0/(4.0*AL1*AL1*R0*TS1*COS(TH)*XX);----------------------------
----------GU2:=RR3[1:IL]*Z1/(4.0*AL1*AL1*R0*TS1*COS(TH)*XX);-------------------
----------GV1:=1.0/(4.0*AL3*AL3*R0*YY);----------------------------------------
----------GV2:=RR4[1:IL]*Z1/(4.0*AL3*AL3*R0*TS1*YY);----------------------------
----------GW1:=Z1/(2.0*AL2*AL2*TS1);--------------------------------------------
----FOR I:=1 TO IL DO ------------------------------------------------------0319
----------I1:=I+1;-------------------------------------------------------------0320
----------I2:=I-1;-------------------------------------------------------------0321
--------------IF I1=IL+1 THEN I1:=1;END;-------------------------------------0324
--------------IF I2=0 THEN I2:=IL;END;---------------------------------------0325
----------T1:=ALPHA(1)[I]+ALPHA(2)[I]+ALPHA(3)[I];
        TA:=-ALPHA(3)[I];
```

```
        SS[11]:=-BETA(1)[I];
        SS[12]:=-BETA(2)[I];
        SS[13]:=BETA(3)[I];
        IF((R(J-1)[I1]>=R(J,1)[I2]).AND.(R(J+1,1)[I]
                                        >R(J-1,1)[I])) THEN
        T3:=-ALPHA(1)[I];
        T5:=-ALPHA(2)[I];
        ENDI;
        IF((R(J,1)[I1]<=R(J-1)[I2]).AND.(R(J+1,1)[I]
                                        <R(J-1,1)[I])) THEN
        T2:=-ALPHA(1)[I];
        T4:=-ALPHA(2)[I];
        ENDI;
        IF((R(J,1)[I1]<R(J,1)[I2]).AND.(R(J+1,1)[I]
                                        >=R(J-1,1)[I])) THEN
        T2:=-ALPHA(1)[I];
        T5:=-ALPHA(2)[I];
        ENDI;
        IF((R(J,1)[I1]>R(J,1)[I2]).AND.(R(J+1,1)[I]
                                        <=R(J-1,1)[I])) THEN
        T3:=-ALPHA(1)[I];
        T4:=-ALPHA(2)[I];
        ENDI;
        ADJ[1]:=GAMAU(1)[I];
        ADJ[2]:=GAMAU(2)[I];
        ADJ[3]:=GAMAV(1)[I];
        ADJ[4]:=GAMAV(2)[I];
        ADJ[5]:=GAMAW(1)[I];
        CALL TRANSFER;
        ENDF;
        ENDP;
        PROCEDURE TROPOI;                                              0512
/*CC   PROCEDURE "TROPOI" DEALS WITH THOSE EQUATIONS WHICH ARE    CCCC  0513
C       SUITABLE FOR TROPOPAUSE INTERIOR POINTS. IT ESTIMATES         C  0514
C       COEFFICIENTS OF EACH CORRESPONDING "LAMDA" TERMS AND          C  0515
CCCC   TRANSFERS THEM TO THE WORKING MATRICES "TALPHA","SS".    CC*/  0516
AUTOMATIC REAL VECTOR(IL)RR10,RR11;                                    0517
        RR10[1]:=R(J,K)[2]-R(J,K)[IL];                                 0518
        RR10[IL]:=R(J,K)[1]-R(J,K)[IL-1];                              0519
        RR10[2:IL-1]:=R(J,K)[3:IL]-R(J,K)[1:IL-2];                     0520
        RR11[1:IL]:=R(J+1,K)[1:IL]-R(J-1,K)[1:IL];                     0521
        ALPHA(1)[1:IL]:=1.0/((R0*COS(TH)*XX)**2);                      0522
        ALPHA(2)[1:IL]:=((1.0-(TAN(TH)*YY/2.0))/((R0*YY)**2))*AL4;
        ALPHA(3)[1:IL]:=((1.0+(TAN(TH)*YY/2.0))/((R0*YY)**2))*AL4;
        A[1:IL]:=((AL1/(AL2*TS1))**2)+(((RR10[1:IL])/(2.0*XX*R0        0525
        *TS1*COS(TH)))**2)+((((RR11[1:IL])/(2.0*YY*R0*TS1))**2)*AL4);
        ALPHA(4)[1:IL]:=+(2.0/((R0*COS(TH)*XX)**2))+(2.0*AL4/((R0*YY)  0
        **2))+(Z2*A[1:IL]/(ZZ));
        ALPHA(5)[1:IL]:=Z2*A[1:IL]/(ZZ);
        BETA(1)[1:IL]:=AL1*AL1/(R0*COS(TH)*XX);                        0530
        BETA(2)[1:IL]:=AL1*AL1/(R0*YY);                                0531
```

```
      BETA(3)[1:IL]:=AL1*AL1/(TS1*ZZ);                                  0532
    BETA(4)[1:IL]:=AL1*AL1*(RR1O[1:IL])/(2.0*XX*ZZ*TS1*RO*COS(TH));     0533
    BETA(5)[1:IL]:=AL1*AL1*(RR1[1:IL])/(2.0*YY*ZZ*TS1*RO);              0534
    BETA(6)[1:IL]:=2.0*AL1*AL1*TAN(TH)/RO;                              0535
    BETA(7)[1:IL]:=4.0*AL1*AL1/RO;
    GU1:=1.0/(4.0*AL1*AL1*RO*COS(TH)*XX);
    GV1:=1.0/(4.0*AL3*AL3*RO*YY);
  FOR I:=1 TO IL DO                                                     0536
    I1:=I+1;                                                            0537
    I2:=I-1;                                                            0538
      IF I1=IL+1 THEN I1:=1;END;                                        0539
      IF I2=0 THEN I2:=IL;END;                                          0540
    T1:=ALPHA(4)[I];                                                    0542
    T2:=-ALPHA(1)[I];                                                   0544
    T3:=-ALPHA(1)[I];                                                   0546
    T4:=-ALPHA(2)[I];                                                   0548
    T5:=-ALPHA(3)[I];                                                   0550
    T7:=-ALPHA(5)[I];                                                   0552
    SS[1]:=+BETA(1)[I];
    SS[2]:=-BETA(1)[I];
    SS[5]:=+BETA(2)[I];
    SS[6]:=-BETA(2)[I];
    SS[9]:=+BETA(3)[I];
    SS[10]:=-BETA(3)[I];
    SS[3]:=-BETA(4)[I];
    SS[7]:=-BETA(5)[I];
    SS[4]:=+BETA(4)[I];
    SS[8]:=+BETA(5)[I];
    SS[12]:=-BETA(6)[I];
    SS[13]:=+BETA(7)[I];
    ADJ[1]:=GAMAU(1)[I];
    ADJ[3]:=GAMAV(1)[I];
    CALL TRANSFER;
  ENDF;                                                                 0554
  ENDP;                                                                 0555
    PROCEDURE TRANSFER;
    /* THIS PROCEDURE TRANSFER CALCULATED VALUES OF
    ELEMENTS OF COEFFICIENT MATRIX (COEMTRX), SOLUTION
    MATRIX (SOLMTRX), ADJUSTING MATRIX (ADJMTRX) TO
    THE CORRESPONDING FILES NAMED AS COEMTRX,SOLMTRX
    ADJMTRX. THESE FILES WILL BE ACCESSED IN WINDADJUST
    MODULE. */
    COUNT:=COUNT+1;
    TX(COUNT)[1:15]:=TALPHA;
    SX(COUNT)[1:13]:=SS;
    ADJX(COUNT)[1:5]:=ADJ;
    TALPHA[1:15]:=0.0;
    ADJ[1:5]:=0.0;
    SS[1:13]:=0.0;
  ENDP;
  ENDM;
```

FILE LAMDA

```
rewind,lamda
$REWIND,LAMDA.
/copysbf,lamda
1       MODULE WINDADJS(UNIT1=COEFMTRX,UNIT2=SOLMTRX,UNIT3=
                        ADJMTRX,UNIT6=RESULT)$ BEGIN MAIN;
        /* THIS PROGRAM ADJUSTS OBSERVED WIND FIELD TO MAKE
        IT MASS-CONSERVATIVE WIND FIELD.ADJUSTING TECHNIQUE
        IS BASED ON VARITIONAL FORMULATION. SIMULTANEOUS
        EQUATIONS FOR LAMDA TERMS ARE SOLVED USING SUCCESSIVE
        OVER RELAXATION (S-O-R) ITERATION METHOD. FILES NAMED AS
        COEFMTRX,SOLMTRX,AND ADJMTRX HAVE NESSASARY
        ESTIMATED COEFFICIENTS. THESE FILES WILL BE READ IN THIS
        PROGRAM. */
    /*  THE FOLLOWING SECTION NEEDS TO BE CHANGED FOR DIFFERENT
        GRID SYSTEM. XTRA1 TO XTRA7 AND ISM1 TO ISM7 ARE
        PAGING PARAMETERS. AL1 IS WEIGHTING FACTOR FOR U VELOCITY
        COMPONENT.                                          */
    /*  BEGINING OF THE SECTION ------*/
        LITERALLY "5.0" DYY;
        LITERALLY "72" JL;
        LITERALLY "37" IL;
        LITERALLY "9" KL;
        LITERALLY "10" KLLL; /* KLLL= KL+1 */
        LITERALLY "23976" IJK;
        LITERALLY "33576" XTRA1;
        LITERALLY "15992" XTRA2;
        LITERALLY "17584" XTRA3;
```

```
        LITERALLY "24512" XTRA4;
    LITERALLY "51152" XTRA5;
LITERALLY "H1192" XTRA7;
    LITERALLY "768" ISM1;
    LITERALLY "640" ISM2;
    LITERALLY "128" ISM3;
    LITERALLY "256" ISM4;
    LITERALLY "256" ISM5;
LITERALLY "256" ISM7;
        LITERALLY "0.01"AL1;
        LITERALLY "0.001" TOLRNCE;
        LITERALLY "1.75"RLXFAC;
        LITERALLY "3.14159265358979"PI;
    /* IL=NUMBER OF GRIDS IN EAST-WEST DIRECTION.
        JL=NUMBER OF GRIDS IN NORTH-SOUTH DIRECTION.
        KL=NUMBER OF GRIDS IN ALTITUDE DIRECTION.
        IJK=TOTAL NUMBER OF GRIDS.
        TOLRNCE=TOLRANCE IN ITERATIONS.
        RLXFAC=RELAXATION FACTOR IN SUCCESIVE OVER RELAXTION
        METHOD. */
/* END OF THE SECTION */
    REAL VECTOR [15] ARRAY(IJK) TX;
    COMMON CT(TX,TDUMP1);
    REAL VECTOR [XTRA1] TDUMP1;
    REAL VECTOR [5] ARRAY(IJK) ADJX;
    COMMON CA(ADJX,TDUMP3);
    REAL VECTOR [XTRA7] TDUMP3;
    REAL VECTOR [IJK] ARRAY(13) SSX;
    COMMON CSS(SSX,TDUMP4);
REAL VECTOR [XTRA2] TDUMP4;
    REAL VECTOR [15] ARRAY(IJK) TLAMDA;
    COMMON B1(TLAMDA,TDUMP11);
    REAL VECTOR [XTRA1] TDUMP11;
    REAL VECTOR [IJK] S.DIV;
COMMON B2(S.DIV,TDUMP5);
REAL VECTOR [XTRA3] TDUMP5;
    REAL VECTOR [5] ADJ;
    REAL VECTOR [IL] ARRAY(JL,KL) ALAMDA,BLAMDA;
    COMMON B3(BLAMDA,ALAMDA,TDUMP6);
    REAL VECTOR [XTRA3] TDUMP6;
    REAL VECTOR [IL] ARRAY(JL,KL) U,V,W,P,ADJUSTU,ADJUSTV,
                                    ADJUSTW;
    COMMON B4(U,V,W,P,TDUMP7);
    REAL VECTOR [XTRA4] TDUMP7;
    COMMON B5(ADJUSTU,ADJUSTV,ADJUSTW,TDUMP8);
    REAL VECTOR [XTRA5] TDUMP8;
    REAL VECTOR [IJK] ARRAY(13) VEL;
    COMMON B6(VEL,TDUMP9);
    REAL VECTOR [XTRA2] TDUMP9;
    REAL VECTOR [KL,L] Z;
    INITIAL (Z=\0.0,0.0333334,0.0666668,0.0666668
```

```
                    ,0.1,0.2,0.2,0.2,0.2,0.2\);
        REAL AL3,T,ERROR,ZZZ,THETA,TH,CON,MAXDIV,MINDIV;
        INTEGER CHECK,I,J,K,N1,I1,I2,I3,I4,J1,J2,K2,
                K3,ITERATION,K1,IW,E1,E2,E3,E4,E5,COUNTING,
                E6,E7,E8,E9,E10,MMMM,JJ;
        REAL VECTOR [333] WLAMDA,WLAMDA1;
      REAL VECTOR [9] KLAMDA;
        REAL MAXLAMDA,MINLAMDA;
              PROCEDURE MAIN;
        OPENMAP('F1',TLAMDA(1)[1],ISM1,E1,'LARGE');
        OPENMAP('F2',S[1],ISM3,E2,'LARGE');
        OPENMAP('F3',BLAMDA(1,1)[1],ISM3,E3,'LARGE');
        OPENMAP('F4',U(1,1)[1],ISM4,E4,'LARGE');
      OPENMAP('F5',ADJUSTU(1,1)[1],ISM5,E5,'LARGE');
        OPENMAP('F6',VEL(1)[1],ISM2,E6,'LARGE');
        OPENMAP('COEMTRX',TX(1)[1],ISM1,E8,'LARGE');
        OPENMAP('SOLMTRX',SSX(1)[1],ISM2,E9,'LARGE');
        OPENMAP('ADJMTRX',ADJX(1)[1],ISM7,E10,'LARGE');
        WRITE UNIT(6)(E1,E2,E3,E4,E5,E6,E8,E9
                ,E10)#'OPEN',9(3XI2)#;
            COUNTING:=0;
        WRITE UNIT(6)(DYY,IL,JL,KL,AL1,TOLRNCE,RLXFAC)#'DYY=',F5.2,
            'IL=',I3,'JL=',I3,'KL=',I3,'AL1=',F7.3,'TOLRNCE=',F8.4,
            'RLXFACTOR=',F6.3#;
        CON:=PI/180.0;
            K2:=KL+1;
        /* INITIALIZING ALAMDA AND BLAMDA ARRAYS-GIVING
            FIRST GUESS VALUES-*/
  /*        THETA:=+90.0+DYY;
        FOR J:=1 TO JL DO
          THETA:=THETA-DYY;
        FOR K:=1 TO KL DO
          BLAMDA(J,K)[1:IL]:=2.0;
          ALAMDA(J,K)[1:IL]:=2.0;
        ENDF;
        ENDF;
      FOR J:=1 TO JL DO
      FOR K:=1 TO KL DO
      FOR I:=1 BY 2 TO IL DO
        BLAMDA(J,K)[I]:=1.0;
        ALAMDA(J,K)[I]:=1.0;
      ENDF;ENDF;ENDF; */
        BLAMDA(5,5)[36]:=1.5;
        ALAMDA(5,5)[36]:=1.5;
      FOR J:=1 TO JL DO
        WLAMDA[J]:=BLAMDA(J,5)[36];
      ENDF;
      WRITE UNIT(6)(WLAMDA[1:JL])#5E14.7#;
      FOR K:=1 TO KL DO
        KLAMDA[K]:=BLAMDA(5,K)[36];
      ENDF;
```

```
      WRITE UNIT(6)(KLAMDA[1:KL])#'KLAMDA=',5E14.7#;
    WRITE UNIT(6)(RLAMDA(5,5)[30:40])#'LAMDA=',5E14.7#;
    WRITE UNIT(6)(RLAMDA(5,5)[1],RLAMDA(5,5)[IL-])
                    -#'LAMDA=',2E14.7#;
    FOR I:=1 TO 13 DO
    VEL(I)[1:IJK]:=0.0;
    ENDF;
    /* INITIALIZING WIND-FIELD U,V,W AND DENSITY-D ARRAYS-*/
    THETA:=-90.0-DYY;
    FOR JL:=1 TO JL DO
    THETA:=THETA+DYY;
    TH:=THETA*CON;
    FOR K:=1 TO K2 DO
    U(J,K)[1:IL]:=10.0;
    V(J,K)[1:IL]:=3.0;
    W(J,K)[1:IL]:=0.0;
    D(J,K)[1:IL]:=1.0;
    ENDF;
    ENDF;
    CALL DIVRGNCE;
    CALL SOR;
    CLOSE('F1',F1);
    CLOSE('F2',F2);
    CLOSE('F3',F3);
    CLOSE('F4',F4);
    CLOSE('F5',F5);
    CLOSE('F6',F6);
    CLOSE('COEFMTRX',F8);
    CLOSE('SOLMTRX',F9);
    CLOSE('ADJMTRX',F10);
    WRITE UNIT(6)(F1,F2,F3,F4,F5,F6,F8
                  ,F9,F10)#'CLOSE', 9(3XI2)#;
    ENDP;

    PROCEDURE DIVRGNCE;
    /*-THIS PROCEDURE ESTIMATES THE VALUE OF RIGHT HAND SIDE
    OF EQUATIONS. AT THE END IT CALCULATES THE DIVERGENCE
    OF OBSERVED OR ADJUSTED WIND FIELD. */
    NL:=0;
    FOR J:=1 TO JL DO
    FOR K:=1 TO KL DO
    FOR I:=1 TO IL DO
    NL:=NL+1;
    I1:=I+1;
    I2:=I-1;
    IF I1=IL+1 THEN I1:=1;ENDI;
    IF I2=0 THEN I2:=IL;ENDI;
    VEL(1)[NL]:=U(J,K)[I1]*D(J,K)[I1];
    VEL(2)[NL]:=U(J,K)[I2]*D(J,K)[I2];
    VEL(3)[NL]:=U(J,K+1)[I]*D(J,K+1)[I];
      IF K=1 THEN
```

```
                VEL(4)[NL]:=(2*U(J,1)[I]-U(J,2)[I])*(2*D(J,1)[I]
                                        -D(J,2)[I]);
        ELSE
            VEL(4)[NL]:=U(J,K-1)[I]*D(J,K-1)[I];
    ENDI;ENDF;ENDF;ENDF;
    NL:=0;
    FOR J:=1 TO JL DO
    FOR K:=1 TO KL DO
    FOR I:=1 TO IL DO
        NL:=NL+1;
        J1:=J+1;
        J2:=J-1;
        IF J1=JL+1 THEN J1:=JL-1;ENDI;
        IF J2=0 THEN J2:=2;ENDI;
        VEL(5)[NL]:=V(J1,K)[I]*D(J1,K)[I];
        VEL(6)[NL]:=V(J2,K)[I]*D(J2,K)[I];
        VEL(7)[NL]:=V(J,K+1)[I]*D(J,K+1)[I];
    IF K=1 THEN
        VEL(8)[NL]:=(2*V(J,1)[I]-V(J,2)[I])*(2*D(J,1)[I]
                                        -D(J,2)[I]);
    ELSE
        VEL(8)[NL]:=V(J,K-1)[I]*D(J,K-1)[I];
    ENDI;ENDF;ENDF;ENDF;
    NL:=0;
    FOR J:=1 TO JL DO
    FOR K:=1 TO KL DO
    FOR I:=1 TO IL DO
        NL:=NL+1;
        VEL(9)[NL]:=W(J,K+1)[I]*D(J,K+1)[I];
    IF K=1 THEN
        VEL(10)[NL]:=(2*W(J,1)[I]-W(J,2)[I])*(2*D(J,1)[I]
                                        -D(J,2)[I]);
    ELSE
        VEL(10)[NL]:=W(J,K-1)[I]*D(J,K-1)[I];
    ENDI;
        VEL(11)[NL]:=U(J,K)[I]*D(J,K)[I];
        VEL(12)[NL]:=V(J,K)[I]*D(J,K)[I];
        VEL(13)[NL]:=W(J,K)[I]*D(J,K)[I];
    ENDF;
        ENDF;
        ENDF;
/*CALCULATING RIGHT HAND SIDE OF EQUATIONS */
    SC1:IJK]:=0.0;
/*  FOR I:=1 TO 13 DO
        SC1:IJK]:=SC1:IJK]+SSX(I)[1:IJK]*VEL(I)[1:IJK];
    ENDF;  */
/*CALCULATING DIVERGENCE OF WIND FIELD*/
    AL3:=2.0*AL1*AL1;
    DIV[1:IJK]:=SC1:IJK]/AL3;
    MAXDIV:=MAX(DIV);
```

```
        MINDIV:=MIN(DIV);
    WRITE(UNIT(6)(MAXDIV,MINDIV)#'MAXDIV=',E14.7,
                                'MINDIV=',E14.7#);
ENDP;
    PROCEDURE SOR;
    /*-THIS-PROCEDURE-IS-A-PROGRAM-FOR-A-SUCCESSIVE-OVER-
      RELAXATION (S-O-R) ITERATION METHOD TO SOLVE SIMULTANEOUS
      EQUATIONS.FIRST-VELOCITIES-AND LAMDA-TERMS-ARE ARRANGED
      IN VECTOR FORMS,-AND-THEN-S-O-R-EQUATION-IN-GENERAL
      FORM-IS USED-FOR-ITERATIONS.-*/
      ITERATION:=0;
LC: NL:=0;
    /*-INITIALIZING TLAMDA ARRAYS-*/
    FOR-I:=1-TO I.JK-DO
      TLAMDA(I)[1:15]:=0.0;
    ENDF;
      CHECK:=1;
    FOR-J:=1-TO-JL-DO
    FOR-K:=1-TO-KL-DO
    FOR-I:=1-TO-IL-DO
      NL:=NL+1;
      I1:=I+1;
      I2:=I-1;
      I3:=I+2;
      I4:=I-2;
      J1:=J+1;
      J2:=J-1;
      K1:=K+1;
      K3:=K-1;
    IF-I1=IL+1-THEN-I1:=1;ENDI;
    IF-I2=0---THEN-I2:=IL;ENDI;
    IF-J1=JL+1-THEN-J1:=JL-1;ENDI;
    IF-J2=0---THEN-J2:=2;ENDI;
    IF-K1=KL+1-THEN-K1:=KL;ENDI;
    IF-K3=0-THEN-K3:=1;ENDI;
    IF-I3=IL+2-THEN-I3:=2;ENDI;
    IF-I3=IL+1-THEN-I3:=1;ENDI;
    IF-I4=0---THEN-I4:=IL;ENDI;
    IF-I4=-1---THEN-I4:=IL-1;ENDI;
      TLAMDA(NL)[1]:=BLAMDA(J,K)[I];
      TLAMDA(NL)[2]:=BLAMDA(J,K)[I1];
      TLAMDA(NL)[3]:=BLAMDA(J,K)[I2];
      TLAMDA(NL)[4]:=BLAMDA(J1,K)[I];
      TLAMDA(NL)[5]:=BLAMDA(J2,K)[I];
      TLAMDA(NL)[6]:=BLAMDA(J,K1)[I];
      TLAMDA(NL)[7]:=BLAMDA(J,K3)[I];
      TLAMDA(NL)[8]:=BLAMDA(J,K1)[I1];
      TLAMDA(NL)[9]:=BLAMDA(J,K3)[I1];
      TLAMDA(NL)[10]:=BLAMDA(J,K1)[I2];
      TLAMDA(NL)[11]:=BLAMDA(J,K3)[I2];
      TLAMDA(NL)[12]:=BLAMDA(J1,K1)[I];
```

```
                TLAMDA(NL)[13]:=BLAMDA(J1,K3)[I];
            TLAMDA(NL)[14]:=BLAMDA(J2,K1)[I];
            TLAMDA(NL)[15]:=BLAMDA(J2,K3)[I];
        /*-ENDING OF VECTORIZATION-*/
        /*-BEGINING OF-S-O-R METHOD-*/
            T:=TLAMDA(NL).DOT.TX(NL);
        BLAMDA(J,K)[I]:=ALAMDA(J,K)[I]+(RLXFAC*(S[NL]-T)/
                                TX(NL)[I]);
    IF((J=1).OR.(J=JL))-THEN
    IF(J=1)-THEN JJ:=2;ENDI;
    IF(J=JL)-THEN JJ:=JL-1;ENDI;
    BLAMDA(J,K)[I]:=SUM(BLAMDA(JJ,K)[1:IL])/IL;
    ENDI;
    BLAMDA(5,5)[36]:=1.5;
    IF(ALAMDA(J,K)[I]=0.0)-THEN-ALAMDA(J,K)[I]:=0.0001;ENDI;
        ERROR:=1.0-(BLAMDA(J,K)[I]/ALAMDA(J,K)[I]);
        IF-ABS(ERROR)>TOLRNCE-THEN CHECK:=CHECK+1;ENDI;
    ENDF;
    ENDF;
    ENDF;
    FOR-J:=1-TO-JL-DO
    WLAMDA[J]:=BLAMDA(J,5)[36];
    ENDF;
    WRITE-UNIT(6)(WLAMDA[1:JL])#5E14.7#;
    FOR-K:=1-TO KL-DO-
    KLAMDA[K]:=BLAMDA(5,K)[36];
    ENDF;
    WRITE-UNIT(6)(KLAMDA[1:KL])#'KLAMDA=',5E14.7#;
    WRITE-UNIT(6)(BLAMDA(5,5)[30:40])#'LAMDA=',5E14.7#;
    WRITE-UNIT(6)(BLAMDA(5,5)[1],BLAMDA(5,5)[IL])
                    #'LAMDA=',2E14.7#;
        MMMM:=0;
        NL:=0;
        FOR-J:=1-TO-JL-DO
        FOR-K:=1-TO-KL-DO
        NL:=NL+1;
    WLAMDA[NL]:=MAX(BLAMDA(J,K));
    WLAMDA1[NL]:=MIN(BLAMDA(J,K));
    IF-(MMMM<3)-THEN
    IF-(WLAMDA1[NL]<0.0)-THEN
    WRITE-UNIT(6)(J,K,WLAMDA1[NL])#'J=',I3,'K=',I3,
                    'WLAMDA1=',E14.7#;
        MMMM:=MMMM+1;
    ENDI;ENDI;
    IF((J=37).AND.(K=1))-THEN
    WRITE-UNIT(6)(J,K,WLAMDA1[NL])#'J=',I3,'K=',I3,
                    'WLAMDA1=',F14.7#;
    ENDI;
        ALAMDA(J,K)[1:IL]:=BLAMDA(J,K)[1:IL];
    ENDF;
    ENDF;
```

```
      MAXLAMDA:=MAX(WLAMDA);
      MINLAMDA:=MIN(WLAMDA(*));
      WRITE(UNIT(6),(MAXLAMDA,MINLAMDA)#'LAMDA=',' MAX=',E14.7,
                                        'MIN=',E14.7#);
      ITERATION:=ITERATION+1;
      WRITE(UNIT(6),(CHECK)#'NUMBER OF ELEMENTS NOT CHECKED=',
                                        I8#);
      COUNTING:=COUNTING+1;
      WRITE UNIT(6),(COUNTING)#'COUNTING OF ITERATIONS=',I8#);
      IF(COUNTING>10)THEN GO TO BBB;ENDI;
      IF CHECK>1 THEN GO TO LC;ENDI;
      WRITE UNIT(6),(ITERATION)#1HO,'NUMBER OF ITERATIONS=',
                                        I3#);
      /* ENDING OF S-O-R METHOD */
      /* PRINTING OF LAMDA TERMS */
/*    FOR J:=1 TO JL DO    */
      FOR J:=1 BY 6 TO JL DO
      FOR K:=1 TO KL DO
      WRITE(UNIT(6),(J,K)#1HO,'J=',I3,'K=',I3#);
      WRITE(UNIT(6),(BLAMDA(J,K)[1:10])#5E14.7#);
      ENDF;
      ENDF;
      /* ADJUSTMENT OF OBSERVED WIND FIELD TO MAKE IT MASS
      CONSERVATIVE */
      NL:=0;
      FOR J:=1 TO JL DO
      FOR K:=1 TO KL DO
      FOR I:=1 TO IL DO
      NL:=NL+1;
      ADJ[1:5]:=ADJX(NL)[1:5];
      I1:=I+1;
      I2:=I-1;
      J1:=J+1;
      J2:=J-1;
      K1:=K+1;
      K3:=K-1;
      IF (I1=IL+1) THEN I1:=1;ENDI;
      IF (I2=0   ) THEN I2:=IL;ENDI;
      IF (J1=JL+1) THEN J1:=JL-1;J2:=JL;ENDI;
      IF (J2=0   ) THEN J2:=1;ENDI;
      IF (K1=KL+1) THEN K1:=KL;ENDI;
      IF (K3=0   ) THEN K3:=1;ENDI;

      ADJUSTU(J,K)[I]:=U(J,K)[I]+ADJ[1]*(BLAMDA(J,K)[I]
                      -BLAMDA(J,K)[I2])-ADJ[2]*(BLAMDA(J,K1)[I]
                      -BLAMDA(J,K3)[I]);
      ADJUSTV(J,K)[I]:=V(J,K)[I]+ADJ[3]*(BLAMDA(J1,K)[I]
                      -BLAMDA(J2,K)[I])-ADJ[4]*(BLAMDA(J,K1)[I]
                      -BLAMDA(J,K3)[I]);
      ADJUSTW(J,K)[I]:=W(J,K)[I]+ADJ[5]*(BLAMDA(J,K1)[I]
                      -BLAMDA(J,K3)[I]);
```

```
            ENDF;
          ENDF;
          ENDF;

          /*--PRINTING OF ADJUSTED WIND FIELD */
     /*   FOR J:=1 TO JL DO   */
          FOR J:=1 BY 6 TO JL DO
          FOR K:=1 TO KL DO
          WRITE UNIT(6)(J,K)#1H0,'J=',I3,'K=',I3#;
            WRITE UNIT(6)(U(J,K)[1:10],ADJUSTU(J,K)[1:10],
                          V(J,K)[1:10],ADJUSTV(J,K)[1:10],
                          W(J,K)[1:10],ADJUSTW(J,K)[1:10])#5E14.7#;
          ENDF;
          ENDF;
          /*--CALCULATING DIVERGENCE OF ADJUSTED WIND FIELD */
            FOR J:=1 TO JL DO
            FOR K:=1 TO KL DO
            U(J,K)[1:IL]:=ADJUSTU(J,K)[1:IL];
            V(J,K)[1:IL]:=ADJUSTV(J,K)[1:IL];
            W(J,K)[1:IL]:=ADJUSTW(J,K)[1:IL];
            ENDF;
            ENDF;
          CALL DIVRGNCE;

BBR:      ENDP;
          ENDM;
EOI ENCOUNTERED
/IDLE
```

FILE MASSC

```
save.massc,windc.mass
/rewind.massc
$REWIND.MASSC.
/copysbf.massc
JSTR.ER.T100.CM120000.
ACCOUNT.689650E
CHARGE.102258.LRC.
DELIVER.R2114VVWASEEM.AKHTAR
GET(MASS)
GET(GLBTROP)
ATTACH.SL.LIB=LIBRARY
RENUE.-
```

```
   RFL,120000.
SI-1,MASS,DUMMY,SI.1BIN.
REPLACE,DUMMY.
REWIND,SI.1BIN.
TOSTAR(I,SI.1BIN=RI//U,GLATROP=//U)
DAYFILE,DAY2.
REPLACE(DAY2)
EXIT.
REPLACE(DUMMY)
DAYFILE,DAY2.
REPLACE(DAY2)
)STORE-689650-102258 MASS          R          U
MASS,T70.
DELIVER(R21-14)
LOAD(SI.1BIN,I-I=SI.ILIB,CN=GO,OU=LODFILE,#200,
  GROL=*C1,GROL=*C2,GROL=*C3)
ATTACH,COFF,VELOCITY.
REQUEST,F1-128.
GO.
DAYFILE,DAY1.
TOAS(Z=689650F,RESULT)
EXIT.
DAYFILE,DAY1.
TOAS(Z=689650F,RESULT)
COMPOMP.
EOI ENCOUNTERED.
```

FILE WINDC

```
/rewind,windc.
$REWIND,WINDC.
/copysbf,windc.
1STR,JOB,T100,CM120000.
ACCOUNT,689650F.
CHARGE,102258,LRC.
DELIVER,R2114VVWASEEM-AKHTAR
GET(WINDP)
GET(GLRTROP)
ATTACH,SL1/UN=LIBRARY.
REDUCE,-.
REL,120000.
SL1,WINDP,DUMMY,SL1BIN.
REPLACE,DUMMY.
REWIND,SL1BIN.
TOSTAR(I,SL1BIN=R1//U,GLRTROP=//(U)
DAYFILE,DAY2.
REPLACE(DAY2)
EXIT.
REPLACE(DUMMY)
DAYFILE,DAY2.
REPLACE(DAY2)
1STORE 689650 102258 WINDP R U 2
WIND,T70.
DELIVER(R2114)
```

```
LOAD(SOLBIN,LIB=SOLLIB,CN=80,OU=LOGFILE,#200,
--GROUP=*CT,GROUP=*CS,GROUP=*CA,GROUP=*CSS)
---ATTACH,COEFMTRX,
---ATTACH,SOLMTRX,
---ATTACH,ADJMTRX,
--- PURGE(COEFMTRX,SOLMTRX,ADJMTRX,SOLMTR)
---REQUEST,COEFMTRX/768,
--- REQUEST,SOLMTRX/640,
---REQUEST,ADJMTRX/256,
---REQUEST,SOLMTR/640,
---DEFINE,COEFMTRX,
---DEFINE,SOLMTRX,
--- DEFINE,ADJMTRX,
---GO,
---DAYFILE,DAY1,
---TRAS(7=689650E,RESULT)
---EXIT,
---DAYFILE,DAY1,
---TRAS(7=689650E,RESULT)
---DUMPDMP,
---EOI--ENCOUNTERED,
--/
```

FILE WIND

```
-rewind,wind-----
 $REWIND,WIND.-----------------------------------------------------------
-/rewind-mass-----------------------------------------------------------
 $REWIND,MASS.-----------------------------------------------------------
 /copysbf,mass-----------------------------------------------------------
1-------MODULE CONTINUITY(UNIT1=GLBTROP,UNIT2=COFF,UNIT6=RESULT----------
                          ,UNIT3=VELOCITY) BEGIN MAIN;--------------------
 /*CC--PROGRAM CONTINUITY LEADS TO AN ADJUSTED MASS --,-------->C---------
      CONSERVATIVE WIND FIELD FROM REAL-TIME OBSERVED DATA.---------------
      THE OBSERVED WIND COMPONENTS U AND V ARE SUBSITUTED ----------------
      IN THE CONTINUITY EQUATION TO ESTIMATE W COMPONENT.-----------------
      THE CONTINUITY EQUATION IS WRITTEN IN SPHERICAL---------------------
      COORDINATES WITH A VERTICAL COORDINATE SYSTEM WHICH TAKES -----------
      INTO ACCOUNT THE VARIABLE TROPOGRAPHY OF THE EARTH'SURFACE----------
      AND VARIABLE TROPOPAUSE HEIGHT.-THIS PROGRAMME IS ------------------
      WRITTEN IN STAR LANGUAGE 1 (SL1) FOR CYBER MACHINE. -----CC*/-------
 /*CC--DEFINITIONS-----------------------------------------------------
      X=EASTWAARD DIRECTION ----------------------------------------------
      Y=NORTHWARD DIRECTION ----------------------------------------------
      Z=VERTICAL DIRECTION- ----------------------------------------------
      XX=GRID SIZE IN X-DIRECTION ----------------------------------------
      YY=GRID SIZE IN Y-DIRECTION ----------------------------------------
      ZZ=GRID SIZE IN Z-DIRECTION ----------------------------------------
      IL=NUMBER OF GRID POINTS IN X-DIRECTION ----------------------------
      JL=NUMBER OF GRID POINTS IN Y-DIRECTION ----------------------------
      KL=NUMBER OF GRID POINTS IN Z-DIRECTION ----------------------------
      THETA &-TH=ANGLE IN DEGREES & RADIANS RESPECTIVELY FROM-------------
              SOUTH POLE(-90) TO NORTH POLE(+90)----------------------------
      R= DISTANCE FROM EARTH CENTER TO THE POINT OF INTEREST--------------
        IN TROPOSPHERE ----------------------------------------------------
      TROP= TROPOPAUSE HEIGHT --------------------------------------------
      U,V & W=WIND COMPONENTS IN X,Y & Z DIRECTIONS RESPECTIVELY----------
       -----END OF DEFINITIONS----CC*/
```

```
   /* THE FOLLOWING SECTION NEEDS TO BE CHANGED FOR A GIVEN
- ---GRID-SYSTEM,--FOR-EXAMPLE:-FOR-A-GRID-SYSTEM-5-DEGREE-LONGI-----------
-------TUDE-X--5-DEGREE-LATITUDE-X-9-VERTICAL-LEVELS-WE-HAVE-IL=72,-------
 -------JL=37,--KL=9-AND-TOTAL-GRID-POINTS-IJK=23976.--XTRA1-TO-XTRA3-AND-----
-------ISM1-TO--ISM3-ARE-PAGING-PARAMETERS-WHOSE-VALUES-SHOULD-BE-ESTIMATED-----
-------FOR-A-GIVEN-SYSTEM.--IN-THIS-PROGRAMME-VARIABLES-R-,U,V,W-AND-----------
-------AND-ALPHA-ARE-PAGED-SEPARETELY-ON-DIFFERENT-LARGE-PAGES.----------------
-------EXAMPLE:-ALPHA-VARIABLE-HAS-TOTAL--136080-BIT-WORDS--(6X315X72).---------
-------THIS-IS-EQUIVALENT-TO-6X(JL-2)X(KL+1)XIL.-----------------------------
-------A-SINGLE-LARGE-PAGE-CAN-HAVE-65536-BIT-WORDS-OR-128-SMALL-PAGES.-------
-------THEREFORE-WE-NEED-THREE-LARGE-PAGES-WHICH-WILL-BE-FILLED-PARTIALLY-----
-------BY-ALPHA-VARIABLE.-EXTRA-PLACE-IN-THE-SECOND-PAGE-WILL-BE-------------
-------FILLED-BY-A-DUMMY-PARAMETER-DUMP3.-THE-DIMENSION-OF-THIS-PARAMETER-----
-------CAN-BE-ESTIMATED-AS-XTRA3=3X65536-136080=60528-AND-PARAMETER-ISM3-------
-------(IN-OPEN-MAP-STATEMENT)=3X128=384.-----------------------------CC*/-----
----------LITERALLY-"5.0"-DXX;-----------------------------------------------
----------LITERALLY-"5.0"-DYY;-----------------------------------------------
----------LITERALLY-"72"-IL;-------------------------------------------------
----------LITERALLY-"37"-JL;--------------------------------------------------
----------LITERALLY-"9"-KL;---------------------------------------------------
----------LITERALLY-"10"KKLL;/*KKLL=KL+1--*/----------------------------------
----------LITERALLY-"23976"-IJK;----------------------------------------------
----------LITERALLY-"315"-IJK1;/*-IJK1=(JL-2)X-KL--*/-------------------------
----------LITERALLY-"41560"XTRA1;---------------------------------------------
----------LITERALLY-"51152"XTRA2;---------------------------------------------
----------LITERALLY-"60528"XTRA3;---------------------------------------------
----------LITERALLY-"128"ISM1;------------------------------------------------
----------LITERALLY-"256"ISM2;------------------------------------------------
----------LITERALLY-"384"ISM3;------------------------------------------------
---/*--END-OF-THE-SECTION-TO-BE-CHANGED-----------------------------------*/---
----------LITERALLY-"ZVECTOR[K]"-ZZ;------------------------------------------
----------LITERALLY-"R(J,K)[1:IIL]"-RO;---------------------------------------
----------LITERALLY-"RTRS[1:IIL]"-TS1;----------------------------------------
----------LITERALLY-"ALPHA(1,NL)[1:IIL]"-A1;----------------------------------
----------LITERALLY-"ALPHA(2,NL)[1:IIL]"-A2;----------------------------------
----------LITERALLY-"ALPHA(3,NL)[1:IIL]"-A3;----------------------------------
----------LITERALLY-"ALPHA(4,NL)[1:IIL]"-A4;----------------------------------
----------LITERALLY-"ALPHA(5,NL)[1:IIL]"-A5;----------------------------------
----------LITERALLY-"ALPHA(6,NL)[1:IIL]"-A6;----------------------------------
----------LITERALLY-"3.14159265389793"-PI;------------------------------------
----------REAL-VECTOR-[KKLL]-Z;-----------------------------------------------
----------REAL-VECTOR-[KL]-ZVECTOR;-------------------------------------------
----------REAL-VECTOR-[JL]-TROP;----------------------------------------------
----------REAL-VECTOR-[IL]-RTRS,RR1,RR2,SS;-----------------------------------
----------REAL-VECTOR-[IL]-ARRAY(6)-VEL;--------------------------------------
----------REAL-VECTOR-[IJK]-WVEL;---------------------------------------------
---/*-PAGING-OF-VARIABLES-R,U,V,W-AND-ALPHA--*/--------------------------------
----------REAL-VECTOR-[IL]-ARRAY(JL,KL)-R;------------------------------------
----------COMMON-C1(R,DUMP1);-------------------------------------------------
----------REAL-VECTOR-[XTRA1]-DUMP1;------------------------------------------
          REAL VECTOR [IL] ARRAY(JL,KKLL) U,V,W;
```

```
          COMMON C2(U,V,W,DUMP2);
          REAL VECTOR [XTRA2] DUMP2;
          REAL VECTOR [IL] ARRAY(6,IJK1) ALPHA;
          COMMON C3(ALPHA,DUMP3);
          REAL VECTOR [XTRA3] DUMP3;
     /*-END OF PAGING */
          INTEGER I,J,K,NL,E1,E2,E3,K1,KK,III;
          REAL MAXW,MINW,ZZZ,THETA,TH,XX,YY,CON;
     INITIAL (Z=\0.0333334,0.0333334,0.0333334,0.0333334
                    ,0.1,0.2,0.2,0.2,0.2,0.2\);
     INITIAL (TROP=\9.2,9.3,9.4,9.5,9.7,9.9,10.1,10.3,10.5,10.7,
                    10.9,11.1,11.8,16.8,17.8,18.0,18.0,18.0,18.0,
                    18.0,18.0,18.0,17.8,16.8,11.8,11.1,10.9,10.7,10.5,
                    10.3,10.1,9.9,9.7,9.5,9.4,9.3,9.2\);
          PROCEDURE MAIN;
          OPENMAP('F1',R(1,1)[1],ISM1,E1,'LARGE');
          OPENMAP('VELOCITY',U(1,1)[1],ISM2,E2,'LARGE');
          OPENMAP('COEFF',ALPHA(1,1)[1],ISM3,E3,'LARGE');
          WRITE UNIT(6)(E1,E2,E3)#'OPEN',3(3X,I2)#;
          III:=II;
          CON:=PI/180.0;
     /*   INITIALIZING U, V AND W VELOCITY VARIABLES    */
          THETA:=-90.0-DYY;
          TH:=THETA*CON;
          FOR J:=1 TO JL DO
          FOR K:=1 TO KKL DO
          U(J,K)[1:IL]:=-10.0;
          V(J,K)[1:IL]:=3.0*COS(TH);
          W(J,K)[1:IL]:=0.0;
          ENDF;ENDF;
     /*CC    THE FOLLOWING STATEMENT "CALL COEFFICIENT" LEADS THE
               EXECUTION TO PROCEDURE COEFFICIENT. THIS PROCEDURE
               ESTIMATES THE NECESSARY COEFFICIENTS FOR THE CONTINUITY
               EQUATION FOR A GIVEN CONDITIONS SUCH
               AS GLOBAL TOPOGRAPHY OF EARTH'S SURFACE, TROPOPAUSE HEIGHT
               AND GRID SIZES. THEREFORE FOR A GIVEN CONDITIONS THIS
               PROCEDURE SHOULD BE ACCESSED ONLY ONCE. THE COEFFICIENTS
               THUS ESTIMATED WILL BE STORED PERMANENTLY IN A FILE
               CALLED "COEFF". IN ORDER TO AVOID EXECUTION OF THIS
               STATEMENT TYPE /* AND */ ON LEFT AND RIGHT SIDE OF
               "CALL COEFFICIENT" STATEMENT.          CC*/

          CALL COEFFICIENT;

     /*   THE FOLLOWING "DO LOOP" FIRST CONVERTS VELOCITY
          DATA IN THE VECTOR FORM AND THEN USING THE CORRESPONDING
          COEFFICIENTS FROM FILE "COEFF" ESTIMATES VERTICAL VELOCITY
          COMPONENT "W" WHICH SATISFIES CONTINUITY EQUATION.   */
          NL:=0;
          FOR J:=2 TO IL-1 DO
               FOR K:=1 TO KL DO
```

```
                NL:=NI+1;
                K1:=K-1;
              IF-(K1=0)-THEN-K1:=1;ENDI;
              VEL(1)[1:IL]:=W(J,K)[1:IL];
              VEL(2)[1]:=U(J,K)[2]-U(J,K)[IL];
              VEL(2)[IL]:=U(J,K)[1]-U(J,K)[IL-1];
              VEL(2)[2:IL-1]:=U(J,K)[3:IL]-U(J,K)[1:IL-2];
              VEL(3)[1:IL]:=V(J+1,K)[1:IL]-V(J-1,K)[1:IL];
              VEL(4)[1:IL]:=V(J,K)[1:IL];
              VEL(5)[1:IL]:=U(J,K+1)[1:IL]-U(J,K1)[1:IL];
              VEL(6)[1:IL]:=V(J,K+1)[1:IL]-V(J,K1)[1:IL];
                SS[1:IL]:=0.0;
              FOR KK:=1-TO-6-DO
                SS[1:IL]:=SS[1:IL]+(VEL(KK)[1:IL]*ALPHA(KK,NL)[1:IL]);
            ENDF;
            IF(K=1)-THEN W(J,K+1)[1:IL]:=SS[1:IL];ENDI;
            IF(K>1) THEN W(J,K+1)[1:IL]:=W(J,K1)[1:IL]+SS[1:IL];ENDI;
          ENDF;
          ENDF;

    /*-TO-ESTIMATE-MAXIMUM-AND-MINIMUM-VALUE-OF-"W"-VELOCITY---*/
          NL:=0;
        FOR-J:=1-TO-JL-DO
        FOR-K:=1-TO-KL-DO
        FOR-I:=1-TO-IL-DO
          NL:=NL+1;
        WVEL[NL]:=W(J,K)[I];
        ENDF;ENDF;ENDF;
        MAXW:=MAX(WVEL);
        MINW:=MIN(WVEL);
        WRITE-UNIT(6)(MAXW,MINW)#'MAXW=',E14.7,'MINW=',E14.7#;
          CLOSE-('E14',E1);
          CLOSE-('VELOCITY',E2);
          CLOSE-('COEFF',E3);
          WRITE-UNIT(6)(E1,E2,E3)#'CLOSE=',3(3XI2)#;
      ENDP;
          PROCEDURE-COEFFICIENT;
    /*---SEE-DETAILED-COMMENT-ABOUT-THIS-PROCEDURE-IN-THE-
        "PROCEDURE-MAIN".                                  */
          ZVECTOR[1:KL]:=(Z[2:KK+1]+Z[1:KL])/2.0;
          XX:=DXX*CON;
          YY:=DYY*CON;
    /*CC--INITIALIZING-ELEMENTS-OF-R(JL,KL)[IL]---CC*/
          FOR-J:=1-TO-JL-DO
          READ-UNIT(1)(R(J,1)[1:IL])#6E13.6#;
          R(J,1)[1:IL]:=6371000.0+R(J,1)[1:IL];
          R(J,KL)[1:IL]:=6371000.0+TROP[J]*1000.0;
          TS1:=R(J,KL)[1:IL]-R(J,1)[1:IL];
          ZZZ:=0.0;
          FOR-K:=2-TO-KL-DO
            ZZZ:=ZZZ+Z[K];
```

```
            R0:=R(J,1)[1:IL]+ZZZ*TS1;
        ENDF:ENDF;
    /*ESTIMATING THE COEFFICIENTS OF CONITUINY EQUATION */
        NL:=0;
        THETA:=-90.0;
        FOR JJ:=2 TO JL-1 DO
        THETA:=THETA+DYY;
        TH:=THETA*CON;
        FOR K:=1 TO KL DO
        NL:=NL+1;
        TS1:=R(J,KL)[1:IL]-R(J,1)[1:IL];
        RR1[1]:=R(J,K)[2]-R(J,K)[IL];
        RR1[IL]:=R(J,K)[1]-R(J,K)[IL-1];
        RR1[2:IL-1]:=R(J,K)[3:IL]-R(J,K)[1:IL-2];
        RR2[1:IL]:=R(J+1,K)[1:IL]-R(J-1,K)[1:IL];
        IF(K=1)THEN
            A1:=(R0-2.0*TS1*ZZ)/R0;
            A2:=-TS1*ZZ/(2.0*R0*COS(TH)*XX);
            A3:=-TS1*ZZ/(2.0*R0*YY);
            A4:=-2.0*YY*A3*TAN(TH);
        ENDI;
        IF(K>1) THEN
            A1:=-4.0*TS1*ZZ/R0;
            A2:=-TS1*ZZ/(R0*COS(TH)*XX);
            A3:=-TS1*ZZ/(R0*YY);
            A4:=-2.0*YY*A3*TAN(TH);
        ENDI;
        A5:=RR1[1:IL]/(2.0*R0*COS(TH)*XX);
        A6:=RR2[1:IL]/(2.0*R0*YY);
    ENDF:ENDF;
        ENDP;
        ENDM;
EOI ENCOUNTERED.
/
```

TD-9608 + TD-9609

U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
NATIONAL METEOROLOGICAL CENTER

OFFICE NOTE 84

Labels for NMC 360/195 Data Fields

Automation Division Staff

JULY 1973

Office Note No. 84

Each NWS data field is prefixed with a 384-bit label. The first 192 bits contain values which indicate the quantity, the surface or layer, and the time of the data being labeled. The next 64 bits contain the initial time, the number of data points, and generating code information. The remaining bits hold, among other things, scale factors partinent to packed data. Appendix C shows each entry and its location within the label.

All data fields can be thought of as some quantity, Q on a horizontal or quasi-horizontal surface S at some level of value L. If the quantity Q is by its nature, a layer-defined quantity, the limits of the layer are given by S_1 at L_1 and S_2 at L_2. Code figure for Q and S are both given in Table 1.

The numerical value of the level L for the corresponding surface S is composed of two numbers: an integer C and a power of ten with the exponent E such that

$$L = C \times 10^{E},$$

Both C and E are signed integers (sign and magnitude). The space allocated for C in the identifier is 20 bits which is large enough to accomodate the binary equivalent of any 5-digit decimal integer. The convention which must be followed in order to provide a unique bit configuration for any given level is as follows:

(a) C must be a 5-digit decimal integer whose leading digit (the highest order digit) must be nonzero, unless L is zero,

Once C is determined, the value of the exponent E naturally follows such that the resulting L will have its true value in the standard units given in Table 1. For example, if L is 500 mb

$$C = 50000$$
$$E = -2$$
$$50000 \times 10^{-2} = 500.00$$

or if L is 0.7 mb

$$C = 70000$$
$$E = -5$$
$$70000 \times 10^{-5} = .70000$$

Appendix B contains the hexadecimal equivalents of commonly used values of C.

Table 1a gives the values for the marker N which is used for identifying different spectral quantities.

Table 2 gives the values for the marker t which is used for identifying fields involving time and explains how F_1 and F_2 are to be used for each value of t.

Table 3 gives the values for the marker m which is used for identifying fields involving layers.

Table 4 gives value to be used for the marker X. In general, this marker will be non-zero only for fields in a guess file.

Table 4a gives the values for the markers CM and CD which identify climatological data.

Table 5 gives the values for the markerś K which identifies the grid to which these data apply.

Table 5a gives the values for the markers KS which indicates if the grid was derived from spectral or other special methods.

The initial hour I of the forecast (or the observation time on which the analysis is based) is entered in accordance with the 24-hour clock GMT to the nearest hours. For example: I = 00 for midnight GMT and I = 12 for noon GMT. Y is the year within the century. For example, for the year 1973, Y - 73. M is the month of the year, and D is the day of the month.

Table 6 gives the values for the marker R which identifies the run within the cycle.

Table 7 gives the values for the marker G assigned to the program which generated the field.

Each datum point value Q on a surface S (Qs) is scaled according to the equation .

$$\hat{Qs} = (Qs - A)/2^n.$$

Qs must be in the standard units given in Table 1.

The procedure followed for scaling is to scan the data to find the maximum and minimum values of Qs. The value of A is set equal to [(Qs) max. + (Qs) min.]/2. An integer n is then chosen such that 2^n is the smallest value which satisfies the condition $([(Qs)\ max-A]/2^n)<1$. This procedure allows maximum accuracy to be maintained in packing the data.

The mid-range value A is placed in the identifiers as a IBM 360 real number, i.e., 8-bit sign and exponent and 24-bit fraction. The scaling value n is inserted as an IBM 360 half-word integer, i.e., 16-bit integer, 2's complement if negative.

The scaled quantity $\hat{Qs}$, an unnormalized fraction, is packed into 16 bits using 2's complement if negative.

TABLE 1 -- Grid 3

Parameters and Surfaces

Number Hexadecimal	Number Decimal	Abbreviation*	Parameter Name	Standard Unit
			Height w/r to mean sea level	
1	1	-HGT--	geopotential	geopotential meter
2	2	-P-ALT	pressure altitude	geopotential meter
			Distance w/r to earth's	
6	6	-DIST-	surface geometric distance	meter
			Pressure	
8	8	-PRES-	atmospheric pressure	mb
			Temperature	
10	16	-TMP--	atmospheric temperature	degree K
11	17	-DPT--	dewpoint temperature	degree K
12	18	-DEPR-	dewpoint depression	degree K
13	19	-POT--	potential temperature	degree K
14	20	-T-MAX	maximum temperature	degree K
15	21	-T-MIN	minimum temperature	degree K
			Vertical Motion	
23	40	-V-VEL	vertical velocity $(\frac{dp}{dt})$	mb/sec
29	41	-ISENV	net vertical displacement	mb
2A	42	-DZDT-	vertical velocity $(\frac{dz}{dt})$	meter/sec
2B	43	-ORGW-	orographical component $(\frac{dz}{dt})$	meter/sec
2C	44	-FRCVV	frictional component $(\frac{du}{dt}, \frac{dv}{dt})$	meter/sec
			Wind	
30	48	-U-GRD	u component of wind (with respect to grid)	meter/sec
31	49	-V-GRD	v component of wind (with respect to grid)	meter/sec
32	50	-WIND-	wind speed	meter/sec
33	51	-T-WND	thermal wind speed	meter/sec
34	52	-VW-SH	vertical speed shear	sec $^{-1}$
35	53	-U-DIV	divergent u component (with respect to grid)	meter/sec
36	54	-V-DIV	divergent v component (with respect to grid)	meter/sec
37	55	-WDIR-	direction from which wind is blowing w/r to north	degree
38	56	-WWND-	westerly component of wind	meter/sec
39	57	-SWND-	southerly component of wind	meter/sec
3A	58	-RATS-	ratio of speeds	dimensionless
3B	59	-VECW-	vector wind (spectral)	not applicable
3C	60	-STAC-	stendiness factor	percent

Number Hazeltine	Number Control	Abbreviation	Parameter Name	Standard Unit
			Fluid Flow Functions	
48	72	-ABS-V	absolute vorticity	sec^{-1}
49	73	-REL-V	relative vorticity	sec^{-1}
4A	74	-DIV--	divergence	sec^{-1}
50	80	-STR-	stream function	$meter^2/sec$
51	81	-V-POT	velocity potential	$meter^2/sec$
			Moisture	
56	88	-R-H--	relative humidity	percent
59	89	-P-WAT	precipitable water	$kg/meter^2$
5A	90	-A-PCP	accumulated total precipitation	meter
5B	91	-P-O-P	probability of precipitation	percent
5C	92	-P-O-Z	probability of frozen precipitation	percent
5D	93	-SNO-D	snow depth	meter
5E	94	-ACPCP	accumulated convective precipitation	meter
5F	95	-SPF-H	specific humidity	dimensionless
60	96	-L-W20	liquid water	dimensionless
			Stability	
70	112	-LFT-X	lifted index	degree K
71	113	-TOTOT	total totals	degree K
72	114	-K-X--	k-index	degree K
73	115	-C-INS	convective instability	degree K
			Wave Components	
78	120	-L-WAV	long wave component of geopotential	geopotential meter
79	121	-S-WAV	short wave component of geopotential	geopotential meter
			Miscellaneous Surfaces	
80	128	------	Mean Sea Level	dimensionless
81	129	------	Earth's Surface	dimensionless
82	130	------	Tropopause	dimensionless
			Sigma Domain	
90	144	------	boundary	dimensionless
91	145	------	troposphere	dimensionless
92	146	------	stratosphere	dimensionless
93	147	------	quiet cap	dimensionless

Number Hexadecimal	Number Decimal	Abbreviation*	Parameter Name	Standard Unit
			Miscellaneous Parameters	
A0	160	-DRAG-	drag coefficients	dimensionless
A1	161	-LAND-	land/sea	dimensionless
A2	162	-FACT-	K factors (700mb to 500mb normal ratio)	dimensionless
A3	163	-1000-	conversion constants (1000mb to sea level pressure)	dimensionless
A4	164	-700-	sea level pressure specification from 700mb heights	mb/meter
A5	165	-REGP-	regression coefficients for probability of precipitation	percent/meter
A6	166	-REGT-	regression coefficients for mean temperature	degree K/meter
A7	167	-REGP-	regression coefficients for mean precipitation	dimensionless
A8	168	-ORTHP-	orthogonal pressure fluation	mb
A9	169	-ALBDO	albedo	dimensionless
AA	170	-ENFLX	energy flux	watt/m^2
AB	171	-HEATR	heating rate	degree K/sec
A0	176			
B1	177			
B2	178			
B3	179			
B4	180			
B5	181			
B6	182			
B7	183			
			Oceanographic Variables	
180	384	-WTMP-	water temperature	degree K
181	385	-WVHGT	height of wind driven waves	meter
182	386	-SWELL	height of sea swells	meter
183	387	-WVSWL	combined height of waves and swells	meter
184	388	-WVPER	period of wind driven waves	second
185	389	-WVDIR	direction from which waves are moving (w/r to north)	degree
186	390	-SWPER	period of sea swells	second
187	391	-SWDIR	direction from which sea swells are moving (w/r to north)	degree

* Abbreviations are 6 characters. A dash (-) is used to indicate a blank.

TABLE 10.

II

Spectral Quantity Variable
(4 bits)

II	[illegible]
0	Not to be assigned
1	Spectral specification
2	Zonal coefficient
3	Spectral amplitude
4	Spectral phase angle

Octet 2

Time Indicator t (4 bits)

t	MEANING	F_1	F_2
0	Indicates the field is instantaneous; e.g., a 500 mb height forecast: φ_3	Forecast hour (tau)	0
1	Indicates the field is formed from two fields of the same type whose valid times are equal but whose forecast hours (taus) may or may not be equal; e.g., difference between two analyses (taus equal) of the same parameter; difference between a forecast and the verifying analysis of the same parameter (taus unequal): $(\varphi_3)_2 - (\varphi_3)_1$	Forecast hour (tau) of $(\varphi_3)_2$	Tau of $(\varphi_3)_2$ minus tau of $(\varphi_3)_1$
2	Indicates the field is formed from two fields of the same parameter whose forecast hours (taus) are equal and whose valid times are unequal; e.g., a tendency field formed by differencing two analyses of the same parameter which are 12 hours apart: $(\varphi_3)_2 - (\varphi_3)_1$.	Forecast hour (tau) of $(\varphi_3)_2$ and $(\varphi_3)_1$	Valid time of $(\varphi_3)_2$ minus valid time of $(\varphi_3)_1$
3	Indicates the field is formed from two fields of the same parameter whose forecast hours (taus) are unequal and whose initial times are equal; e.g., a forecast tendency field: $(\varphi_3)_2 - (\varphi_3)_1$.	Forecast hour (tau) of $(\varphi_3)_2$	Forecast hour (tau) of $(\varphi_3)_2$ minus forecast hour (tau) of $(\varphi_3)_1$
4	Indicates the field is formed from a number of fields of the same parameter to obtain average or normal values. If the average applies to a number of days, f_1 is used to indicate this number; and if the average applies to a number of years, f_2 is used to indicate that number.	Days used in average or 0	0 or years used in average

TABLE 2 - continued

5	Indicates the field is non-instantaneous and applies during some time period; e.g., a field of forecast probability of precipitation during some time period.	Forecast hour (tau) at end of period.	Tau at ending of period minus tau at beginning of period.

TABLE 3

n

LEVEL DIFFERENCE MAPPER

(4 Bits)

n	MEANING
0	Indicates S_2 and L_2 are not applicable.
1	Indicates a field is formed by taking the value of Q at S_1 minus the value of Q at S_2.
2	Indicates a field of Q for the layer bounded by S_1 and S_2 not covered by category n = 1.

TABLE 4

X
EXCEPTION NUMBER
(6 Bits)

X	MEANING
0	Indicates the initial time of the field is the same as the current date of the observation cycle.
1	Indicates the initial time of the field is six hours prior to the current date of the observation cycle.
2	Indicates the initial time of the field is twelve hours prior to the current date of the observation cycle.
3	Indicates the initial time of the field is eighteen hours prior to the current date of the observation cycle.
4	Indicates the initial time of the field is twenty-four hours prior to the current date of the observation cycle.
5	Indicates the initial time of the field is thirty hours prior to the current date of the observation cycle.
6	Indicates the initial time of the field is thirty-six hours prior to the current date of the observation cycle.
.	.
.	.
.	.
254	Indicates the initial time of the field is one thousand five hundred twenty-four hours prior to the current date of the observation cycle.
255	Not applicable. The exception number (X) does not apply for this data or the value of X is greater than 254.

Table 1a

CM and CD

CHRONOLOGY MARKERS

(8 bits each)

(MONTH-HOUR)

CM	MEANING
00	Not applicable.
01	JAN 0000 GMT
02	FEB 0000 GMT
.	.
.	.
.	.
12	DEC 0000 GMT
13	JAN 1200 GMT
14	FEB 1200 GMT
.	.
.	.
24	DEC 1200 GMT

(DAY OF MONTH)

CD	MEANING
00	Not applicable
01	First
02	Second
.	.
.	.
.	.
31	Thirty-first

TABLE 5

K

GRID TYPE NUMBER
(8 bits)

K	MEANING
0	1977 - point octagonal grid (Northern Hemisphere) (Grid increment = 381 km)
1	1679 - point grid (73 x 23) Tropical (Grid increment = 5° longitude)
2	1752 - point grid (73 x 24) Tropical (Grid increment = 5° longitude)
3	3021 - point grid (53 x 57) Northern Hemisphere (Grid increment = 381 km)
4	510 - point ED (U,V,T) grid
5	3021 - point (53 x 57) fine mesh grid (North America) (Grid increment = 190.5 km)
6	1977 - point fine mesh octagon (North America) (Grid increment = 190.5 km)
7	2329 - point (53 x 53) octagon (North America)
8	5104 - point facsimile grid (116 x 44)
9	143 - U. S. cities for Max/Min Temp.
10	216 - U. S. cities for Probability of Precipitation
11	286 - U. S. cities for Precipitation Amount
12	1702 - point tropical grid (74 x 23) (Grid increment = 5° longitude)
13	576 - point longitude/latitude grid (Grid increment = 10°)
14	108 - U. S. station array
15	40 - U. S. station array
16	1560 - point grid (39 x 40) eastern half of U. S. (Grid increment = 95.2 km)
17	221 - point grid (13 x 17) centered over U. S. (Grid increment = 381 km)
18	Reserved
19	1977 - point octagon (Southern Hemisphere) (Grid increment = 381 km)
20	2555 - point Mercator grid (45 x 59) (Grid increment = 1½° longitude)
21	1387 - point (73 x 19) longitude/latitude Southern Hemisphere grid (Grid increment = 5°)
22	1387 - point (73 x 19) longitude/latitude Northern Hemisphere grid (Grid increment = 5°)
23	783 - point subset of grid 5 (fine mesh (29 x 27)) Planetary Boundary Layer (PBL) grid
24	651 - point (31 x 21) facsimile grid covering U.S. (Grid increment = 190.5 km)

TABLE 5 CONTINUED

25	3021 - point grid (53 x 57) Southern Hemisphere (Grid increment = 381 km)
26	2385 - point (53 x 45) fine mesh grid (North America) (Grid increment = 190.5 km)
27	4225 - point (65 x 65) Northern Hemisphere grid (Grid increment = 381 km at 60° N)
28	4225 - point (65 x 65) Southern Hemisphere grid (Grid increment = 381 km at 60° S)
29	5365 - point (145 x 37) longitude/latitude Northern Hemisphere grid (Grid increment = 2.5°)
30	5365 - point (145 x 37) longitude/latitude Southern Hemisphere grid (Grid increment = 2.5°)
31	327 - point TDL field composed of a string of 4 grids: U.S. (255 points), Alaska (56 points), Hawaii (12 points), and Puerto Rico (4 points). See TDL Technical Memo 73-5
32	Not Assigned
33	8326 - point (181 x 46) longitude/latitude Northern Hemisphere grid (Grid increment = 2°)
34	8326 - point (181 x 46) longitude/latitude Southern Hemisphere grid (Grid increment = 2°)
255	Not Applicable

TABLE 5a

KS

Spectral Method Marker
(3 bits)

KS	MEANING
0	Not to be assigned
1	Hough spectral method

TABLE 6

R

RUN MARKER
(8 Bits)

R	MEANING
0	New operational run (LFM)
1	First operational run with an observation cycle (RADAT)
2	Second operational run (RAOB)
3	Third operational run (OPNL)
4	Fourth operational run
5	Fifth operational run (10 + 0 FINAL)

TABLE 7

G

CODE NUMBER OF PROGRAM GENERATING D.M.

(8 Bits)

DEC	HEX	NAME OF GENERATING PROGRAM
00	0	Objective analysis
01	1	Barotropic Forecast model described in NMC Office Note 15
02	2	Mesh model 1958 described in NMC Office Note 15
03	3	Mesh model 1964 (with improved terrain) described in Office Note 24
04	4	Reed 1000 mb forecast model described in NMC Tech. Memo 26
05	5	3-level baroclinic forecast model described in NMC Tech. Memo 22
06	6	4-level baroclinic forecast model
07	7	4-layer Primitive Equation (PE) model
08	8	6-layer PE model
09	9	Maximum and minimum temperature forecast
10	A	Sea height and swell forecast
11	B	Tropical Analysis
12	C	Tropical Forecast
13	D	Bat Analysis
14	E	Tropical Forecast
15	F	Tropical Forecast with Satellite [illegible]
16	10	Sub-synoptic Advection Model
17	11	Compute long wave components
18	12	Trajectory Forecast
19	13	Limited-area Fine Mesh [illegible]

TABLE 7 CONTINUED

20	14	Limited-area Fine Mesh Forecast
21	15	Perfect Prog Precipitation Forecast
22	16	Hough Analysis
23	17	NCFA - Eddy Analysis & Sanbar Forecast
24	18	NMRC/NCAR Climatology data
25	19	Snow cover
26	1A	Planetary boundary layer analysis and forecast
27	1B	Extended Forecast data processor
28	1C	PE and Trajectory Model Output Statistics
29	1D	8-layer spheral ("global") PE model (5° mesh)
30	1E	8-layer northern hemispheral ("hemiglobal") PE model (2.5° mesh)

APPENDIX B

Hexadecimal Equivalents for Commonly Used Values of C

Decimal	Hexadecimal
10000	02710
15000	03A98
15240	03B88
18250	04772
20000	04E20
21340	05360
25000	061A8
27315	06AB3
27430	06B26
30000	07530
30480	07710
33333	08235
35000	08BB8
36580	08EE4
40000	09C40
42570	0A64A
45000	0AFC8
50000	0C350
55000	0D6D8
60000	0EA60
65000	0FDE8
65667	10083
70000	11170
75000	124F8
80000	13880
85000	14C08
90000	15F90
91400	16508
95000	17318
99000	182E8

U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
NATIONAL METEOROLOGICAL CENTER

OFFICE NOTE 184

Procedures for Creating or Unpacking FGGE Level III Data Sets in International Exchange Formats

Armand J. Desmarais
Development Division

JULY 1978

Wanda,
See exceptions for your type.

Procedures for Creating or Unpacking FGGE Level III Data Sets

General

The documentation for FGGE Level III formats is described in Appendix 11, FGGE International Data Management Plan, "Format for the International Exchange of Level III Data Sets During the FGGE."

Purpose

This note describes the basic steps needed for creating or unpacking Level III data sets with the use of attached subroutine listings. These subroutines were written for use on IBM 360 machines that have 32-bit words. Some modifications may be necessary to utilize these routines on other machines.

Level III Tape Files

File 1 — TEST file

File 2 — TAPE HEADER file

File 3 — GRID DESCRIPTOR file

Files 4-n — LEVEL III DATA file(s)

To Create Level III Magnetic Tape

The basic sequence of events in a computer program should be as follows, assuming that the required analysis fields are available in an analysis file and are ready for processing:

A. Write the required TEST file on output tape (File 1).

B. Screen available analyses to determine first and last dates to be processed and written on the output tape. Prepare the documentation for the TAPE HEADER file with appropriate dates (File 2), and write it on the output tape (See example, Attachment 1).

C. Describe the grid(s) used for the Level III data (See example, Attachment 1). Write GRID DESCRIPTOR file on output tape.

D. Select an analysis field from the analysis file, prepare the necessary unique identification words (see subroutine W3FI32, Attachment 2), scale and pack the data (see subroutine W3AI00, Attachment 3), and write the packed field in a LEVEL III DATA file on the output tape.

E. Repeat D above for other fields for the given date/time.

F. Write an end-of-file tape mark on the output tape.

G. Repeat steps D, E, and F above for any remaining date/times.

H. Terminate the output tape with at least two (2) consecutive end-of-file tape marks.

not applicable

To Unpack Level III Data

The format required to read FGGE Level III data sets is written in clear text in the second file (TAPE HEADER) of each tape. The third file of each tape will contain information concerning the arrangement of the data for the grid(s) used in the DATA file(s) that start with the fourth file.

ignore

ignore

A. Read a record from selected DATA file.

B. Determine if you desire to process and unpack the record. Subroutine W3FI33 (Attachment 4) could be used to convert the first 256 bits of the record identification to individual field identifiers.

C. If the field is desired, the packed data (beginning at bit 385) can be unpacked and rescaled with the use of subroutine W3AI01 (see Attachment 5). If not desired, repeat A and B above.

A sample FORTRAN program to locate and unpack N. Hemisphere 700 mb heights, and to print out some values over the United States, is given in Attachment 6. Descriptions of built-in and intrinsic functions used in the various subroutines are given in Attachment 7.

APPENDIX 1

EXAMPLE -- TAPE HEADER FILE

```
FGGE3A00317801020078010812
 10780          FIXED LENGTH RECORDS (BLOCKSIZE = RECORD SIZE)
 20A4           USE  ( 100(27A4) )  FOR FULL RECORD
 0123456789=:>  /STUVWXYZ.(-JKLMNOPQR*);+ABCDEFGH.)(<
 9-TRACK,  800 BPI
 BINARY,  ODD PARITY
 IBM 360/195,       32 BITS/WORD,  8 BITS/BYTE.
 NATIONAL METEOROLOGICAL CENTER, NWS, NOAA, WASHINGTON, D.C., USA
  ( 100(27A4) )
  HOUGH FUNCTION ANALYSIS METHOD, GLOBAL      (FLATTERY)
    FIRST GUESS COEFFICIENTS DERIVED FROM A 9-LAYER PRIMITIVE EQUATION
    FORECAST MODEL ON A 2.5 X 2.5 LATITUDE/LONGITUDE GRID.
  100. =  (42640000) HEXADECIMAL            FLOATING POINT REPRESENTATION
  -100. = (C2640000) HEXADECIMAL            FLOATING POINT REPRESENTATION

    ANALYSIS FIELDS NORMALLY PROVIDED:
    . 12 LEVELS OF U- and V-WIND COMPONENTS, TEMPERATURES, AND HEIGHTS
       (AT 1000, 850, 700, 500, 400, 300, 250, 200, 150, 100, 70, AND 50 MB);
      6 LEVELS OF RELATIVE HUMIDITIES (AT 1000, 850, 700, 500, 400, AND 300 MB);
       TROPOPAUSE PRESSURE AND TEMPERATURE (MODELED);  SNOW FIELD;
       SEA SURFACE TEMPERATURE (FROM SATELLITE DATA); SEA LEVEL PRESSURE.

MISSING ANALYSES:     NONE

 ...   END OF TEXT  ....................................
```

this info is ok for your case.

Print data for one file record

EXAMPLE -- GRID DESCRIPTOR FILE

```
GRID 029          RECTANGLE
GEOGRAPHIC
RIGHT-HANDED
GRIDLINES = 037, DELTAJ = 2.50, FIRSTPT=(0.00N, 0.00E)
J=ALL, NI=145,   DELTAI = 2.50
  ...  END OF TEXT  ..................................


GRID 030          RECTANGLE
GEOGRAPHIC
RIGHT-HANDED
GRIDLINES = 037, DELTAJ = 2.50, FIRSTPT=( 90.00S, 0.00E)
J=ALL, NI=145,   DELTAI = 2.50
  ...  END OF TEXT  ..................................
```

basic info ok, too

APPENDIX 2

W3FI32
Pack ID
(Modified for FGGE Level III)

FORTRAN H EXT +
IBM 360
NMC, DCA

PURPOSE:

To convert an array of the 27 data field identifiers into an array of the first 8 identification words. (See Appendix 11, FGGE International Data Management Plan, "Format for the International Exchange of Level III Data Sets During the FGGE."

USAGE:

Calling Statement

```
CALL W3FI32(KARRAY,PKDNT)
```

where,

KARRAY = array containing the 27 data field identifiers (INTEGER*4).
PKDNT = array to receive the 8 identification words.

NOTE 1

(a) If any number n in (KARRAY(I),I=1,27) is erroneously large, W3FI32 will print:
'VALUE IN KARRAY(I)=n ID TOO LARGE TO PACK'

(b) If any number n in KARRAY(I) is erroneously negative, W3FI32 will print:
'VALUE IN KARRAY(I)=n SHOULD NOT BE NEGATIVE'

(c) If either (a) or (b) occurs, that portion of the packed word corresponding to KARRAY(I) will be set to binary ones.

Examples

1. Suppose KARRAY(4)=30. Then W3FI32 will print 'VALUE IN KARRAY(4)=30 IS TOO LARGE TO PACK' and will place ones in the first 4 bits of PKDNT(2).

2. To form PKDNT from the 27 data field identifers for 500 mb height from FMANL (limited-area fine mesh analysis file) dated 00Z March 15, 1974, KARRAY should be initialized as follows:

KARRAY	IDENTIFIER	KARRAY	IDENTIFIER	KARRAY	IDENTIFIER
(1)=1	Q	(10)=0	F_2	(19)=0	Unused
(2)=8	S_1	(11)=0	N^2	(20)=1523	NW
(3)=0	F_1	(12)=0	C_2	(21)=74	JJ
(4)=0	t	(13)=0	E_2	(22)=3	MM
(5)=50000	C_1	(14)=0	CD	(23)=15	YY
(6)=-2	E_1	(15)=0	CM	(24)=0	GG
(7)=0	m	(16)=0	KS	(25)=0	R
(8)=0	X	(17)=5	K	(26)=19	G
(9)=0	S_2	(18)=0	Unused	(27)=3021	J

Then the resulting PKDNT array would be as follows (in hexadecimal):

PKDN (1) 00100800
(2) 00C35082
(3) 00000000
(4) 00000000
(5) 00000005
(6) 000005F3
(7) 4A030F00
(8) 00130BCD

W3FI32

```
      SUBROUTINE W3FI32 (IARRAY,KIDNT)
C     SUBROUTINE W3FI32 PACKS DATA FIELD IDENTIFIERS.
      DIMENSION KIDNT(8),ITABLE(27),IARRAY(27)
      DATA ITABLE/Z00140C01,Z00080C01,Z00000801,Z001C0402,Z01081402,
     1Z01000802,Z001C0403,Z00140803,Z00080C03,Z00000803,Z001C0404,
     2Z01081404,Z01000804,Z00180805,Z00100805,Z00080805,Z00000805,
     3Z001C0406,Z00100C06,Z00001006,Z00180807,Z00100807,Z00080807,
     4Z00000807,Z00180808,Z00100808,Z00001008/
      DATA KX/ZFFFFFFFF/
      DATA MASK/Z000000FF/
C     MAKE KIDENTS=0
      DO 20 I=1,8
      KIDNT(I)=0
   20 CONTINUE
      ISIGN=0
      DO 40 I=1,27
      ISC=ITABLE(I)
      I1=LAND(ISC,MASK)
      I2=LAND( SHFTR(ISC,8),MASK)
      I3=LAND( SHFTR(ISC,16),MASK)
      I4=LAND(SHFTR(ISC,24),MASK)
C     SIGN TEST
      IV=IARRAY(I)
      IF(IV.GE.0)GO TO 12
      IF(I4.NE.0) GO TO 1
      WRITE (6,22) I,IV
   22 FORMAT(///,1X,' VALUE IN KARRAY(',I2,')=',I11,' SHOULD NOT BE NEGA
     /TIVE'///)
      GO TO 30
    1 IV=IABS(IV)
      MSIGN=1
      ISIGN=MSIGN
      K=I2/4
      DO 25 M=1,K
   25 ISIGN=SHFTL(ISIGN,4)
      ISIGN=SHFTR(ISIGN,1)
      IV=LOR(IV,ISIGN)
   12 CONTINUE
C     MAG TEST
      IF( SHFTR(IV,I2).EQ.0)GO TO 100
      WRITE(6,21)I,IV
   21 FORMAT(///,1X,' VALUE IN KARRAY(',I2,')=',I11,' IS TOO LARGE TO PA
     /CK'///)
   30 IV=KX
      IA=32-I2
      IV=SHFTR(IV,IA)
C     SHIFT
  100 KIDNT(I1)=LOR(KIDNT(I1), SHFTL(IV,I3))
   40 CONTINUE
      RETURN
      END
```

W3AI00

```
      SUBROUTINE W3AI00(REAL4,PACK,LABEL)
C
C SCALE AND CONVERT A FIELD OF REAL*4 DATA TO HALF-WORD INTEGERS AND
C PACK TOGETHER WITH LABEL DATA IN FGGE LEVEL III DATA FORMAT.
C
C     CALL W3AI00(REAL4,PACK,LABEL)
C
C WHERE REAL4 = ARRAY OF REAL*4 DATA TO BE PACKED.
C       PACK  = OUTPUT ARRAY. WORDS 1-8 COPY LABEL WORDS 1-8. WORDS
C               9-12 ARE GENERATED, WORDS 13,14,... ARE PACKED DATA.
C               ALLOW 12+(J+1)/2 FULLWORDS, WHERE J IS NUMBER OF POINTS.
C       LABEL =INPUT LABEL DATA. WORD 8, BITS 16-31, MUST CONTAIN THE
C               COUNT OF THE NUMBER OF POINTS IN ARRAY REAL4 FOR THIS
C               ROUTINE TO WORK.
C
C GENERATED DATA TO BE FOUND IN PACK.....
C WORD 9  BITS  0-15: NUMBER OF BYTES IN WHOLE RECORD.
C         BITS 16-31: EXCLUSIVE-OR CHECKSUM OF WHOLE RECORD (EXCEPT
C                     CHECKSUM ITSELF), BY HALFWORDS.
C WORD 10 BITS  0-31: REAL*4 CENTERING VALUE A = THE MEAN OF THE MAX
C                     AND MIN VALUES OF ARRAY REAL4.
C WORD 11 BITS  0-23: ZERO
C         BITS 24-31: INTEGER*2 SHIFT VALUE N, THE LEAST INTEGER SUCH
C                     THAT ABS(X-A)/2**N IS LESS THAN 1 FOR ALL X IN
C                     ARRAY REAL4.  N IF NEGATIVE IS IN 2'S COMPLEMENT
C                     FORM.  VALUE OF N WILL NOT EXCEED 127.
C WORD 12 BITS  0-31: ZERO
C WORD 13 BITS  0-15: FIRST PACKED DATUM
C         BITS 16-31: SECOND PACKED DATUM
C WORD 14 BITS  0-15: THIRD PACKED DATUM
C  . . .
C
C THE BINARY POINT OF THE PACKED DATA IS CONSIDERED TO BE TO THE RIGHT
C OF THE LEFTMOST, OR SIGN BIT.  DATA IS IN 2'S COMPLEMENT FORM IF
C NEGATIVE.  IF PACK IS EQUIVALENCED TO AN INTEGER*2 ARRAY HLF, THE
C DATA MAY BE RECONSTRUCTED AS FOLLOWS:
C     REAL4(J) = HLF(J+24)*2.**(N-15) + A
C WHERE A AND N ARE TO BE FOUND IN WORDS 10 AND 11.
C
      REAL*4 REAL4(1)
      INTEGER*2 LABEL(1),PACK(1),IA(2)
      EQUIVALENCE (A,IA(1)),(X,IX)
C
C TRANSFER LABEL DATA TO WORDS 1-8.  GET WORDCOUNT, COMPUTE BYTES.
C
      DO 10 I=1,16
      PACK(I) = LABEL(I)
 10   CONTINUE
      J = LABEL(16)
      M = J+24
      PACK(17) = M+M
      PACK(18) = 0
C
C FIND MAX, MIN OF DATA, COMPUTE A AND N.
C
      RMAX = REAL4(1)
      RMIN = RMAX
      DO 20 I=2,J
```

```
W3AI00
          RMAX = AMAX1(RMAX,REAL4(I))
          RMIN = AMIN1(RMIN,REAL4(I))
       20 CONTINUE
          A = 0.5*(RMAX+RMIN)
          X = RMAX-A
          N = LAND(SHFTR(IX,24),127)
          N = 4*(N-64)
          IF (TBIT(IX,8)) GO TO 30
          N = N-1
          IF (TBIT(IX,9)) GO TO 30
          N = N-1
          IF (TBIT(IX,10)) GO TO 30
          N = N-1
       30 N = MAX0(-127,MIN0(127,N))
          PACK(19) = IA(1)
          PACK(20) = IA(2)
          PACK(21) = 0
          PACK(22) = N
          PACK (23) = 0
          PACK(24) = 0
C
C NOW PACK UP THE DATA
C
          TWON = 2.**(15-N)
          DO 60 I=1,J
          X = (REAL4(I)-A)*TWON
          K = X+SIGN(0.5,X)
          IF (K.LT.(-32767)) GO TO 40
          K = MIN0(32767,K)
          GO TO 50
       40 K = -32767
       50 PACK(I+24) = K
       60 CONTINUE
C
C      COMPUTE CHECKSUM AND STORE
C
          IXOR = 0
          DO 70 I=1,M
          K = PACK(I)
          IXOR = LXOR(IXOR,K)
       70 CONTINUE
          PACK(18) = IXOR
          RETURN
          END
```

W3FI33
Unpack ID
(Modified for FGGE Level III)

FORTRAN H EXT +
IBM 360
NMC,DCA

PURPOSE:

To convert an array of the first 8 identification words into an array of 27 data field identifiers. (See Appendix 11, FGGE International Data Management Plan, "Format for the International Exchange of Level III Data Sets During the FGGE.")

Calling Statement

```
CALL W3FI33(PKDNT,KARRAY)
```

where,

PKDNT = array containing the 8 identification words.
KARRAY = array to receive the 27 data field identifiers (INTEGER*4).

Example

Suppose the 8 identification words for 500 mb height from FMANL (limited-area fine mesh analysis file) dated 00Z March 15, 1974 are given (in hexadecimal)

PKDNT	(1) 00100800	(5) 00000005
	(2) 00C35082	(6) 000005F3
	(3) 00000000	(7) 4A030F00
	(4) 00000000	(8) 00130BCD

Then the resulting KARRAY array would be as follows:

KARRAY	IDENTIFIER	KARRAY	IDENTIFIER	KARRAY	IDENTIFIER
(1)=1	Q	(10)=0	F_2	(19)=0	Unused
(2)=8	S_1	(11)=0	N	(20)=1523	NW
(3)=0	F_1	(12)=0	C_2	(21)=74	JJ
(4)=0	t	(13)=0	E_2	(22)=3	MM
(5)=50000	C_1	(14)=0	CD	(23)=15	YY
(6)=-2	E_1	(15)=0	CM	(24)=0	GG
(7)=0	m	(16)=0	Ks	(25)=0	R
(8)=0	X	(17)=5	K	(26)=19	G
(9)=0	S_2	(18)=0	Unused	(27)=3021	J

W3FI33

```
      SUBROUTINE W3FI33 (IDNT,LARRAY)
      DIMENSION LARRAY(27),IDNT(8),ITABLE(27),J(6),JS(6)
C     SUBROUTINE W3FI33 UNPACKS THE 8 IDENTIFICATION WORDS INTO AN ARRAY
C     OF 27 DATA FIELD IDENTIFIERS.
      DATA J/Z0000000F,Z000000FF,Z00000FFF,Z0000EFFF,Z000FFFFF,
     /Z00FFFFFF/
      DATA JS/Z00000007,Z0000007F,Z000007FF,Z00007FFF,Z0007FFEF,
     /Z007FFFFF/
      DATA MASK/Z000000FF/
      DATA ITABLE/Z00140C01,Z00080C01,Z00000801,Z001C0402,Z01081402,
     1Z01000802,Z001C0403,Z00140803,Z00080C03,Z00000803,Z001C0404,
     2Z01081404,Z01000804,Z00180805,Z00100805,Z00080805,Z00000805,
     3Z001C0406,Z00100C06,Z00001006,Z00180807,Z00100807,Z00080807,
     4Z00000807,Z00180808,Z00100808,Z00001008/
      DO 50 I=1,27
      ISC=ITABLE(I)
      I1=LAND(ISC,MASK)
      I2=LAND(SHFTR(ISC,8),MASK)
      I3=LAND( SHFTR(ISC,16),MASK)
      I4=LAND(SHFTR(ISC,24),MASK)
      IX=I2/4
      LARRAY(I)=LAND(SHFTR(IDNT(I1),I3),J(IX))
C     TO SEE IS THE NUMBER IS MINUS
      IF(I4.EQ.0) GO TO 50
      IB=LARRAY(I)
      IC=SHFTL(IB,1)
      DO 30 M=1,IX
   30 IC=SHFTR(IC,4)
      IF(IC.EQ.0) GO TO 50
C     ABS. VALUE,THEN MINUS.
      LARRAY(I)=-(LAND(IB,JS(IX)))
   50 CONTINUE
      RETURN
      END
```

APPENDIX 5

W3AI01

```
      SUBROUTINE W3AI01(PACK,REAL4,LABEL)
C
C UNPACK AND FLOAT A FIELD OF PACKED DATA IN FGGE LEVEL III DATA
C   FORMAT ACCORDING TO SPECIFICATIONS IN THE ID WORDS OF THE
C PACKED DATA.  ALSO MOVE THE ID WORDS TO A LABEL ARRAY.
C
C      CALL W3AI01(PACK,REAL4,LABEL)
C
C WHERE PACK  = ARRAY OF ID WORDS AND PACKED DATA TO BE UNPACKED.
C       REAL4 = REAL*4 ARRAY TO RECEIVE THE UNPACKED DATA WHICH WILL
C               BE DE-SCALED ACCORDING TO ID SPECIFICATIONS.
C       LABEL = A 12-WORD ARRAY INTO WHICH THE ID WORDS WILL BE COPIED.
C
C PACK MUST CONTAIN THE FOLLOWING FIELDS.....
C WORD  8 BITS 16-31: NUMBER OF DATA POINTS J.
C WORD  9 BITS 16-31: EXCLUSIVE-OR CHECKSUM OF REST OF ARRAY BY
C                     HALFWORDS.  THUS CHECKSUM OF ENTIRE ARRAY SHOULD
C                    BE ZERO.
C WORD 10 BITS  0-31: REAL*4 DATA OFFSET VALUE A.
C WORD 11 BITS 16-31: INTEGER*2 SHIFT VALUE N, IN 2'S COMPLEMENT FORM
C                     IF NEGATIVE.
C WORD 13 BITS  0-15: DATUM 1
C      BITS 16-31: DATUM 2
C WORD 14 BITS  0-15: DATUM 3
C     ---     ETC    ---
C
C THERE WILL BE 12+(J+1)/2 FULLWORDS IN ARRAY PACK, AND J FULLWORDS IN
C ARRAY REAL4.
C
      REAL*4 REAL4(1)
      INTEGER*2 LABEL(1),PACK(1),IA(2)
      EQUIVALENCE (A,IA(1))
C
C TRANSFER ID WORDS TO LABEL.
C
      DO 10 I=1,24
      LABEL(I) = PACK(I)
 10   CONTINUE
C
C GET WORD COUNT, A, N.
C
      J = PACK(16)
      IA(1) = PACK(19)
      IA(2) = PACK(20)
      N = PACK(22)
      TWON = 2.**(N-15)
C
C UNPACK, CONVERT REAL*4 DATA
C
      DO 20 IM=1,J
      I = J+1-IM
      REAL4(I) = PACK(I+24)*TWON+A
 20   CONTINUE
      RETURN
      END
```

```
REQUESTED OPTIONS: MAP,XREF,LC(80)
OPTIONS IN EFFECT: NAME(MAIN) NOOPTIMIZE LINECOUNT(80) SIZE(MAX) AUTODBL(NONE)
                   SOURCE EBCDIC NOLIST NODECK OBJECT MAP NOFORMAT NOGOSTMT XREF ALC NOANSF NOTERM FLAG(I)

                      FUNCTIONS INLINE ARE:   NONE

                C***    SAMPLE PROGRAM TO READ AND UNPACK FGGE LEVEL III 2.5 X 2.5
                C***   LATITUDE/LONGITUDE ANALYSIS FIELDS,  WRITTEN IN INTERNATIONAL
                C***   EXCHANGE FORMAT.
                C***   THIS SAMPLE SEARCHES FOR AND UNPACKS 700 MB HEIGHTS (NO. HEMIS.)
                C***    AND DISPLAYS VALUES OVER THE UNITED STATES --- FROM 30N TO 50N,
                C***      AND FROM 235E(125W) TO 280E(80W).
                C                  ---  DESMARAIS,  NMC, WASHINGTON.
    ISN 0004          DIMENSION LABEL(12), IBUF(2695), RECT(145,37), IDS(6)
    ISN 0003          LOGICAL*1  LABX(48)
    ISN 0004          DATA IDS / Z00100800, Z01117082, 0, 0,  Z11D /
    ISN 0005          EQUIVALENCE (LABX(1),LABEL(1))
    ISN 0006          NFILES = 1
                C***    ASSUME THAT THE TAPE  IS PRE-POSITIONED.....................
    ISN 0007          DO  500  KK = 1, NFILES
    ISN 0008          N = 0
    ISN 0009        7 N = N + 1
    ISN 0010          READ (10, 3030, ERR=307, END=320)  IBUF
                C***    SEARCH FOR DESIRED FIELD.
    ISN 0011     3030 FORMAT ( 256 (33A4) )
    ISN 0012          DO  12  I = 1, 5
    ISN 0013          IF (IBUF(I) .NE. IDS(I))  GO TO 7
    ISN 0014       12 CONTINUE
                C***    FOUND DESIRED FIELD          (700 MB HEIGHTS)....
    ISN 0016          PRINT 13, (IBUF(I), I=1,12)
    ISN 0017       13 FORMAT ('     FOUND FIELD WITH THE FOLLOWING IDENTS...'// 1X,1229,
                   1////)
                C***    UNPACK FROM IBUF  TO RECT
    ISN 0018          CALL #3A101 (IBUF, RECT, LABEL)
                C
                C***       RECT(1,1)    AT 0 DEGREES N, 0 DEGREES EAST
                C***       RECT(2,1)    AT 0 DEGREES N, 2.5 DEGREES E.
                C***       RECT(3,1)    AT 0 DEGREES N, 5.0 DEGREES E.       ETC.
                C***       RECT(145,1) AT 0 DEGREES N,   0 DEGREES E.
                C***       RECT(1,2)     AT 2.5 DEGREES N, 0 DEGREES E.
                C***       RECT(2,2)     AT 2.5 DEGREES N, 2.5 DEGREES E.   ETC
                C***    RECT(1,37) THRU (RECT(145,37) AT NORTH POLE.
                C
                C***    FOR SOUTHERN HEMISPHERE GRIDS   (5TH IDENTIFICATION WORD = 1E,)
                C***     RECT(1,1) THRU RECT(145,1) WOULD BE AT SOUTH POLE.
                C***     RECT(1,37) AT 0 DEGREES N, 0 DEGREES E.
                C***    RECT(2,37) AT 0 DEGREES N, 2.5 DEGREES E.       ETC
                C
                C***    PRINT SELECTED 700 MB HEIGHTS (METERS)
    ISN 0019          PRINT 21
    ISN 0020       21 FORMAT ('      SELECTED 700 MB HEIGHTS (METERS) OVER THE UNITED ST
                   1ATES. (LOWER LEFT VALUE AT 30N, 125W)',///)
    ISN 0021          JJ = 21
    ISN 0022          DO  40  J = 1, 9
    ISN 0023          PRINT 22, (RECT(I,JJ), I=95, 113)
    ISN 0024       22 FORMAT (1X, 19F6.0,///)
    ISN 0025          JJ = JJ - 1
    ISN 0026       40 CONTINUE
    ISN 0027          PRINT 45, LABX(4), (LABX(I),I=25,28)
    ISN 0028       45 FORMAT (///,'   VALID ',I2,' HOURS AFTER ',I2,'/',I3,'/',I2,
                   1',  ',  I2,'00 GMT.')
    ISN 0029          PRINT 310
    ISN 0030      310 FORMAT (///,'    .....  RAN TO PLANNED EXIT ..........')
    ISN 0031          STOP
    ISN 0032      307 PRINT 308, KK, N
    ISN 0033      308 FORMAT ('  READ ERROR ON UNIT 10, FILE NUMBER ',I2,', RECORD ',
                   1  I3,',     WILL CONTINUE...')
    ISN 0034          GO TO 7
    ISN 0035      320 PRINT 321, KK
    ISN 0036      321 FORMAT (//,' --- HIT END-OF-FILE MARK NUMBER ',I2,' ON UNIT 10.')
    ISN 0037      500 CONTINUE
    ISN 0038          STOP
    ISN 0039          END
```

Wanda, process the 1st record separately, then proceed. You may have to read with non-formatted i/o

FOUND FILLD WITH THE FOLLOWING IDENTS...
00100900 01117082 00000000 000C0000 00000110 00000A87 4E010200 0A0014F5 2A1AC45C 4J829400 00000009 00000000

SELECTED 700 MB HEIGHTS (METERS) OVER THE UNITED STATES. (LOWER LEFT VALUE AT 30N. 125W)

3049. 3057. 3062. 3062. 3056. 3041. 3018. 2985. 2943. 2898. 2856. 2822. 2801. 2789. 2783. 2779. 2775. 2773. 2777.

3033. 3041. 3049. 3056. 3059. 3054. 3039. 3014. 2982. 2945. 2909. 2877. 2853. 2835. 2820. 2808. 2796. 2788. 2786.

3028. 3035. 3043. 3053. 3060. 3061. 3052. 3035. 3011. 2984. 2953. 2922. 2893. 2868. 2848. 2830. 2813. 2800. 2795.

3036. 3042. 3049. 3058. 3063. 3065. 3060. 3050. 3033. 3011. 2984. 2953. 2924. 2898. 2875. 2854. 2836. 2822. 2819.

3052. 3059. 3062. 3065. 3069. 3069. 3066. 3060. 3048. 3030. 3007. 2982. 2956. 2933. 2911. 2891. 2874. 2865. 2868.

3069. 3075. 3077. 3077. 3077. 3075. 3071. 3066. 3058. 3047. 3033. 3016. 2998. 2978. 2959. 2943. 2932. 2929. 2932.

3083. 3088. 3089. 3090. 3089. 3085. 3078. 3070. 3066. 3064. 3061. 3054. 3042. 3027. 3014. 3005. 3000. 2998. 2998.

3096. 3098. 3098. 3100. 3100. 3096. 3088. 3081. 3078. 3080. 3084. 3084. 3080. 3073. 3068. 3065. 3062. 3058. 3054.

3106. 3105. 3104. 3106. 3108. 3108. 3104. 3099. 3096. 3095. 3098. 3103. 3107. 3110. 3111. 3110. 3108. 3105. 3102.

VALID 0 HOURS AFTER 78/ 1/ 2. 000 GMT.

..... RAN TO PLANNED EXIT .....

APPENDIX 7

Built-in Functions

IV = LAND (a, b)

where a,b may be a 1-, 2-, or 4-byte logical integer expression. The value of LAND is obtained by AND-ing the individual bits of the arguments. The resulting value, IV, will be considered to be logical*4, but may be used as an integer.

IV = LOR (a, b)

where a, b may be a 1-, 2-, or 4-byte logical or integer expression. The value of LOR is obtained by OR-ing the individual bits of the arguments. The resulting value, IV, will be considered to be logical*4, but may be used as an integer.

IV = LXOR (a, b)

where a, b may be a 1-, 2-, or 4-byte logical or integer expression. The value of LXOR is obtained by exclusive OR-ing the individual bits of the arguments. The resulting value, IV, will be considered to be logical*4, but may be used as an integer.

IV = SHFTL (J, K)

IV = SHFTR (J, K)

where J is a 4-byte variable,

K is the actual number of bits to be shifted.

The values of SHFTL and SHFTR are obtained by shifting the first argument, J, left or right, respectively, the number of bits specified by the second argument, K. The resulting value, IV, will be considered to be logical*4, but may be used as an integer.

IV = TBIT (A, K)

where A is a variable, 4-bytes or less,

K is the number assigned to the bit to be tested

The value of TBIT is .TRUE. or .FALSE. depending whether bit position K of the variable A is ON or OFF (ON=1, OFF=0). Bit 0 is the leftmost bit of variable A. The result, IV, will be declared as logical*4.

Intrinsic Functions

AMAX1 - Maximum, Real

$A=AMAX1(X_1,X_2,\ldots,X_n)$

where A and X are real *4

AMIN1 - Minimum, Real

$A=AMIN1(X_1,X_2,\ldots,X_n)$

where A and X are Real *4

MAX0 - Maximum, Integer

$L=MAX0(M_1, M_2,\ldots,M_n)$

where L and M are Integer *4

MIN0 - Minimum, Integer

$L=MIN0(M_1,M_2,\ldots,M_n)$

where L and M and Integer *4

IABS - Absolute Value, Integer

L = IABS(M)

where L and M are Integer *4

SIGN -

$Y=SIGN(X_1,X_2)$

where X and Y are Real *4

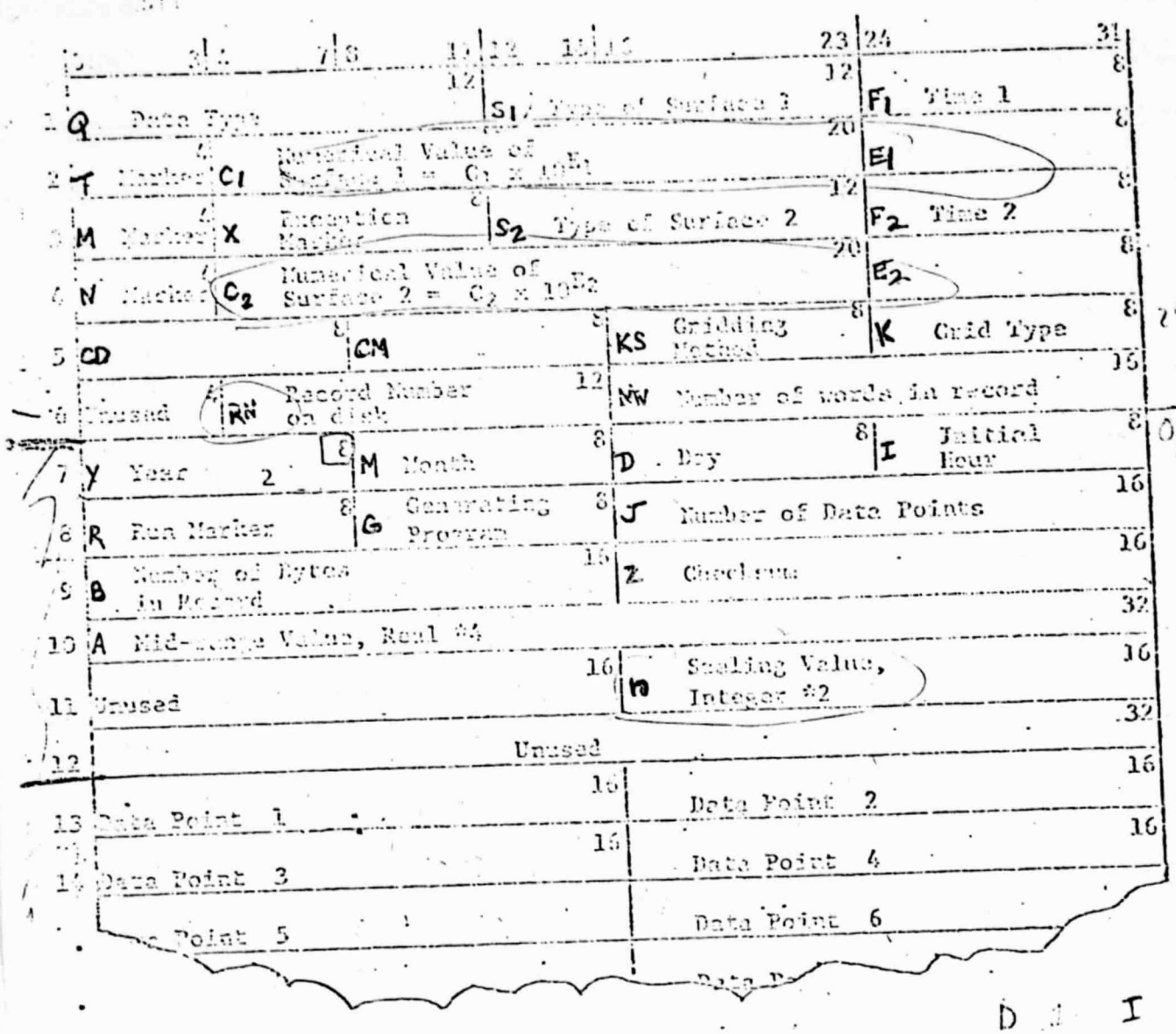
$Y=(\text{sign of } X_2).|X_1|$

C-2

Wascom → 7-2675
for Dr. Peters

APPENDIX C

7-2675
7-1994



360 FORMAT FOR PACKED DATA FIELDS

*Rev. Dec. 1973

10780

National Oceanic and Atmospheric Administration
ENVIRONMENTAL DATA SERVICE
National Climatic Center
Federal Building
Asheville, N. C. 28801

D5231(17136)/SD

September 18, 1978

TO: Mr. Beck
NASA
Langley Research Center
Hampton, VA 23665

FROM: Richard M. Davis, Chief
Computer Products Branch

SUBJECT: Tape Dumps (Your Telcon 9/8/78)

Enclosed are hex tape dumps of the first and last five records from the NMC global and N. Hemisphere tapes for April 1-15, 1978.

Enclosure

DECODE Program Listing

```
/* (SUBRG,STRG,STRZ,SIZE,FOFL,OFL):*/
DECODE:                          /* TO UNPACK NASA WEATHER TAPE */
        PROCEDURE
        OPTIONS(MAIN);


/*
      This program requires no special control cards to decode
      all FGGE level III data packed on a National Meteorological
      Center weather tape. Proper JCL cards must be included,
      however. SYSUT1 and SYSUT2 are the DDnames for the input
      and output datasets respectively. If only a part of the
      data is desired, some program modification will be necessary.
      The basic conversion subroutines may remain unchanged.
      See the writeup, UNPACKING FGGE LEVEL III DATASETS IN
      INTERNATIONAL EXCHANGE FORMATS, for further information.
*/
```

Using Decode

```
DCL 01 RECIN ALIGNED,          /* STRUCTURE FOR INPUT RECORDS */
    05 PART1,                  /* RECEIVES BINARY DIGITS FOR  */
    10 DATA_TYPE BIT(12),      /* CONVERSION TO DECIMAL       */
    10 TYPE_S1   BIT(12),
    10 TIME1     BIT(8),
    10 MARKER1   BIT(4),
    10 C1        BIT(20),
    10 BSIGN     BIT(1),
    10 E1        BIT(7),
    10 MARKER2   BIT(4),
    10 XMARKER   BIT(8),
    10 TYPE_S2   BIT(12),
    10 TIME2     BIT(8),
    10 MARKER3   BIT(4),
    10 C2        BIT(20),
    10 BSIGN2    BIT(1),
    10 E2        BIT(7),
    10 CD        BIT(8),
    10 CM        BIT(8),
    10 KS        BIT(8),
    10 K         BIT(8),
    10 UNUSED    BIT(4),
    10 RN        BIT(12),
    10 NW        BIT(16),
    10 Y         BIT(8),
    10 M         BIT(8),
    10 D         BIT(8),
    10 I         BIT(8),
    10 RMARKER   BIT(8),
    10 G         BIT(8),
    10 J         BIT(16),
    10 B         BIT(16),
    10 Z         BIT(16),
    10 A         BIT(32),
    10 UNUSED2   BIT(16),
    10 N         BIT(16),
    10 UNUSED3   BIT(32),
    05 DATA(5365) BIT(16),
    05 XX        BIT(16);
```

Using Decode

```
DCL 01 TEMPREC UNALIGNED,                  /* STRUCTURE TO RECEIVE     */
    05 PART1,                              /* NUMBERS AFTER CONVERSION */
    10 DATA_TYPE FIXED BINARY(31),
    10 TYPE_S1   FIXED BINARY(31),
    10 TIME1     FIXED BINARY(31),
    10 MARKER1   FIXED BINARY(31),
    10 C1        FIXED BINARY(31),
    10 BSIGN     FIXED BINARY(31),
    10 E1        FIXED BINARY(15),
    10 MARKER2   FIXED BINARY(31),
    10 XMARKER   FIXED BINARY(31),
    10 TYPE_S2   FIXED BINARY(31),
    10 TIME2     FIXED BINARY(31),
    10 MARKER3   FIXED BINARY(31),
    10 C2        FIXED BINARY(31),
    10 BSIGN2    FIXED BINARY(31),
    10 E2        FIXED BINARY(15),
    10 CD        FIXED BINARY(31),
    10 CM        FIXED BINARY(31),
    10 KS        FIXED BINARY(31),
    10 K         FIXED BINARY(31),
    10 UNUSED    FIXED BINARY(31),
    10 RN        FIXED BINARY(31),
    10 NW        FIXED BINARY(31),
    10 Y         FIXED BINARY(31),
    10 M         FIXED BINARY(31),
    10 D         FIXED BINARY(31),
    10 T         FIXED BINARY(31),
    10 RMARKER   FIXED BINARY(31),
    10 G         FIXED BINARY(31),
    10 J         FIXED BINARY(31),
    10 B         FIXED BINARY(31),
    10 Z         FIXED BINARY(15),
    10 A         FLOAT,
    10 UNUSED2   FIXED BINARY(31),
    10 N         FIXED BINARY(31),
    10 UNUSED3   FIXED BINARY(31),
    05 DATA(1387) FIXED BINARY(15),
    05 XX        FIXED BINARY(31);
```

Using Decode

```
DCL 01 OUTREC  UNALIGNED,          /* STRUCTURE RETURNED BY  */
    05 PART1,                      /*          SORT          */
    10 DATA_TYPE CHAR(14),
    10 C1        CHAR(14),
    10 K         CHAR(14),
    10 Y         CHAR(14),
    10 M         CHAR(14),
    10 D         CHAR(14),
    10 I         CHAR(14),
    05 DATA(1387)CHAR(9);
```

```
DCL 01 WRITEREC  UNALIGNED,        /* STRUCTURE TO WRITE OUT */
    05 PART1,                      /*   TO TAPE AFTER SORT   */
    10 DATA_TYPE CHAR(14),         /* WITH DATA CONSOLIDATED */
    10 C1        CHAR(14),         /* FROM 2 RECORDS INTO 1  */
    10 Y         CHAR(14),
    10 M         CHAR(14),         /*  ONE OUTPUT RECORD IS  */
    10 D         CHAR(14),         /*  NOW TWO TIME PERIODS  */
    10 I         CHAR(14),         /*   (ONE DAY) OF DATA    */
    05 DATA(1387)CHAR(9),
    05 DAT2(1387)CHAR(9);
```

Using Decode

```
DCL SYSUT1       FILE RECORD INPUT,       /*   INPUT TAPE FILE   */
    SYSUT2       FILE RECORD OUTPUT,      /* SORTED OUTPUT FILE  */
    SYSPRINT     FILE STREAM OUTPUT;      /*    PRINT  FILE      */


DCL (EOF,BADQ) BIT(1),                    /*       FLAGS         */
    (II,JJ,JJJ,KK,
     LIMIT,
     REC_NUMBER) FIXED BIN(15) INIT(1),
     BLKS  FIXED BIN(31),                 /*    USED FOR COUNTERS   */
     RETCODE FIXED BIN(31,0);             /* RETURN CODE FROM PLISRT*/


DCL (UNSPEC,
     PLIRETC,
     PLISRTD,
     STRING,
     FLOOR)    BUILTIN;                  /*     PLI FUNCTIONS */


ON OVERFLOW BEGIN;
       PUT SKIP(4) EDIT('******  OVERFLOW ERROR ENCOUNTERED  ******')
                                                        (A);
       PUT SKIP(2) EDIT('******  RETURN OUTPUT TO BOB CROVO  ******')
                                                        (A);
       PUT SKIP(2) EDIT('******       FOR   DEBUGGING       ******')
                                                        (A);
       STOP;
       END;

EOF = '0'B;
ON ENDFILE(SYSUT1) EOF = '1'B;
READ FILE(SYSUT1) INTO(RECIN); /* READ 1ST RECORD (HEADER LABEL) INTO
                                  STRUCTURE RECIN TO THROW AWAY      */
```

Using Decode

```
BIG_LOOP:                          /* NUMBER OF TIME PERIODS TO TRANSLATE*/
DO BLKS=1 TO 100000 UNTIL(EOF);/* FOR ACTUAL PRODUCTION RUNS */
/* DO BLKS = 1 TO 2 */             /* FOR DEBUGGING */


READ_LOOP:
/*
   Read remaining records into structure RECIN and do conversion from
   binary to decimal by making the assignment from the binary structure
   (RECIN) to the fixed binary(31) structure (TEMPREC) by name.
   Finally, convert structure to character representation for output
   to tape in character structure OUTREC.
*/


/* PLISRTD calls GETREC 116 times before starting to sort records.
   After sort, PUTREC is used to write to tape. Fields to be sorted
   are YEAR, MONTH, DAY, TIME PERIOD, Q, C Descending, and K (grid),
   also Descending.
*/
CALL PLISRTD(
     ' SORT FIELDS=(43,56,CH,A,1,14,CH,A,15,14,CH,D,29,14,CH,D)',
                   ' RECORD TYPE=F,LENGTH=(12581) ',
                   200000,RETCODE,GETREC,PUTREC,'','AP');

IF RETCODE¬=0 THEN DO; PUT DATA(RETCODE);     /*  BAD SORT  */
                       CALL PLIRETC(RETCODE);
                       STOP;
                       END;


 /*    <<<<<<< THE FOLLOWING STMT IS FOR DEBUGGING ONLY >>>>>>>    */
PUT SKIP EDIT(' HAVING RETURNED FROM PLISRTD WITH RETCODE = ',RETCODE,
   ' RETURN TO TOP OF LOOP AFTER READING DUMMY RECORD')(A,F(12,3),A);
 /*    <<<<<<< REMOVE THE PRECEDING STMT FOR ACTUAL RUNS>>>>>>>    */


READ FILE(SYSUT1) INTO(RECIN); /* READ 1ST RECORD (HEADER LABEL) INTO
                                  STRUCTURE RECIN TO THROW AWAY      */
END; /* OF BIG_LOOP */


PUT SKIP(8) EDIT('******  NORMAL PROGRAM TERMINATION  ******')
                                                    (COL(10),A);
PUT SKIP(9) EDIT('******  ',BLKS, ' RECORDS PROCESSED ******')
                                                    (COL(10),A);
STOP;
```

Using Decode

```
GETREC:                          /* GETREC READS 116 RECORDS BEFORE */
                                 /* CALLING PLIRETC(8) FOR PLISRTD  */
   PROCEDURE RETURNS (CHAR(12581));
   DCL KNT FIXED BIN(15) STATIC INIT (1);
   READ:  IF KNT = 117 THEN DO; KNT = 1;
                                REC_NUMBER = 1;
                                CALL PLIRETC(8);/*LAST RECORD TO SORT*/
                                RETURN(STRING(OUTREC));
                                END;
          BADQ = CONVERT;
          KNT = KNT + 1;
          IF BADQ THEN GOTO READ;    /*       SKIP THIS RECORD       */
          CALL PLIRETC(12);          /* NEED MORE RECORDS FOR SORT */
          RETURN(STRING(OUTREC));
   END; /* OF GETREC */



PUTREC:
   PROCEDURE (STRUCT);          /* STRUCT DEF ON OUTREC MAY BE BETTER */
         DCL STRUCT CHAR(12581);
         DCL Q FIXED BIN(31); /*    USED AS TEMPORARY VARIABLE       */
         DCL NEWCALL BIT(1) STATIC INIT('1'B);
         STRING(OUTREC) = STRUCT;
         Q = OUTREC.DATA_TYPE;
         Q = 48 * (Q=1) + 49 * (Q=2) + 16 * (Q=3) + (Q=4) + 88 * (Q=5);
(NOSTRZ):OUTREC.DATA_TYPE = Q;
         IF NEWCALL THEN DO; NEWCALL = '0'B;
                             REC_NUMBER = REC_NUMBER + 1;
                             WRITEREC = OUTREC, BY NAME;
                             END;
                   ELSE DO; NEWCALL = '1'B;
                             WRITEREC.DAT2 = OUTREC.DATA;
                             WRITE FILE(SYSUT2) FROM(WRITEREC);
                             END;
         CALL PRINT;      /* FOR DEBUGGING ONLY--BEST REMOVE ON RUN */
         CALL PLIRETC(4);
   END; /* OF PUTREC */
```

Using Decode

```
CONVERT:                    /* CONVERT TAPE DATA TO DECIMAL NUMBERS */
   PROCEDURE  RETURNS(BIT(1));   /* 1 FOR FAILURE, 0 FOR SUCCESS */
   DCL Q FIXED BIN(31);          /*  Q IS USED AS TEMPORARY VAR  */



   READ FILE(SYSUT1) INTO(RECIN);     /* READ RECORD INTO
                                            STRUCTURE RECIN */
(NOSTRZ):
   TEMPREC.PART1=RECIN.PART1, BY NAME;/* CONVERT FROM BINARY TO  */
                                      /* DECIMAL BY ASSIGNMENT   */
   IF (TEMPREC.TYPE_S1 ¬= 8) THEN
      RETURN('1'B);                   /*   TEST FOR DESIRED  S   */

   Q = TEMPREC.DATA_TYPE;
   Q = (Q=48) + 2 * (Q=49) + 3 * (Q=16) + 4 * (Q=1) + 5 * (Q=88);
   IF (Q = 0) THEN RETURN('1'B);      /*   TEST FOR DESIRED  Q'S */
   TEMPREC.DATA_TYPE = Q;

   TEMPREC.XX = RECIN.XX;
   UNSPEC(TEMPREC.Z) = RECIN.Z;        /*  TRANSLATE DATA INTO    */
   UNSPEC(TEMPREC.A) = RECIN.A;        /*     THE CORRECT FORM.   */

   IF (RECIN.BSIGN = '1'B) THEN   /* GIVE EXPONENT E1 CORRECT SIGN */
                           TEMPREC.E1 = -(TEMPREC.E1);
   IF (RECIN.BSIGN2 = '1'B) THEN  /* GIVE EXPONENT E2 CORRECT SIGN */
                           TEMPREC.E2 = -(TEMPREC.E2);
   TEMPREC.C1 = FLOOR((TEMPREC.C1 * (10 ** TEMPREC.E1)) + .5);

   KK = 0;

   DO JJ = 1 TO  37 BY 2;          /* TRANSLATE ONLY ALTERNATE ARRAY */
    JJJ = (JJ - 1 ) * 145;
   DO II = 1 TO 145 BY 2;          /*   VALUES INTO DECIMAL VALUES   */
    KK = KK + 1;
    UNSPEC(TEMPREC.DATA(KK)) = RECIN.DATA(JJJ + II);
    TEMPREC.DATA(KK) =
            TEMPREC.A + (TEMPREC.DATA(KK)*(2**(TEMPREC.N-15)));
   END;
   END;
(NOSTRZ):
   OUTREC = TEMPREC, BY NAME;/*CONVERT TO CHARACTER FOR EBCDIC OUTPUT*/

   RETURN('0'B);

   END; /* OF CONVERT */
```

Using Decode

```
PRINT:                                          /* OUTPUT CONVERTED VALUES */
   PROCEDURE;

   /*
      OUTPUT THE RESULT OF THE CONVERSION, GIVING THE
      NECESSARY VARIABLES AND THE VALUES ASSOCIATED WITH THEM.
   */
   PUT PAGE;
   PUT SKIP EDIT ('RECORD NUMBER -- ')     (COL(1),A)
                 (REC_NUMBER)              (F(7));
   PUT SKIP EDIT ('Q =',OUTREC.DATA_TYPE)  (A,F(6))
                 ('C1 =',OUTREC.C1)        (X(24),A,F(6))
                 ('K =',OUTREC.K)          (X(24),A,F(6));
   PUT SKIP EDIT ('Y =',OUTREC.Y)          (A,F(6))
                 ('M =',OUTREC.M)          (X(24),A,F(6))
                 ('D =',OUTREC.D)          (X(24),A,F(6))
                 ('I =',OUTREC.I)          (X(24),A,F(6));
   PUT SKIP(4) EDIT('---------->    DATA FROM RECORD    <----------')
                   (COL(38),A);
   PUT SKIP(2);
   /*
      Output data entries along with the array subscripts, for
      reference.  The following LIMIT may be from 1 to 1387.
      A multiple of 5 is preferred.
   */
   LIMIT = 225;
   DO KK = 1 TO LIMIT BY 5;
      PUT SKIP EDIT ('(',KK,' - ',KK+4,')',
                  OUTREC.DATA(KK),OUTREC.DATA(KK+1),
                  OUTREC.DATA(KK+2),OUTREC.DATA(KK+3),OUTREC.DATA(KK+4))
                   (COL(1),A,F(5),A,F(5),A,5 (A,X(2)));
   END;
   RETURN;
   END; /* OF PRINT  */

END;    /* OF DECODE */
```