A DATA-MANAGEMENT SYSTEM FOR DETAILED AREAL INTERPRETIVE DATA

By Carmelo F. Ferrigno

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations Report 86-4091

Denver, Colorado
1986

UNITED STATES DEPARTMENT OF THE INTERIOR

DONALD PAUL HODEL, Secretary

GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information
write to:

District Chief
U.S. Geological Survey
Water Resources Division
Box 25046, Mail Stop 415
Denver Federal Center
Denver, CO 80225

Copies of this report can
be purchased from:

U.S. Geological Survey
Books and Open-File Reports
Federal Center, Bldg. 41
Box 25425
Denver, CO 80225
[Telephone: (303) 236-7476]

CONTENTS

FIGURES

TABLES

CONVERSION FACTORS

For those readers who prefer to use metric units rather than inch-pound units, the conversion factors for the terms used in this report are listed below:

Multiply inch-pound units	By	To obtain SI units
mile (mi)	1.609	kilometer
square mile (mi^2)	2.590	square kilometer

A DATA-MANAGEMENT SYSTEM FOR DETAILED AREAL INTERPRETIVE DATA

By Carmelo F. Ferrigno

ABSTRACT

A data storage and retrieval system has been developed to organize and preserve areal interpretive data. This system can be used by any study where there is a need to store areal interpretive data that generally is presented in map form. This system provides the capability to grid areal interpretive data for input to ground-water flow models at any spacing and orientation. The data storage and retrieval system is designed to be used for studies that cover small areas such as counties.

The system is built around a hierarchically structured data base consisting of related latitude-longitude blocks. The information in the data base can be stored at different levels of detail, with the finest detail being a block of 6 seconds of latitude by 6 seconds of longitude (approximately 0.01 square mile). This system was implemented on a mainframe computer using a hierarchical data-base management system. The computer programs are written in Fortran IV and PL/1.

This report describes the design and capabilities of the data storage and retrieval system, and the computer programs that are used to implement the system. Supplemental sections of the report contain the data dictionary, user documentation of the data-system software, changes that would need to be made to use this system for other studies, and information on the computer software tape.

INTRODUCTION

The Data-Management System (DMS) described by Luckey and Ferrigno (1982) is designed to be used for large study areas. For example, this system is appropriate for a study of the High Plains aquifer which includes 174,000 mi^2 in parts of eight States. However, if the study area is small, then that system may be a poor choice for managing the study area data and the system described in this report may be a much better choice.

As part of the High Plains Regional Aquifer-System Analysis (RASA), a detailed study of pumpage and return flow was initiated for two areas in Nebraska and Texas. The Nebraska test area consisted of Chase, Dundy, and Perkins Counties. The Texas test area consisted of Castro and Parmer Counties. These test areas were very small in comparison to the entire High Plains aquifer as shown in figure 1. Therefore, it would be useful to store data at a finer detail than is allowed in the original High Plains RASA DMS. This lead to the development of a second DMS that can store data for a 6-second by 6-second block (latitude-longitude extent, approximately 0.01 mi^2). The original DMS can store data for a 1-minute by 1-minute block

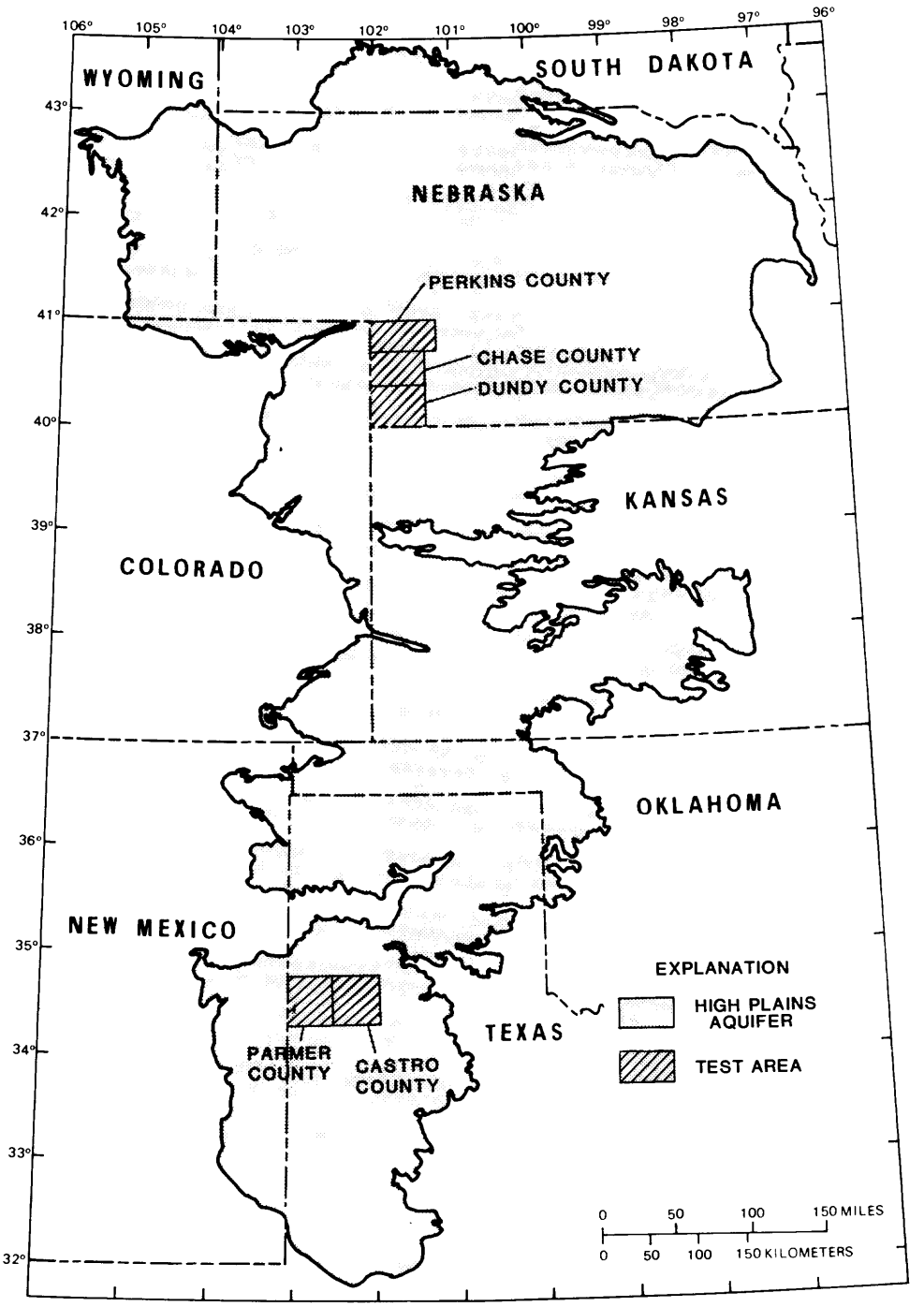

Figure 1.--High Plains aquifer and test areas in Nebraska and Texas.

(approximately 1.0 mi^2). The new DMS is directly modeled after the original system. The DMS, including the structure of the data base and the computer programs for data generation, data storage and retrieval, and data manipulation, can be used for any study where there is a need to store areal interpretive data that generally is presented in map form.

Records in the data base represent information about a rectangular block. These records are then related or grouped in a hierarchical, or inverted tree, structure. A hierarchical data-base structure means that all records are related to one and only one record at each higher level, and there is one specially designed record, called the entry or root record, that is at the highest level of the structure (Martin, 1977).

A schematic representation of the DMS is given in figure 2. Computer programs for data storage and retrieval interact with the hierarchical data base and a Data-Base Management System (DBMS). A DBMS is a group of computer programs for processing organized collections of data. A number of DBMS's for hierarchical data bases are available. When a DBMS is applied to a specific data base, a DMS is the result. The DMS consists of data-preparation programs, programs that interact with the data base and the DBMS, and data-application programs. This DMS was implemented on an IBM mainframe using the System 2000[1] DBMS. The programs are written in Fortran IV and PL/1 and make extensive use of IBM Job Control Language (JCL).

A hierarchical data structure is maintained in the data-preparation and data-application programs even though these programs use sequential card-image files for input and output. In fact, the data-preparation and data-application programs are designed so the programs that interact with the data base and the DBMS could be bypassed if necessary. Hence, in the early stages of a study, data preparation and data application can be performed without actually establishing a data base. The data-preparation programs can then be used to bring all the available data into a common format and the data-application programs can be used to check the consistency of the data.

The major use of the DMS is to organize and preserve data and to have the data easily available for analysis. The DMS can be used to transform data retrieved from the data base into input for ground-water flow models (Trescott and others, 1976). The model blocks can be of any size and the grid does not have to be uniform. In addition, the model grid can be at any orientation to the data-base grid. If the data in the data base needs revision, the DMS can be used to temporarily update the data and check the validity of the update before a permanent revision is made or the DMS can be used to directly update the data.

The DMS can be used to generate contour maps and three-dimensional views. The DMS also could be linked to other machine-graphics systems.

---

[1]The use of brand names in this report is for identification purposes only and does not constitute endorsement by the U.S. Geological Survey.

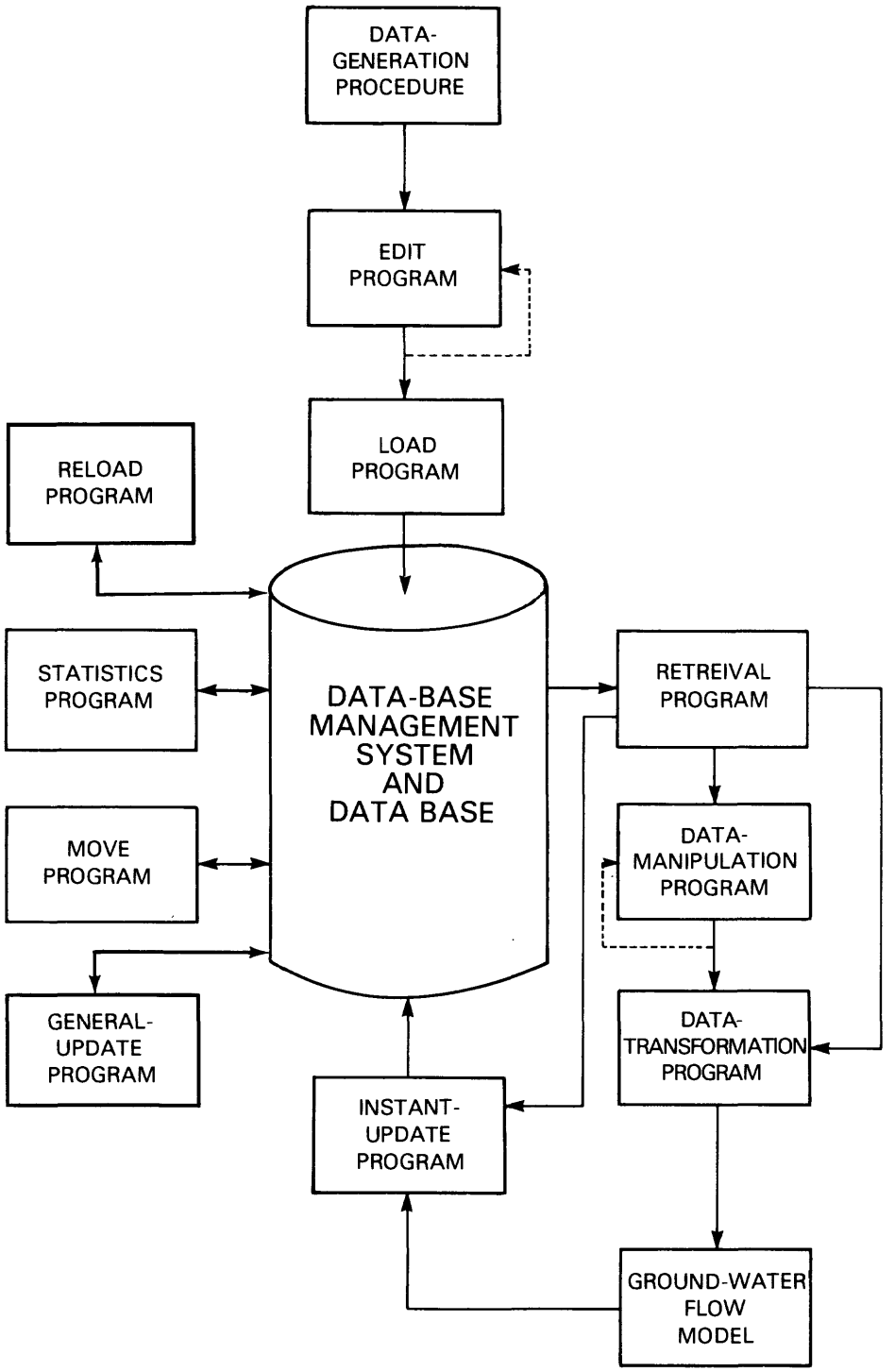

Figure 2.--Flowchart of data-management system.

PURPOSE AND SCOPE

This report provides a description of the DMS. In addition, the report outlines the data-base structure and all major computer programs of the DMS. This information is needed to adapt this system for use by other studies to store areal map information.

The report is in two parts: a main body and supplemental sections. The main body gives a general description of the DMS; the supplemental sections give all of the details needed to use the DMS or adapt it to another study. While the main body of the report needs to be read by all potential users of the system, some users may skip parts of the supplemental sections. The supplemental sections contain the detailed information that is needed to actually implement and use the DMS. In addition, those sections will aid system analysts and programmers who need to adapt part or all of the DMS to another study and other computer systems.

STRUCTURE OF THE DATA-MANAGEMENT SYSTEM

A complete DMS consists of a data base, procedures for generating, loading, updating, and retrieving from the data base, and computer programs to use the retrieved data to accomplish specific tasks. The system described in this report can be used to manage diverse geographic information for small study areas. The geographic information usually starts out in map form and is converted to latitude-longitude-value triplets before being stored in the data base. Such geographic information can be organized easily within a hierarchically structured data base.

Data-Base Structure

To store data within a hierarchical structure, elements are needed that uniquely identify the data stored below the root level. Because the data are needed for a 6-second by 6-second block, the root-level record could consist of components describing the latitude, longitude, and study unit (a data-base component identifying the study area, data-base component 2 in table 1, Supplement I) associated with the node. Such a structure could require a large disk-storage area. This would be the most efficient method of storing the data if all of the data sets are needed in 6-second by 6-second blocks. However, it is possible that some of the source data may not be available in sufficient detail to store information for each 6-second by 6-second block within the study area.

An alternative data-base structure that requires less disk storage consists of root-level records that contain the study unit and parameter names. By using a parameter name in conjunction with a study-unit name, the same parameter could be stored for different study units even if the study units occupy the same geographical area. Using this structure, the data for each parameter would be stored at the levels below the root level. The number of levels are mostly determined by the detail of the source data and the detail of the data as required for the study. If the source data are sparse, then the data would most likely not be stored for each 6-second by 6-second

block. There may be sufficient information to store the data for a 1-minute by 1-minute block, a 10-minute by 10-minute block or a 1-degree by 1-degree block. This leads to a data-base structure where the detail of the stored data can vary. The various levels of detail by which data can be stored are called data densities. Each tree or logical data set would then consist of the data for one parameter and one study unit. With this structure the size of the data base is greatly decreased.

The main factor contributing to the decreased size of the data base is that the latitude and longitude is not stored for each 6-second by 6-second block. Only one coordinate for each 1-minute by 1-minute block is stored and from this coordinate, the latitude and longitude of each 6-second by 6-second block easily can be determined.

Data-Storage Levels

Combining a hierarchically-structured data base with the variable-density concept leads to flexibility in storing data. For the first High Plains RASA DMS (Luckey and Ferrigno, 1982), four data densities were chosen to store the required data: 3-degree data, 1-degree data, 10-minute data, and 1-minute data. This DMS also has four data densities: 1-degree data, 10-minute data, 1-minute data, and 6-second data. The 3-degree data were eliminated and 6-second data were added to allow the storage of more detailed data. This arrangement makes it easier to store data for small study areas. In all instances, the root level of data (level 0) contains, at a minimum, the parameter and study-unit names. These two names uniquely define a logical data set. This root-level record also contains other general information about the parameter. For example, this record contains fields that describe the units of measurement and statistical information. The number of levels below level 0 is determined by the data density. The data densities are numbered 1 through 4, with density 4 having the greatest detail. A schematic representation of the data-base structure for each density is shown in figure 3; the corresponding geographical areas that the data represent are shown in figure 4.

If only one value is needed for each 1-degree by 1-degree block within the study area, then the data are stored at a density of 1. Density-1 data are also referred to as 1-degree data. One-degree data consists of one record at level 0, and a series of level-1 records describing the 1-degree by 1-degree blocks within the study area. The 1-degree data records contain the value and the latitude and longitude of the southeast corner of the 1-degree by 1-degree block. The records also contain four statistical components: the minimum, the maximum, the number of values, and the standard deviation. For 1-degree data, these statistics reflect the data used to compute the single value for the 1-degree by 1-degree block. These same statistical components can be found at the lower levels of the data-base structure.

A density-2 parameter has one value for each 10-minute by 10-minute block within the study area. Density-2 data are also referred to as 10-minute data. The records for each 10-minute by 10-minute block are stored at level 2 of the data-base structure. The 10-minute data records are logically connected to the appropriate 1-degree statistical records. Level 1 contains a record for

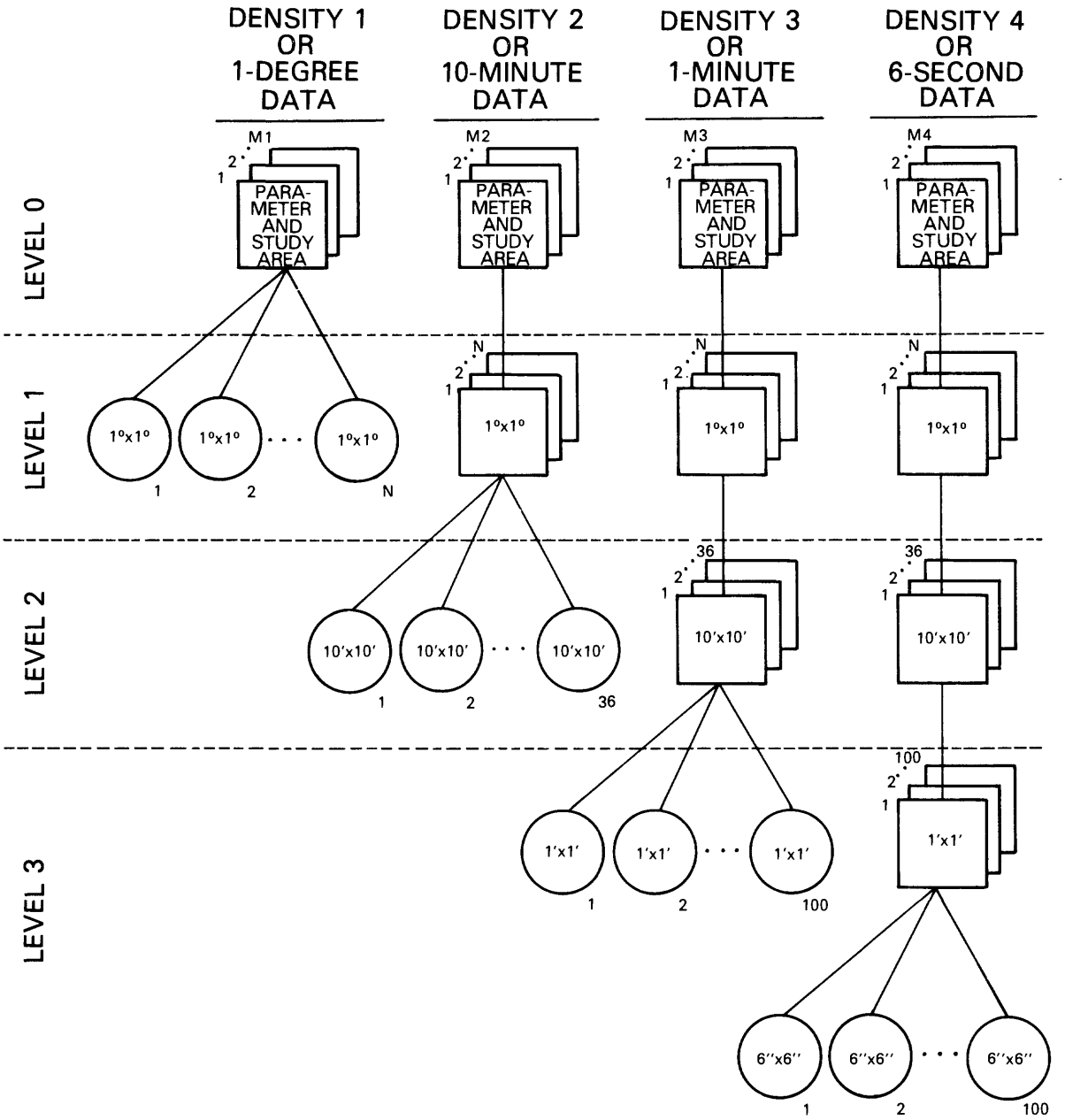

Figure 3.--Schematic showing structure of the data base for the four possible densities.

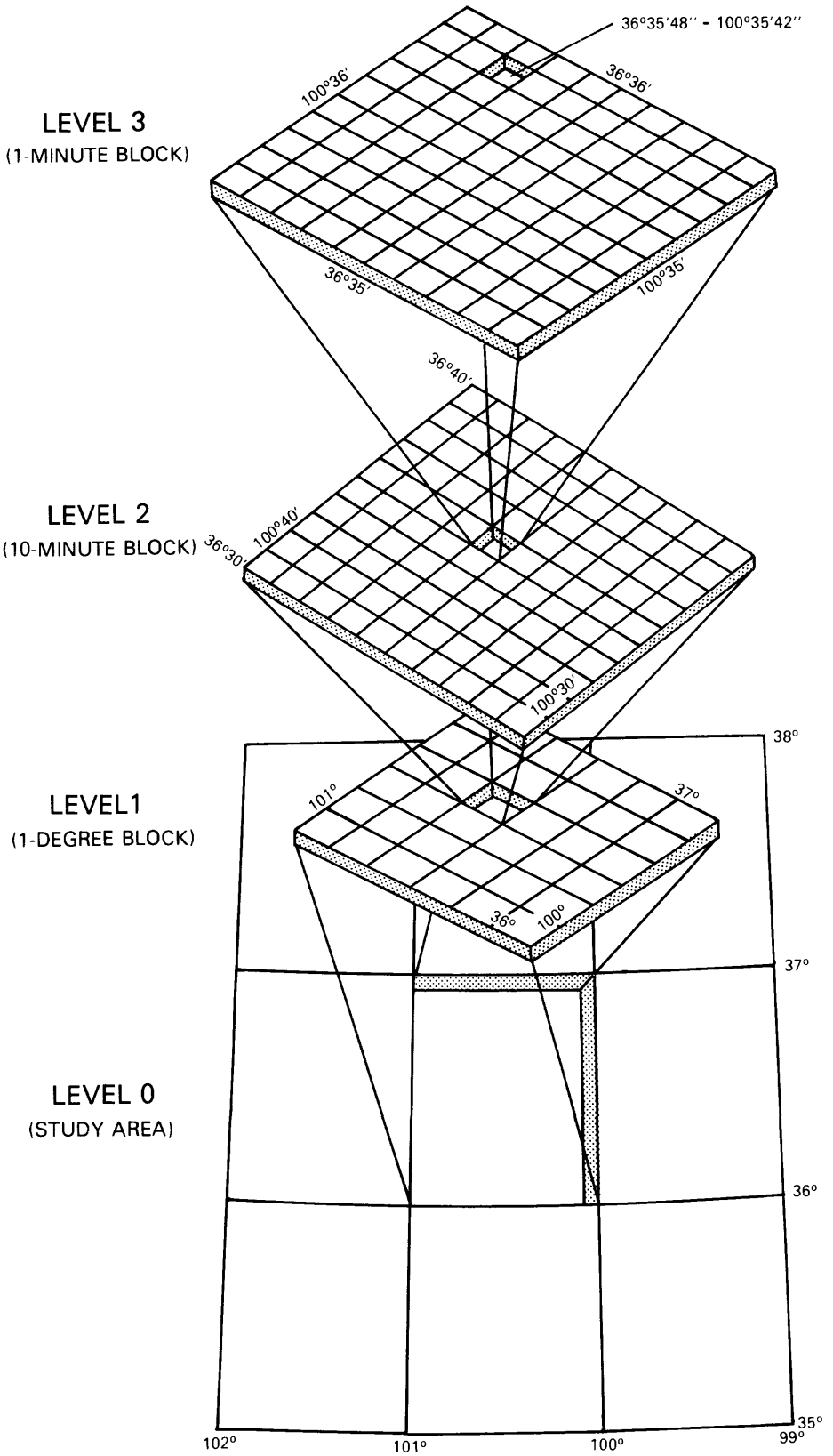

Figure 4.--Geographic area of the data-base levels.

each 1-degree by 1-degree block within the study area. The 10-minute data records contain the latitude and longitude of the southeast corner of the block, the value, and the four statistical components. The four statistical components summarize the data used to compute the single value for the 10-minute by 10-minute block.

A density-3 parameter has one value for each 1-minute by 1-minute block within the study area. Density-3 data are also referred to as 1-minute data. The 1-minute data records are placed at level 3 of the data-base structure. The 1-minute data records are logically connected to the appropriate 10-minute statistical record at level 2, which, in turn, is logically connected to the appropriate 1-degree statistical record at level 1. The 1-minute data record contains the latitude and longitude of the southeast corner of the block, the value, and the four statistical components. Again, these statistics are determined from the data used to compute the single value for the 1-minute by 1-minute block.

A density-4 parameter has one value for each 6-second by 6-second block within the study area. Density-4 data are also referred to as 6-second data. The 6-second by 6-second blocks are grouped with the appropriate 1-minute by 1-minute block. A record for each of the 1-minute by 1-minute blocks is stored at level 3 of the data-base structure. This record contains the latitude and longitude of the southeast corner of the block, and the 100 values for the corresponding 100 6-second by 6-second blocks. The record also contains 5 statistical components, which, in this case, are determined from the 100 values.

Logically, the 6-second data are at a level below the 1-minute data, because they are of greater detail. However, the 6-second data are physically stored at level 3 for two reasons: (1) To simplify the programming, especially those programs that load and retrieve the data; and (2) to decrease the number of logical pointers needed to connect related data. If there were an actual level 4 in the structure, logical pointers would be needed to connect records at levels 3 and 4.

As discussed previously, the data base has a hierarchical structure of four levels with the root-level record containing information that uniquely identifies a particular parameter with a study area through the use of the study-unit name. The lowest level or base level at which a parameter's data resides depends on the density; the actual values will be found only at this base level. The higher levels contain statistical information that reflect the data at the lowest level. These statistics are described later.

Four major factors determine the density at which a parameter is stored: (1) The detail of the data needed for the study, (2) the detail of the source material (usually a map), (3) the distribution or variance of the data over the map, and (4) the geographical extent of the data blocks. A large block is useful for storing data that does not vary greatly; a small block is more useful for a parameter that varies within a short distance. A 1-degree by 1-degree block is approximately 4,000 mi^2 at 32° latitude and 3,500 mi^2 at 42° latitude. A 10-minute by 10-minute block is approximately 111 mi^2 at 32° latitude and 97 mi^2 at 42° latitude. A 1-minute by 1-minute block is

approximately 1.1 mi^2 at 32° latitude and 1.0 mi^2 at 42° latitude. Finally, a 6-second by 6-second block is approximately 0.011 mi^2 at 32° latitude and 0.010 mi^2 at 42° latitude.

Statistics

Actual values are found at the base level of the data-base structure at which a parameter is stored. Statistical components also can be found at this level. These statistics (minimum, maximum, number of values, and standard deviation) are determined from the data used to compute the single value for the block. At levels above the base level, the same statistical components plus the mean value will be found. Statistics at the higher levels are computed and stored in the data base. For example, 1-minute data has its actual values stored at level 3 with the four statistical components. At levels 0 through 2, five statistical components (minimum, maximum, mean, number of values, and standard deviation) can be found. The statistics at level 2 are computed by using the values for all of the 1-minute by 1-minute blocks within the 10-minute by 10-minute block. The statistics at level 1 are computed by using values for all of the 1-minute by 1-minute blocks within the 1-degree by 1-degree block. Finally, a set of statistics are stored at level 0 that reflect all of the 1-minute by 1-minute blocks within the study area. Therefore, the value field can have two meanings: (1) At the base level of a parameter, the value field is the actual value for that block; and (2) at higher levels within the structure, the value field is the mean of the values at the base level within the block.

The main purpose of the statistical information is to have data available for a particular parameter at more than one level of detail. This provides the user of the data with several views of the same information.

Data Updating

An important aspect of a DMS is to provide an efficient method of updating data. There are two methods available to update data within this data base. The first method can be used to directly replace values for any data density. This method should be used when it is certain that the new values are correct. This method is reserved for use by the Data Base Administrator (DBA). The second method is called the proposal concept. A proposal is a separate record that can be added to the data-base structure for a parameter. It consists primarily of a proposed value and the latitude and longitude of the data block. This proposed value is for a block at the base level at which a parameter is stored. For example, a proposed value for 10-minute data would correspond to a 10-minute by 10-minute block. A user can propose a new value for any valid block within the study area.

Proposals reside in a temporary area of the data base; actual data resides in the permanent area of the data base. When a user proposes a new value, it does not directly replace the permanent value. General users of the DMS are not given update authority to the permanent data; replacement of

permanent data is a function of the DBA. This updating process is a compromise between allowing only the DBA to update the data and allowing the general users to update as desired. The DBA periodically updates the permanent data by moving selected proposals to the permanent area of the data base.

System Portability

This system is only directly portable to IBM or IBM-compatible mainframes that have Fortran IV and PL/1 compilers and the System 2000 DBMS. The DMS consists of data-preparation programs, programs that interact with the data base and the DBMS, and data-application programs. The data-preparation programs and the data-application programs all use the same fixed format for input and output. This adds to the portability of the DMS, because, regardless of the DBMS selected, the data-generation and data-application programs are usable provided that the selected computer is an IBM or IBM-compatible mainframe that has compilers for the Fortran IV and PL/1 programming languages. It is also possible that extensive changes would be needed to the IBM JCL depending upon the operating system used by the computer. The programs that interact with the data base and the DBMS would require extensive changes if the DBMS were changed. The program changes would be accomplished more easily if a DBMS based on a hierarchical data-structure were selected because this system was implemented using the System 2000 DBMS which uses a hierarchical structure for storing data. If another type of DBMS were selected (such as a relational DBMS), these program changes would be even more extensive.

DATA GENERATION

Most of the data that are to be stored in the data base probably would originate as data portrayed on some type of map. The maps are usually prepared at various scales and at various levels of detail. After the maps are obtained or constructed, average values for data-base latitude-longitude blocks can be obtained by using either automated or manual procedures.

The automated procedures for generating 10- or 1-minute data are described by Luckey and Ferrigno (1982, p. 15). The programs involved with these automated procedures have not been converted for use with this data base. If necessary, these programs could be easily converted for use with this data base. The programs that are used to generate 10-minute data could be converted to generate 1-minute data for this data base and the programs used to generate 1-minute data could be converted to generate 6-second data for this data base.

For small study areas, a large portion of the data can be generated by a manual procedure. A manual procedure is chosen when the number of data points to be generated is small. For this data base, a manual procedure can probably be used for 10-minute and 1-minute data.

A template showing the 1-minute by 1-minute blocks within a 10-minute by 10-minute block, or the 10-minute by 10-minute blocks within a 1-degree by

1-degree block, is constructed. A separate template is needed for each range of latitude, because width of the blocks decreases to the north. The template is placed over the map and an average value for each block is picked. Average values are placed on a coding sheet that already has the latitude and longitude of the block coded. The data are then ready for data entry and editing.

Data Editing

Data editing starts before beginning the data-generation process and ends just before the data are loaded into the data base. The importance of data editing cannot be overemphasized because the DMS cannot be properly used without careful selection of the data that are entered into the data base. Prior to the start of the data-generation process, the source data are visually examined to determine their accuracy. After the data are generated, the data are carefully machine-edited prior to loading into the data base.

There are two main reasons for carefully editing data:

1. Data should be accurate. To insure accuracy, a manual-editing procedure is used. The data are scanned and checked against source material. In some instances generated data are contoured and then overlaid on the source map.

2. Data needs to be edited to insure that it can be loaded into the data base. The Edit Program is used to check data for loading. There are four main categories of checks performed by the Edit Program: (1) Syntax checks, (2) logical-order checks, (3) duplication checks, and (4) latitude and longitude checks. The syntax checks determine if the records follow standard formats for use in the data-base Load Program. Logical-order checks insure that records are in the proper sequence. Duplication checks determine if the data had been previously loaded into the data base.

Latitude and longitude checks take several forms. Latitude and longitude are checked to see if they are within the study area. Geographic coordinates are examined to determine if the assigned values are appropriate for the record type. For example, the minutes and seconds must be zero if the record describes a 1-degree by 1-degree block. Finally, the coordinates are examined to determine if proper 10-minute data records are grouped with the appropriate 1-degree statistical record, and that 1-minute data records are grouped with the appropriate 10-minute statistical record.

The result of the editing process is a set of data that is ready to be loaded into the data base. The editing process insures that the data represent the original map and can be loaded without problem. Additional information on the Edit Program is provided in Supplement II.

Types of Data to be Stored

The general philosophy in using this DMS is to store only primary data and use computer programs to calculate secondary data. Examples of primary data are water-level altitudes and base-of-aquifer altitudes; secondary data

would be saturated thickness. By not storing secondary data, it is easier to keep the data base internally consistent. For example, if saturated thickness was stored and then modified, that modification would require changes to either the water-level or base-of-aquifer altitude. If other secondary data, such as water-level changes and transmissivity were also stored, the results of the modifications becomes even more complex.

Five general classes of data were stored in the first High Plains RASA data base and these same classes of data could be stored in this data base. The five general classes of data are: (1) Aquifer geometry, (2) aquifer and water characteristics, (3) water levels, (4) climatic data, and (5) land- and water-use data. Table 1 of the report on the first High Plains RASA data base (Luckey and Ferrigno, 1982) contains a list of the parameters that were stored in that data base. It should be noted that it would not be appropriate to store 1-minute data in this data base that has as many points as the 1-minute data stored in the first High Plains RASA data base. In the first High Plains RASA data base, only one latitude and longitude coordinate was stored for each 100 1-minute data points; whereas, in this data base, latitude and longitude coordinates would have to be stored for each 1-minute data point.

The use of this DMS is not confined to the storage of data used for aquifer analysis. The examples used in this report are of data used in the study of aquifers simply because this system was initially developed for use in an aquifer study. This DMS is useful for storing and manipulating any mappable data.

DATA LOADING

The data are loaded into a hierarchical data-base for storage, update, and retrieval. The loading procedure is dependent upon the DBMS. The DBMS stores the data and creates appropriate indices to the data. These indices are useful for updating or retrieving the data. See the documentation for the Load Program, in Supplement II, for information on the loading procedure for this DMS.

DATA UPDATING

There are two methods available for updating the data base. The first method directly updates the data and is reserved for use by the DBA. This method can be used when the correctness of the updates is known. See the documentation for the General-Update Program, in Supplement II, for details about this updating process.

The second method is a two-step process. In the first step, accomplished through the use of the Instant-Update Program, users propose changes to the data. In the second step, the DBA replaces the appropriate permanent-values with the accepted proposed-values using the Move Program. The following describes these two steps in detail.

Proposals

There are three types of proposals that can exist in the data base for each parameter. When a user initially proposes a change to permanent data, a record, called a test proposal, is loaded into the data base. The record contains a flag field that indicates that it is a test proposal. There are three possible actions that the user can perform upon the test proposal. First, the proposal can be left in its present state; in this instance, the test proposal would be removed automatically from the data base after a predetermined amount of time. Second, the user may decide that the test proposal is not acceptable as a replacement for the present permanent-value. The user may then request, through the Instant-Update Program, that the test proposal be rejected. Once rejected, the proposal appears to the user as no longer existing within the data base. The Instant-Update Program does not actually remove the rejected proposal; this function is performed by the Move Program. The third alternative is to accept the test proposal as the replacement for the current permanent-value. The Instant-Update Program is used to flag the test proposal as acceptable, and it then becomes an accepted proposal. The Instant-Update Program does not actually replace the permanent value; this task is performed by the Move Program.

A typical application of this update process is using proposals in conjunction with modeling. As a result of modeling, a user may determine that a group of permanent values is unacceptable. By using the Instant-Update Program, the modeler can propose test values for the unacceptable data. Then, a set of data can be retrieved from the data base for the model area with the request that test proposals be included in the retrieved data. For each latitude-longitude block retrieved that has a corresponding test proposal, the test value will replace the permanent value. This data set can then be used for modeling. While modeling, the user can use the Instant-Update Program to mark the test proposals as accepted or rejected, and possibly add new test proposals.

Thus, through the Instant-Update Program, the user can manipulate test proposals. These proposals can be added to the data base, or their status can be changed to rejected or accepted. Using this method, the general user does not alter the permanent data; hence the integrity of the data base is maintained. Actual changes to the permanent data are performed by the DBA in the second step of the update process. Additional information on the Instant-Update Program is provided in Supplement II.

Move of Proposals

The second step is accomplished by the DBA using the Move Program. For each parameter, the program performs three functions. First, the program locates all the rejected proposals and removes them from the data base. Second, the program locates all the accepted proposals. For each accepted proposal, the program locates the corresponding permanent data-record and replaces the present permanent-value with the new accepted-value. The accepted proposal then is removed from the data base. After all the accepted proposals have been processed, statistics are recomputed for the entire parameter. Third, the program locates all test proposals. Any proposals that have existed for longer than a predetermined amount of time are removed from the data base.

The frequency with which the Move Program is used is dependent on update activity. The DBA is able to monitor this activity through the use of an information file that is produced by the Instant-Update Program. This move process can be done for all parameters or for selected parameters within the data base in one program execution. Additional information on the Move Program is provided in Supplement II.

DATA RETRIEVAL

The retrieval procedure is used to extract a set of data from the data base for use in an application program. There are four specifications required to describe the data to be retrieved. The first two are the parameter and study-unit names. With these two names, the Retrieval Program can pinpoint the location in the data base where the retrieval of data will begin. The third specification is the level of the retrieval. This value determines the lowest level in the data-base structure from which data are to be obtained. Usually this level is the base level at which a parameter resides. For example, the base level for 1-minute data is level 3. Hence, if the actual values are required for 1-minute data, a level-3 retrieval needs to be specified. In some instances, a user may desire only the statistical data for a particular parameter. This can be done by specifying any level above the base level. For example, a user may retrieve the 10-minute statistical records for 1-minute data by specifying a level-2 retrieval. The fourth specification is the area for which data are needed. This area is described as a rectangle where the limits of the rectangle are given as minimum and maximum latitudes and longitudes. This area can be part or all of the study area. Because geographic coordinates are specified as whole degrees, it is likely that the retrieved data will be for an area larger than needed. Available application programs trim the data to the actual area where data are needed.

The Retrieval Program has one major option: the user may request that test proposals be included in the retrieved data. This option only can be used when retrieving data from the base level. With this option, the program searches for test proposals. Prior to outputting a record, the program determines if there is a test proposal present that corresponds to the record. If a test proposal is found, the test value replaces the permanent value in the retrieved data. If, for a particular record, more than one test proposal exists, the most recent test proposal is used.

The format of the retrieved data is identical to the format used to initially load the data into the data base. A discussion of this point can be found in the section on system portability.

The Retrieval Program is used to produce data for use in application programs. It is a very simple retrieval process, in that its only purpose is to extract data that are to be manipulated by another program. This procedure does not have any options to perform complex retrievals. An example of a complex retrieval would be to request retrieval of some data, where the values are greater than a certain value. The DBMS that manages the data base has facilities for performing complex retrievals, but the format of a DBMS retrieval is not compatible with the application programs. Additional information on the Retrieval Program is provided in Supplement II.

DATA-BASE REORGANIZATION

After the data base has been updated many times, the tables used to store the data and other information required by the data base becomes disorganized and can degrade the efficiency of retrievals and updates. The Reload Program can then be used by the DBA to reorganize the tables within the data base. The reorganization of the data base will usually result in the reduction of the total size of the data base and a reduction in computer time required to access data. Additional information about the Reload Program is provided in Supplement II.

APPLICATION PROGRAMS

There are several application programs available that use data retrieved from the data base. The first program produces mathematical combinations of data. Its primary purpose is to generate secondary parameters. The second program is the interface between the data base and ground-water flow models (Trescott and others, 1976). This program transforms data-base data into a format for finite-difference model programs. The third program produces data plots, contour maps, and three-dimensional views of the data.

Data Manipulation

The Data-Manipulation Program is provided to mathematically manipulate data from the data base using five standard functions. The first function takes the logarithm of a value in the form:

$$A*\log_{10}(X) + C$$

where A and C are constants and X is the value. The second function performs an antilogarithm calculation in the form:

$$A*\text{antilog}_{10}(X) + C$$

where A and C are constants and X is the value. The third function performs a linear combination, including addition and subtraction, in the form:

$$A*X1 + B*X2 + C$$

where A, B, and C are constants and X1 and X2 are values for different parameters. The fourth and fifth functions perform multiplication and division in the forms:

$$A*X1*X2 + C$$

$$A*(X1/X2) + C$$

where A and C are constants and X1 and X2 are values for different parameters.

To add flexibility to the program, the user has the option of bypassing the standard functions and inputing the expression that will be used to manipulate the data. This gives the user the opportunity to use many of the standard mathematical functions such as the trigonametric and hyperbolic functions. Detailed information on this option may be found in the documentation for the Data-Manipulation Program.

The primary input data for this program is a set of data in standard retrieval format. This data can be retrieved from the data base or it can be the output from a prior execution of the Data-Manipulation Program. The major use of this program is to generate secondary parameters from the input data. Secondary parameters are data that are produced from parameters already stored in the data base. These stored data are termed primary parameters.

Examples of secondary parameters are change in storage and return flow. To illustrate use of the program, the steps to generate return flow will be discussed. Return flow is the difference between pumpage and change in storage (assuming no natural recharge). The required steps are:

1. Multiply water-level change times specific yield (change in storage);
2. Add up pumpage for desired period; and
3. Subtract change in storage from total pumpage.

These steps result in a secondary parameter, called return flow, in standard load format. In fact, the return-flow parameter could be loaded into the data base at this point.

The secondary parameters could be stored directly in the data base. However, this could result in data-management problems. A secondary parameter, such as change in storage, is dependent on several primary parameters for its values. Hence, if one or more of the primary parameters are changed, then changes in the secondary parameter also would occur. This would result in additional updating of the data base, which would require more effort and time than recalculating the secondary parameter with the Data-Manipulation Program. Also, the size of the data base is maintained at a minimum by not storing secondary parameters.

Another use of the program is to check consistency of primary parameters. For example, by calculating the secondary parameter, depth to water, the relation between water level and land surface can be examined. The secondary parameter, depth to water, can then be compared with a depth-to-water map in order to find inconsistencies between the water level and land surface primary parameters. Additional information on the Data-Manipulation Program is provided in Supplement II.

Data Transformation

The Data-Transformation Program is the interface between the data base and ground-water flow models (Trescott and others, 1976). This program produces a model matrix for a finite-difference model used for aquifer simulation. The program processes data retrieved from the data base. It will process 10-minute, 1-minute, and 6-second data.

The program is divided into two major steps. In step 1, values and their corresponding geographic coordinates are extracted from retrieved data. The longitude and latitude are converted to X and Y values. These cartesian coordinates are then rotated by an angle specified by the user. The result of this step is a group of (X, Y, value) triplets that are in the coordinate system of the model grid. In step 2, values are calculated for each of the nodes of the model. There are three techniques that can be used to produce these values. The user can choose between an average, a weighted-average, or a trend-surface technique (Davis, 1973).

For average and weighted-average techniques, converted data-base data are trimmed to fit the model area. For each (X, Y, value) triplet, the model block in which the point is located is determined. For the average technique, a sum of the data-base values falling within the model block and the number of values within the block are calculated. For the weighted-average technique, the distance between the coordinates of the data-base point and the coordinates of the center of the model block are calculated. The program calculates (1) the data-base value divided by the square of the distance and (2) the reciprocal of the square of the distance. Sums of each of these calculations are kept for each model block. Then, for each model block, an average or weighted-average value is calculated by using the data that are associated with the block. For the average technique, the sum of the data-base values is divided by the number of values in the model block and this result is assigned to that model block. For the weighted-average technique, the sum of the data-base values divided by the square of the distance is divided by the sum of the reciprocal of the square of the distances and this result is assigned to the model block. If some of the model blocks lack data, they cannot be assigned a value. In the trend-surface technique, all converted data are used to calculate the node values and all model blocks are assigned a value. In this technique, the converted data-base data are fit to a polynomial surface and a value is calculated for a model block by solving the polynomial using the coordinates of the center of the model block. For all three techniques, the model matrix is produced in one of two user-selectable formats.

The most important aspect of the Data-Transformation Program is that the model grid is completely independent of the data-base grid. An example of a hypothetical model grid is given in figure 5. There are no restrictions on the finite-difference model grid pattern. The model blocks can be of any desired size and the grid does not have to be uniform. In addition, the model grid can be oriented at any angle to the data-base grid. This gives the user flexibility in choosing the model grid. Additional information on the Data-Transformation Program is provided in Supplement II.

Graphics

The Graphics Program generates contour maps and three-dimensional views of the data. The program is based on computer programs developed by California Computer Products, Inc. (Calcomp). The contour maps and three-dimensional views can be reproduced on Calcomp and Calcomp-compatible flat-bed and drum plotters. The Graphics Program can be used without knowing how the Calcomp computer programs actually function.

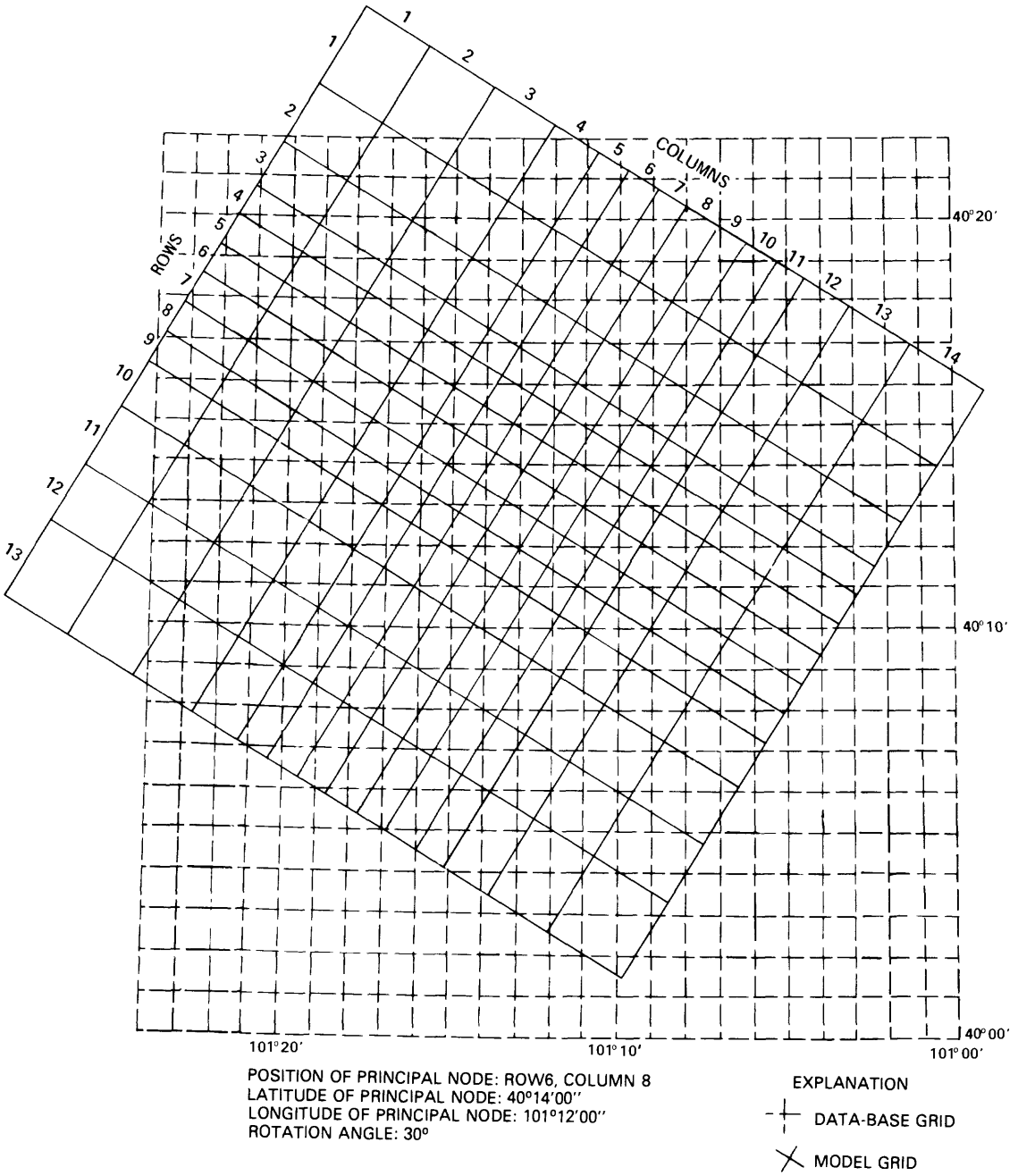

Figure 5.--Hypothetical model grid overlaid on the data-base grid.

To generate contour maps, the user needs to specify: (1) Title of map, (2) rotation angle of map on plotter, (3) width of plotter, and (4) interval between index contours. The Graphics Program then generates the data and commands needed by the Calcomp computer programs to produce the plotter commands.

To generate three-dimensional views, the user specifies the following: (1) Title of plot, (2) position of observer, (3) distance of observer from surface, (4) smoothness of plot, and (5) exaggeration of plot in the Z direction. The Graphics Program then generates the data and commands needed by the Calcomp computer programs to produce the plotter commands. The user has the option of generating multiple views of the same surface in one operation of the program.

The data base was initially linked to the Calcomp computer programs because they were available and usable. The data base can be linked to other available commercial graphics program packages as needed. Additional information on the Graphics Program is provided in Supplement II.

SUMMARY

The High Plains RASA has developed a data storage, retrieval, and update system to organize and manipulate detailed areal interpretive data. This system provides the capability to grid areal data for ground-water flow models at any grid spacing and orientation. The data storage and retrieval system can be adapted for other studies. The system is only directly portable to IBM mainframes that provide the System 2000 DBMS and have compilers for the Fortran IV and PL/1 languages. The system is designed for use with small study areas.

The system is built around a hierarchically-structured data base consisting of related latitude-longitude blocks. Various parameters in the data base can be stored at different levels of detail, with the finest detail being a 6-second by 6-second block (approximately 0.01 mi^2). Statistics about the parameter are stored at all higher levels in the data base, including 1-minute by 1-minute blocks, 10-minute by 10-minute blocks, and 1-degree by 1-degree blocks, so the parameter is available at all levels above the base level. If the data are not available in sufficient detail to store it for each 6-second by 6-second block, then the data may be stored for 1-minute by 1-minute, 10-minute by 10-minute or 1-degree by 1-degree blocks. The system allows direct updating of the data base and a second method of updating the data that provides immediate updates, while preserving the integrity of the permanent part of the data base.

REFERENCES

Davis, John C., 1973, Statistics and data analysis in geology: New York, John Wiley, 550 p.

Luckey, R.R., and Ferrigno, C.F., 1982, A data-management system for areal interpretative data for the High Plains in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Water-Resources Investigations 82-4072, 112 p.

Martin, James, 1977, Computer data-base organization: Englewood Cliffs, N.J., Prentice-Hall, 713 p.

Trescott, P.C., Pinder, G.F., and Larson, S.P., 1976, Finite-difference model for aquifer simulation in two dimensions with results of numerical experiments: U.S. Geological Survey Techniques of Water Resources Investigations, bk 7, chap. C1, 115 p.

SUPPLEMENT I: DATA-BASE DICTIONARY

Within the data base, the data are arranged in a hierarchical structure. At the top of this structure, which is referred to as the root level or level 0, is a group of data items (called the entry record) that uniquely define the data stored at the lower levels of the structure. The study unit and parameter names are part of the entry record. At level 1, data for 1-degree by 1-degree blocks are stored. At level 2, data for 10-minute by 10-minute blocks are stored. At level 3, data for 1-minute by 1-minute blocks are stored. Logically, data for 6-second by 6-second blocks should be stored at level 4, but these data actually are stored at level 3. Proposal and remarks data are maintained at level 1.

Data at each level are stored in groups called schema records. A schema record consists of a group of logically related data items; these data items are referred to as schema or component items. Each schema record is assigned a name and a number and each schema item has a unique component name and component number.

Schema records and the corresponding schema items associated with the data base are listed in table 1. For each schema item, the table indicates the data type. For this data base, five data types are used: integer, decimal, text, character, and date. Both text and character data types are used to store alphanumeric data. When a character item is stored in the data base, the leading, trailing, and extraneous imbedded blanks are removed; for text items, all blanks are retained when the items are stored.

The schema records named LEV3 (for 6-second data) and PROP contain 100 schema items that correspond to the 100 6-second by 6-second blocks within a 1-minute by 1-minute block. All of these schema items are not listed, because the items are identically formatted. Arrangement of the 6-second schema items within the 1-minute by 1-minute block is shown in figure 6.

Table 1.--*Data-base dictionary*

Schema or component		Format		Remarks
Number	Name	Type	Length	
		LEVEL 0 - ENTRY RECORD		
0	ENTRY	-	-	Level-0 schema record
1	AQPARM	Character	38	Concatenation of study-unit and parameter names
2	UNIT	Character	8	Study-unit name
3	PNAME	Character	30	Parameter name
4	STUDY	Character	30	Project name
5	DENSE	Integer	2	Data density
6	SCALE	Integer	3	Characteristic of parameter value (value x 10^{scale})
8	FLAG	Integer	1	Indicates types of existing proposals for parameter
9	LASTD	Date	7	Date of last update (YYYMMDD)
10	LASTT	Integer	6	Time of last update (HHMMSS)
11	MN0	Integer	5	Minimum parameter-value
12	MX0	Integer	5	Maximum parameter-value
13	AVG0	Integer	5	Average parameter-value
14	NUM0	Integer	7	Number of values used to compute statistics
15	SDV0	Decimal	7	Standard deviation - 2 decimal places
17	UNTDSC	Text	10	Description of unit of measurement
21	RESA	Integer	9	Reserved field
22	RESB	Decimal	10	Reserved field - 3 decimal places
23	RESC	Text	10	Reserved field
30	MVI	Integer	5	Missing-value indicator
		LEVEL 1 - REMARKS RECORD		
50	REMARK	-	-	Level-1 schema record - general parameter information - child of entry record
51	RMDATE	Date	7	Date remark was formulated (YYYMMDD)
52	REMSEQ	Integer	3	Remark sequence number
53	REM1	Text	50	First line of remark
54	REM2	Text	50	Second line of remark
55	REM3	Text	50	Third line of remark
56	REM4	Text	50	Fourth line of remark
57	REM5	Text	50	Fifth line of remark

Table 1.--*Data-base dictionary*--Continued

Schema or component		Format		Remarks
Number	Name	Type	Length	
		LEVEL 1 - 1-DEGREE DATA RECORD		
100	LEV1	-	-	Level-1 schema record - child of entry record
101	LAT1	Integer	6	Latitude of southeast corner of 1-degree block (DDMMSS)
102	LONG1	Integer	7	Longitude of southeast corner of 1-degree block (DDDMMSS)
103	LAT1D	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
104	LONG1D	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
111	MN1	Integer	5	Minimum parameter-value
112	MX1	Integer	5	Maximum parameter-value
113	AVG1	Integer	5	Average parameter-value
114	NUM1	Integer	5	Number of values used to compute statistics
115	SDV1	Decimal	7	Standard deviation - 2 decimal places
		LEVEL 2 - 10-MINUTE DATA RECORD		
200	LEV2	-	-	Level-2 schema record - child of 1-degree data record
201	LAT2	Integer	6	Latitude of southeast corner of 10-minute block (DDMMSS)
202	LONG2	Integer	7	Longitude of southeast corner of 10-minute block (DDDMMSS)
203	LAT2D	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
204	LONG2D	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
211	MN2	Integer	5	Minimum parameter-value
212	MX2	Integer	5	Maximum parameter-value
213	AVG2	Integer	5	Average parameter-value
214	NUM2	Integer	5	Number of values used to compute statistics
215	SDV2	Decimal	7	Standard deviation - 2 decimal places

Table 1.--*Data-base dictionary*--Continued

Schema or component		Format		Remarks
Number	Name	Type	Length	
		LEVEL 3 - 1-MINUTE DATA RECORD		
300	LEV3	-	-	Level-3 schema record - child of 10-minute data record
301	LAT3	Integer	6	Latitude of southeast corner of 1-minute block (DDMMSS)
302	LONG3	Integer	7	Longitude of southeast corner of 1-minute block (DDDMMSS)
303	LAT3D	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
304	LONG3D	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
311	MN3	Integer	5	Minimum parameter-value
312	MX3	Integer	5	Maximum parameter-value
313	AVG3	Integer	5	Average parameter-value
314	NUM3	Integer	5	Number of values used to compute statistics
315	SDV3	Decimal	7	Standard deviation - 2 decimal places
		LEVEL 3 - 6-SECOND DATA RECORD		
300	LEV3	-	-	Level-3 schema record - child of 10-minute data record
301	LAT3	Integer	6	Latitude of southeast corner of 1-minute block (DDMMSS)
302	LONG3	Integer	7	Longitude of southeast corner of 1-minute block (DDDMMSS)
303	LAT3D	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
304	LONG3D	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
311	MN3	Integer	5	Minimum parameter-value
312	MX3	Integer	5	Maximum parameter-value
313	AVG3	Integer	5	Average parameter-value
314	NUM3	Integer	5	Number of values used to compute statistics
315	SDV3	Decimal	7	Standard deviation - 2 decimal places
401-500	AA-JJ	Integer	5	6-second parameter matrix - 10 rows by 10 columns - see figure 6 for arrangement

Table 1.--*Data-base dictionary*--Continued

Schema or component		Format		Remarks
Number	Name	Type	Length	
		LEVEL 1 - MULTI-VALUE PROPOSAL RECORD		
599	PROP	-	-	Level-1 schema record - used to propose changes to parameter values for 6-second data - child of entry record
601	LATP	Integer	6	Latitude of southeast corner of 1-minute block (DDMMSS)
602	LONGP	Integer	7	Longitude of southeast corner of 1-minute block (DDDMMSS)
603	LATPD	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
604	LONGPD	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
605	PERSON	Character	25	Last name of person submitting proposals
606	PRDATE	Date	7	Date proposal loaded into the data base (YYYMMDD)
607	PRTIME	Integer	6	Time proposal loaded into the data base (HHMMSS)
608	PRFLAG	Integer	1	Flag indicating the status of the proposal
701-800	PAA-PJJ	Integer	5	6-second parameter proposal matrix - 10 rows by 10 columns - see figure 6 for arrangement
		LEVEL 1 - SINGLE-VALUE PROPOSAL RECORD		
899	PROP2	-	-	Level-1 schema record - used to propose changes to parameter values for 10-minute and 1-minute data-child of entry record
901	LATP2	Integer	6	Latitude of southeast corner of data block (DDMMSS)
902	LONGP2	Integer	7	Longitude of southeast corner of data block (DDDMMSS)
903	LATPD2	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
904	LNGPD2	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
905	PERSN2	Character	25	Last name of person submitting proposals
906	PRDAT2	Date	7	Date proposal loaded into the data base (YYYMMDD)
907	PRTIM2	Integer	6	Time proposal loaded into the data base (HHMMSS)
908	PRFLG2	Integer	1	Flag indicating the status of the proposal
911	MN9	Integer	5	Minimum parameter-value
912	MX9	Integer	5	Maximum parameter-value
913	AVG9	Integer	5	Proposed parameter-value
914	NUM9	Integer	3	Number of values used to compute statistics
915	SDV9	Decimal	7	Standard deviation - 2 decimal places

AA	AB	AC	AD	AE	AF3	AG	AH	AI	AJ
401	402	403	404	405	406	407	408	409	410
BA	BB	BC	BD	BE3	BF	BG	BH	BI	BJ
411	412	413	414	415	416	417	418	419	420
CA	CB	CC	CD	CE	CF	CG	CH	CI	CJ
421	422	423	424	425	426	427	428	429	430
DA	DB	DC	DD	DE	DF	DG	DH	DI	DJ
431	432	433	434	435	436	437	438	439	440
EA	EB	EC	ED	EE	EF	EG	EH•	EI	EJ
441	442	443	444	445	446	447	448	449	450
FA	FB	FC	FD	FE	FF	FG	FH	FI	FJ
451	452	453	454	455	456	457	458	459	460
GA	GB	GC	GD	GE3	GF	GG	GH	GI	GJ
461	462	463	464	465	466	467	468	469	470
HA	HB	HC	HD	HE	HF	HG	HH	HI	HJ
471	472	473	474	475	476	477	478	479	480
IA	IB	IC	ID	IE	IF	IG	IH	II	IJ
481	482	483	484	485	486	487	488	489	490
JA	JB	JC	JD	JE	JF	JG	JH	JI	JJ
491	492	493	494	495	496	497	498	499	500

A) LEV3 Schema Record

PAA	PAB	PAC	PAD	PAE	PAF	PAG	PAH	PAI	PAJ
701	702	703	704	705	706	707	708	709	710
PBA	PBB	PBC	PBD	PBE	PBF	PBG	PBH	PBI	PBJ
711	712	713	714	715	716	717	718	719	720
PCA	PCB	PCC	PCD	PCE	PCF	PCG	PCH	PCI	PCJ
721	722	723	724	725	726	727	728	729	730
PDA	PDB	PDC	PDD	PDE	PDF	PDG	PDH	PDI	PDJ
731	732	733	734	735	736	737	738	739	740
PEA	PEB	PEC	PED	PEE	PEF	PEG	PEH	PEI	PEJ
741	742	743	744	745	746	747	748	749	750
PFA	PFB	PFC	PFD	PFE	PFF	PFG	PFH	PFI	PFJ
751	752	753	754	755	756	757	758	759	760
PGA	PGB	PGC	PGD	PGE	PGF	PGG	PGH	PGI	PGJ
761	762	763	764	765	766	767	768	769	770
PHA	PHB	PHC	PHD	PHE	PHF	PHG	PHH	PHI	PHJ
771	772	773	774	775	776	777	778	779	780
PIA	PIB	PIC	PID	PIE	PIF	PIG	PIH	PII	PIJ
781	782	783	784	785	786	787	788	789	790
PJA	PJB	PJC	PJD	PJE	PJF	PJG	PJH	PJI	PJJ
791	792	793	794	795	796	797	798	799	800

B) PROP Schema Record

EXPLANATION

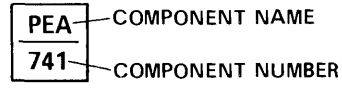

Figure 6.--Arrangement of 6-second block schema items in a 1-minute block.

SUPPLEMENT II: PROGRAM-USER DOCUMENTATION

This supplement contains documentation for the major computer programs used for the second High Plains RASA DMS. The purpose of the documentation is to describe procedures for using the programs; the documentation is not intended to be used as a program-maintenance manual. However, in some instances, information is present that may be useful in the process of software maintenance. To aid the programmer, the programs are internally documented. In addition, Supplement III provides information to help a programmer adapt the software for use by another study.

There are 11 sets of documentation contained within this supplement. The first 8 sets of documentation are for programs that interact with the data base and the DBMS. The final three sets of documentation are for application programs that use data retrieved from the data base.

The Edit Program

The Edit Program checks the correctness of the input records for other programs in the DMS. These other programs are called the destination programs. The principal destination programs are:

1. The Load Program;
2. The Retrieval Program; and
3. The Instant-Update Program.

The main function of the Edit Program is to check the syntax of input records for these destination programs. The program also can perform other types of checks. These additional checks are either automatically or optionally performed by the program. The nature of these additional checks depends upon the types of records that are being edited.

For Load Program input, the Edit Program automatically checks the syntax of each record. For record types 100#, 200#, and 300#, geographic coordinates are checked. The program determines if the latitude and longitude values are within the study area. Latitude and longitude also are checked to determine if their values are proper for the record type. Geographic coordinates are used to determine that the 10-minute by 10-minute blocks are grouped with the correct 1-degree by 1-degree block and that the 1-minute by 1-minute blocks are grouped with the correct 10-minute by 10-minute block. The program determines if the statistical components are only given values on the records corresponding to the base level at which the data are stored. The program can optionally check for possible duplication of data within the data base. This check is useful if the parameter is to be loaded in more than one execution of the Load Program. The program determines which 1-degree by 1-degree blocks, if any, have already been loaded into the data base for the parameter. Then, while editing a 100# record (which corresponds to a 1-degree by 1-degree block), the program determines if that block of data has previously been stored in the data base. If the 1-degree by 1-degree block is already loaded into the data base, the program prints a message and aborts execution.

For Retrieval Program input, syntax checks automatically are performed. The program also determines if the parameter name on the 005# record is valid by matching the name against a list of valid data-base parameters. If the parameter name is valid, the program then determines if the parameter is actually stored within the data base. For the 006# record, the program verifies that the retrieval area, as described by the latitude and longitude extremes, is within the study area.

For Instant-Update Program input, the program automatically checks syntax of the input records. The parameter name, on the 002# record, is checked against a list of valid parameter names. The person's name, on the 599# record, is checked against a list of authorized program users. The remaining checks depend upon the action requested on the 599# record. For example, if a request is made to delete a proposal or to accept a proposal, the program determines if that proposal exists. On the 600# record, geographic coordinates are checked to see that the values are within the study area.

This synopsis does not include all checks performed by the Edit Program. In general, checks performed by the program follow guidelines stated in the user documentation for each of the destination programs; however, no attempt was made to perform every conceivable check that could be made for these input records. The checks that are made are more than sufficient to allow proper execution of the Load, Retrieval, and Instant-Update Programs.

Input

The main input to the program is the data set that is to be edited. Description of the records in the data set can be found in the appropriate destination-program documentation.

The only other input is called the parameter record. Information on this record guides execution of the Edit Program. The parameter record supplies information that is needed to access the data base, including data-base name, data-base password, and the study-unit name. The parameter record also contains the input-unit device number for the records to be edited. If the number of input records is small, the input records can be placed directly within the input stream. In this case, the input records are placed following the parameter record and the input-unit device number is set to 5. Large volumes of input data normally will be stored in a disk file. The input-unit device number of this disk file would then be given on the parameter record. The parameter record also contains the output-unit device number (which will be described in the section on output).

The parameter record also contains an execution-options array. Option 1 is used when editing Load Program input; if set to 0, the program performs only syntax checking; if set to 1, duplication checking also is performed. Option 2 is used when any type of input data is edited. If set to 1, the edited records are printed within the standard print file. Option 3 is not used and needs to be set to 0. Option 4 identifies the destination program for which the records are to be edited. Options 5-10 are not used. The format of the parameter record is:

Columns	Variable	Format	Content
01-30	TITLE	7A4,A2	Title of this run of program
31-32	-	2X	Blank
33-34	INFIL	I2	Input-record logical-unit number
35-36	-	2X	Blank
37-38	OUTFIL	I2	Logical-unit number onto which edited records are to be written
39-40	-	2X	Blank
41-50	OPT	10I1	Options Array:
		Element	Value — Explanation
		1	0 — Check syntax
			1 — Check duplication
		2	0 — Print diagnostics only
			1 — Print all input records
		3	-- — Unused
		4	Destination-program identification:
			0 — Records for Load Program
			1 — Records for Load Program
			3 — Records for Retrieval Program
			4 — Records for Instant-Update Program
		5	-- — Unused
		6	-- — Unused
		7	-- — Unused
		8	-- — Unused
		9	-- — Unused
		10	-- — Unused
51-52	-	2X	Blank
53-60	AQNAME	2A4	Study-unit name (a data-base component identifying the study area, data-base component 2 in table 1, Supplement I)
61-68	DBNAME	2A4	Data-base name (the name by which System 2000 recognizes the data base)
69-72	PASSWD	A4	Data-base password
73-76	-	4X	Blank
77-79	USERID	A3	Users initials; printed on report; needs to be non-blank
80	CODE	A1	P

Output

The printed output contains summary information, including: (1) Number of records edited, (2) number of non-fatal errors detected, and (3) number of fatal errors detected. A fatal error is one that would cause the destination program to execute improperly. For example, if the parameter name is invalid on the 005# record (retrieval-request record), then the Retrieval Program would not be able to retrieve data from the data base. This error would be considered a fatal error. A record containing a fatal error would have to be corrected and the program rerun.

The printed output contains a separate section for error messages. The error messages consist of: (1) An error number, (2) a copy of the record in error, and (3) the error message. The error number is an eight-digit number. If the first digit is 1, the error resulted from an incorrect input record. This type of error is correctable by the program user. In this instance, digits 7 and 8 of the error code is the column where the error was detected. If the first digit is a 2, the error resulted from System 2000 (the DBMS) processing. If one of these errors occurs, the Edit Program is immediately aborted. This type of error only could occur if the data base is damaged. If an error message starting with a 2 is found in the printout and the error resulted in the run being aborted, the computer staff responsible for the program needs to be contacted. If the error message begins with a 9, the error probably resulted from a change in the normal flow of the program because of a previously detected error. For example, if the latitude or longitude on a 100# record (1-degree by 1-degree block load record) is incorrect, it would not be possible to determine if the subsequent 200# records (10-minute by 10-minute block load record) actually represent 10-minute by 10-minute blocks within the 1-degree by 1-degree block. In all instances, digits 2-4 of the error code represent the error number. If the fifth digit is 0, the error was considered minor. If the digit is 1, the error will cause the Edit Program to immediately abort.

The edited records are output to the unit defined on the parameter card. This device usually is a disk file. Output of the edited data is discussed further in the section describing the procedure. Only records without fatal errors are output; if a record contains a fatal error, it is not written to the output device.

Sample Procedure

The Edit program is actually a system of 3 programs. The program DTEDIT1 is used to check input records for the Load Program. The program DTEDIT2 is used to check input records for the Retrieval Program and the program DTEDIT3 is used to check input records for the Instant-Update Program. The name of the program that is to be executed is specified in the IBM Job Control Language (JCL).

The Edit Program may be executed by itself or in conjunction with the Load, Retrieval, or Instant-Update Programs. It is usually executed by itself when the accuracy of the input records is in doubt. In this instance, the output file is not necessary, but it may be useful to list the correct

records. The procedure used to execute the Edit Program is set up so the output can be written to a disk file, but this can be changed when the program is executed. The procedure also allows the input to originate from a disk file, but a small input data set could be part of the input stream. This is done by specifying the input unit as logical-unit 5. A procedure called DTEDIT is used to execute the Edit Program. To execute the Edit Program, with both input and output disk files, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//STEPNAME  EXEC  DTEDIT,NAME1=input,NAME2=output,PROG=programname
//GOED.SYSIN  DD  *
      parameter record
//
```

In this example, *programname* is the name of the version of the Edit program that is to be executed, *input* is the name of the file containing the data to be edited, and *output* is the name of the file to place the edited data.

When executing the Edit Program in conjunction with a destination program, the output file from the Edit Program is defined as a temporary system file that is passed to the destination-program step. The following example shows the Edit Program executed in conjunction with the Retrieval Program. The input file is small and; consequently, it is read instream.

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//STEP1  EXEC  DTEDIT,PROG=DTEDIT2
//GOED.FT10F001  DD  *
    retrieval-request records for Retrieval Program
//GOED.FT11F001  DD  DSN=&&EDOUT,UNIT=SYSDK,DISP=(,PASS),VOL=,
//    DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),SPACE=(TRK,(6,6),RLSE)
//GOED.SYSIN  DD  *
    parameter record for Edit Program
//STEP2  EXEC  DTRETV,NAME1=output
//GORT.FT10F001  DD  DSN=&&EDOUT,DISP=(OLD,DELETE)
//GORT.SYSIN  DD  *
    parameter record for Retrieval Program
//
```

In this example it is assumed that the parameter record for the Retrieval Program specified unit 10 for the retrieval request records. The procedure used to execute the Edit Program is as follows:

```
//DTEDIT PROC TIMEG=2,REG=290K,UNIT1=3350,VOL1=myvol,
//          NAME1=NULLFILE,UNIT2=3350,VOL2=myvol,NAME2=NULLFILE,
//          SP1=1,SP2=10,PROG=DTEDIT1
//*****************************************************************************
//*  DTEDIT: A PROCEDURE TO RUN THE DATA BASE EDIT PROGRAM
//*
//*   USER MAY SUPPLY THE FOLLOWING PARAMETERS(DEFAULTS IN PARENTHESES)
//*
```

```
//*      TIME   TOTAL RUN TIME (2)
//*      REG    REGION SIZE (290K)
//*             CHANGE TO 750K IF DATA BASE IS TO BE ACCESSED
//*      UNIT1  EDIT PROGRAM INPUT RECORDS DEVICE (3350)
//*      VOL1   EDIT PROGRAM INPUT RECORDS VOLUME (myvol)
//*      NAME1  EDIT PROGRAM INPUT RECORDS DSNAME (NULLFILE)
//*      UNIT2  EDIT PROGRAM OUTPUT RECORDS DEVICE (3350)
//*      VOL2   EDIT PROGRAM OUTPUT RECORDS VOLUME (myvol)
//*      NAME2  EDIT PROGRAM OUTPUT RECORDS DSNAME (NULLFILE)
//*      PROG   NAME OF LOAD MODULE TO BE EXECUTED (DTEDIT1)
//*             CHANGE TO DTEDIT2 IF CHECKING RETRIEVAL PROGRAM
//*             INPUT RECORDS
//*             CHANGE TO DTEDIT3 IF CHECKING INSTANT-UPDATE PROGRAM
//*             INPUT RECORDS
//*
//*********************************************************************
//GOED EXEC PGM=&PROG,TIME=&TIMEG,REGION=&REG
//STEPLIB  DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//         DD  DSN=SYS1.FORTG.LINKLIBX,DISP=SHR
//*
//*  DEFINE THE STANDARD I/O DEVICES
//*
//FT05F001 DD  DDNAME=SYSIN
//FT06F001 DD  SYSOUT=A
//FT07F001 DD  SYSOUT=B
//*
//*  LOGICAL UNIT 10(DEFAULT) USED TO INPUT DATA TO EDIT PROGRAM
//*  LOGICAL UNIT 11(DEFAULT) USED FOR OUTPUT OF DATA THAT PASSES EDIT
//*
//FT10F001 DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//FT11F001 DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(NEW,KEEP,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//             SPACE=(TRK,(6,6),RLSE),DSNAME=&NAME2
//*
//*  LOGICAL UNIT 20 USED FOR ERROR CODES AND MESSAGES FILE
//*  LOGICAL UNIT 21 USED FOR OUTPUT OF ERROR MESSAGES TO USER
//*
//FT20F001 DD  DSN=errorfile,DISP=SHR
//FT21F001 DD  SYSOUT=A
//*
//*  DEFINE FILES CONTAINING PARAMETER AND USERS' NAMES
//*
//FT22F001  DD  DSN=userfile,DISP=SHR
//FT23F001  DD  DSN=parmsfile,DISP=SHR
//*
//*  DEFINE FILES USED BY SYSTEM 2000
//*
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD DUMMY
//S2KCOMD  DD DUMMY
//S2KUDUMP DD DUMMY
```

```
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE01 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE02 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=SHR
//dbname2 DD DSN=databasefile2,DISP=SHR
//dbname3 DD DSN=databasefile3,DISP=SHR
//dbname4 DD DSN=databasefile4,DISP=SHR
//dbname5 DD DSN=databasefile5,DISP=SHR
//dbname6 DD DSN=databasefile6,DISP=SHR
```

Note: In this procedure and subsequent procedures shown in this report, *dbname* is the first seven characters of the data-base name.

The Load Program

The Load Program is used to load large volumes of data into the data base. The program uses the System 2000 (the DBMS) optimized-loading procedure. One parameter normally is loaded per program execution, but 6-second data may be loaded in several executions, because of the large quantity of data involved. If a parameter is loaded into the data base in more than one execution of the program, whole 1-degree by 1-degree blocks of data must be loaded in a single operation. This restriction is placed in the program to make it easy to account for what data has been loaded. This restriction also simplifies the editing procedure used to check the data prior to loading.

The System 2000 optimized-load procedure, which is designed to load large aggregates of data, stores the data into contiguous locations within the data base. This ensures that the number of accesses to the data base are minimized when the data are retrieved. If a parameter is loaded in several executions of the program, the executions need to be done consecutively in order to have all data for the parameter in one area of the data base.

The data needs to be thoroughly edited prior to loading by using the Edit Program. If the data successfully passes the Edit Program checks, there should be no difficulty in properly loading the data. The Load Program performs some minor data checks, but these are not sufficient to ensure the correctness of the data. The Edit Program needs to be used, because certain types of errors in the input data can cause System 2000 processing to fail

at a time when the data base is being updated; this would likely damage the data base. For this reason, the first step in the loading process is to save the data base on magnetic tape; then, if necessary, the data base can be restored to its former condition prior to an aborted attempt to load data.

Input

The first input is the parameter record. The format of this record is found in the documentation for the Edit Program. The only fields on this record that are directly used by the Load Program are the data-base password and the code found in column 80. The remaining fields are for use by the Edit Program. The password must be the master password for the data base.

The data to be loaded are input in standard record formats; specifications of these records are found following this discussion. The Load Program can process all densities of data. The input begins with the master records 001#, 002#, and 003#. These records contain general information pertaining to the parameter and contribute the data that comprises the root-level record. Optionally, the next records in the input data set are remarks data. These records are used for a short narrative description of the parameter and may include information on the origin of the data. The sequence of records following the master and remarks data depend upon the data density.

A parameter stored in the data base as 1-degree data has one value for each 1-degree by 1-degree block that is partially or completely within the study area. A 1-degree by 1-degree block is identified by the record type 100#. The 100# record contains the latitude and longitude of the southeast corner of the 1-degree by 1-degree block, as well as the value. The record also contains statistical components that may be optionally valued. These statistics are the minimum, the maximum, the number of values, and the standard deviation. These statistics are a reflection of the data used to compute the single value for the 1-degree by 1-degree block. The number of 100# records is equivalent to the number of 1-degree by 1-degree blocks partially or completely within the study area.

A parameter stored in the data base as 10-minute data has one value for each 10-minute by 10-minute block within the study area. A 10-minute by 10-minute block is identified by the record type 200#. The 200# record contains the latitude and longitude of the southeast corner of the 10-minute by 10-minute block as well as the value. The record may also optionally contain the minimum, the maximum, the number of values, and the standard deviation. These statistics are a reflection of the data used to compute the single value for the 10-minute by 10-minute block. In the input data set, 10-minute by 10-minute blocks are grouped within 1-degree by 1-degree blocks. The 1-degree by 1-degree block is represented by a 100# record; however, in this case, all five of the statistical components must be blank. Values for these statistical components are computed after the data have been loaded into the data base.

A parameter stored in the data base as 1-minute data has one value for each 1-minute by 1-minute block within the study area. The 1-minute by 1-minute block is identified by the record type 300#. This record contains the

latitude and longitude of the southeast corner of the block as well as the same five statistical components found on the 100# and 200# records. Because the 1-minute by 1-minute block represents the base level for the parameter, all statistical components on the 300# record can be valued, but only the value component is required. The 1-minute by 1-minute blocks are grouped within 10-minute by 10-minute blocks; which in turn, are grouped within 1-degree by 1-degree blocks. All statistical components must be blank on the 100# and 200# records. These statistics are computed after the data are loaded.

A parameter stored in the data base as 6-second data has one value for each 6-second by 6-second block within the study area. The format of this data is slightly different from the preceding data. A 1-minute by 1-minute block is represented by a 300# record that contains the latitude and longitude of the southeast corner of the block and the five statistical components. The 300# record is followed by ten records, types 401# through 410#, that contain values for the 100 6-second by 6-second blocks within the 1-minute by 1-minute block. These ten records may be considered a 10-by-10 matrix covering the 1-minute by 1-minute block. The 401# record contains ten values for the northern-most row of 6-second data. Within a row, values are placed from west to east. If a 1-minute by 1-minute block is located along the boundary of the study area, some of the 6-second cells may be outside the area. These cells are not given a valid value, but instead are given a value called the missing-value indicator (MVI). This value, which is found on the 003# record, indicates that a particular point has no valid value. Only 6-second by 6-second blocks that have valid values are used in computing statistics. As with 1-minute data, the 1-minute by 1-minute blocks are grouped under the corresponding 10-minute by 10-minute blocks; which in turn, are grouped with the proper 1-degree by 1-degree blocks. Statistical components on the 100#, 200#, and 300# records are all blank.

001#: First master record for the Load Program

Columns	Component or variable	Key or non-key	Format	Content
01-04	TYPE	-	A4	001#
05-10	-	-	6X	Blank
11-40	C4	NK	7A4,A2	Project Name
41-49	C21	NK	I9	Reserved Value A
50-60	C22	NK	F11.3	Reserved Value B
61-70	C23	NK	2A4,A2	Reserved Value C
71-72	-	-	2X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

002#: Second master record for the Load Program

01-04	TYPE	-	A4	002#
05-10	-	-	6X	Blank
11-18	C2	K	2A4	Study-unit name
19-48	C3	K	7A4,A2	Parameter Name
49	C5	NK	I1	Density (1=1-degree, 2= 10-minute, 3=1-minute, 4=6-second)
50-52	C6	NK	I3	Scale factor (value x 10^{scale})
53-62	C17	NK	2A4,A2	Units of measurement - description
63-72	-	-	10X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

003#: Third master record for the Load Program

01-04	TYPE	-	A4	003#
05-10	-	-	6X	Blank
11-25	-	-	15X	Blank
26-30	C11	NK	I5	Minimum
31-35	C12	NK	I5	Maximum
36-40	C13	NK	I5	Average
41-47	C14	NK	I7	Number of values
48-55	C15	NK	F8.2	Standard Deviation
56-62	-	-	7X	Blank
63-67	C30	NK	I5	Missing-Value Indicator (MVI)
68-72	-	-	5X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

050#: Remarks record

01-04	TYPE	-	A4	050#
05-10	-	-	6X	Blank
11-16	C51	NK	A4,A2	Remark date to store in data base (MMDDYY)
17-18	-	-	2X	Blank
19-21	C52	NK	I3	Remark sequence number
22	ITEM	-	I1	Remark record number 1=First record of set 2=Second record of set 3=Third record of set 4=Fourth record of set 5=Fifth record of set
23-72	TEXT	NK	12A4,A2	Text of remark
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

Note: The 050# record must come in sets of five.

100#: One-degree data record

01-04	TYPE	-	A4	100#
05-10	-	-	6X	Blank
11-16	C101	K	I6	Latitude (DDMMSS)
17	-	-	1X	Blank
18-24	C102	K	I7	Longitude (DDDMMSS)
25	-	-	1X	Blank
26-30	C111	NK	I5	Minimum
31-35	C112	NK	I5	Maximum
36-40	C113	NK	I5	Average (Value)
41-45	C114	NK	I5	Number of values
46-53	C115	NK	F8.2	Standard Deviation
54-72	-	-	19X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

200#: Ten-minute data record

01-04	TYPE	-	A4	200#
05-10	-	-	6X	Blank
11-16	C201	K	I6	Latitude (DDMMSS)
17	-	-	1X	Blank
18-24	C202	K	I7	Longitude (DDDMMSS)
25	-	-	1X	Blank
26-30	C211	NK	I5	Minimum
31-35	C212	NK	I5	Maximum
36-40	C213	NK	I5	Average (Value)
41-45	C214	NK	I5	Number of values
46-53	C215	NK	F8.2	Standard Deviation
54-72	-	-	19X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

300#: One-minute block record--1-minute data

01-04	TYPE	-	A4	300#
05-10	-	-	6X	Blank
11-16	C301	NK	I6	Latitude (DDMMSS)
17	-	-	1X	Blank
18-24	C302	NK	I7	Longitude (DDDMMSS)
25	-	-	1X	Blank
26-30	C311	NK	I5	Minimum
31-35	C312	NK	I5	Maximum
36-40	C313	NK	I5	Value
41-45	C314	NK	I5	Number of values
46-53	C315	NK	F8.2	Standard Deviation
54-72	-	-	19X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

300#: One-minute block record--6-second data

01-04	TYPE	-	A4	300#
05-10	-	-	6X	Blank
11-16	C301	NK	I6	Latitude (DDMMSS)
17	-	-	1X	Blank
18-24	C302	NK	I7	Longitude (DDDMMSS)
25-72	-	-	48X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

Note: This record must be followed by exactly ten 4xx# records.

401#-410#: Six-second data records

01-04	TYPE	-	A4	4xx#; xx=(01,02,...,10)
05-10	-	-	6X	Blank
11-60	C401-C500	NK	10I5	See note below
61-72	-	-	12X	Blank
73-80	INSEQ	-	I8	Sequence Number, nondecreasing

Note: The components, C401-C500, are entered ten components per record such that the records simulate a 10-by-10 matrix geographically overlaying the 1-minute by 1-minute block:

Record	Components
401#	C401-C410
402#	C411-C420
403#	C421-C430
404#	C431-C440
405#	C441-C450
406#	C451-C460
407#	C461-C470
408#	C471-C480
409#	C481-C490
410#	C491-C500

Output

Output from the Load Program consists of a short, printed report providing basic information pertaining to the loaded data. The report includes a list of the 1-degree by 1-degree blocks that were loaded into the data base and a graphic picture that indicates the location of all of the 1-degree by 1-degree blocks within the High Plains RASA study area with the 1-degree by 1-degree blocks that were loaded highlighted.

Sample Procedure

To execute the Load Program, a procedure called DTLOAD is used. A data set called DATAIN contains the input data. This data set consists of the parameter record, followed by the proper combination of the 001#, 002#, 003#, and so forth, records. The Load Program should be executed in conjunction with the Edit Program. Hence, the first step would be to execute the Edit Program, which produces a file containing input to the Load Program. The parameter record and the other records would be input to the Edit Program in the manner described in that documentation. The file containing the edited data would be passed to the Load Program step, which would load the data into the data base. To execute the Load Program, the following JCL should be used:

```
// . . . JOB . . .
//*SETUP tapenum/HR
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//STEPNAME  EXEC  DTLOAD
//GO.DATAIN  DD  *
     parameter and data records
//
```

The second line describes the magnetic tape used to save the data base prior to loading any new data. The procedure used to execute the Load Program is as follows:

```
//DTLOAD PROC REG=800K,TIMEG=2,SP1=1,SP2=10
//GO     EXEC PGM=DTLOAD,REGION=&REG,TIME=&TIMEG,
//       PARM='ISA(16K)'
//STEPLIB  DD DSN=mylib,DISP=SHR
//         DD DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD DSN=SYS1.S2K,DISP=SHR
//SYSLIB   DD DSN=SYS1.S2K,DISP=SHR
//         DD DSN=SYS1.PLIBASE,DISP=SHR
//SYSPRINT DD SYSOUT=A
//PLIDUMP  DD  DUMMY
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD SYSOUT=A,DCB=BLKSIZE=882
//S2KCOMD  DD DUMMY
//S2KUDUMP DD SYSOUT=A
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE01 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE02 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//***  S2K WORK FILES:
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
```

```
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//*** DATA BASE FILES:
//dbname1 DD DSN=databasefile1,DISP=OLD
//dbname2 DD DSN=databasefile2,DISP=OLD
//dbname3 DD DSN=databasefile3,DISP=OLD
//dbname4 DD DSN=databasefile4,DISP=OLD
//dbname5 DD DSN=databasefile5,DISP=OLD
//dbname6 DD DSN=databasefile6,DISP=OLD
//TAPES2K  DD  DSN=DBBACKUP,UNIT=(TAPE62,,DEFER),VOL=SER=tapenum,
//    DISP=(OLD,KEEP)
//KEEPFILE DD DUMMY
```

The Retrieval Program

The Retrieval Program retrieves data from the data base. There are four specifications required to identify the data to be retrieved: (1) The study-unit name, (2) the parameter name, (3) the description (in geographic coordinates) of a rectangular area in which data are to be retrieved, and (4) the lowest level within the data-base structure where data are to be extracted. Using these four specifications, the Retrieval Program produces a set of data in the same format that is used to initially load the data in the data base.

By using the level specification, the user has the capability to retrieve statistical records for a parameter. For example, 10-minute data has its primary data stored at level 2 of the data-base structure. In order to retrieve the statistics that are stored in the 1-degree statistical records, the retrieval level should be set to 1. Retrieved data would then consist of 100# records, where the 100# records (1-degree statistical records) contain the statistics computed from values for the 10-minute by 10-minute blocks that are within the 1-degree by 1-degree block.

The user can optionally request the inclusion of test proposals in the retrieved data. This is done through use of the variable, called K599, that can be found on the 006# retrieval request record. If K599 is set to 1, the output will include test proposals.

When the user requests the retrieval of test proposals, the proposed value will replace the current accepted-value in the output. If 6-second data are being retrieved, then for each 1-minute by 1-minute block (which represents 100 6-second by 6-second blocks), the current accepted 6-second values are replaced only if the 6-second by 6-second block has a corresponding test proposal. If more than one test proposal is located for a block, the latest test proposal is output. For 6-second data, the test proposals are merged into a single proposal, beginning with the oldest test proposal and proceeding to the newest test proposal.

Input

The input to the program is as follows:

Record (1) - Parameter Record - Read from unit 5

Col 1-30	TITLE	7A4,A2	Title of run
Col 31-32	-	2X	Blank
Col 33-34	INFIL	I2	Input-record device
Col 35-36	-	2X	Blank
Col 37-38	OUTFIL	I2	Retrieved-data output device
Col 39-40	-	2X	Blank
Col 41-50	OPT	10I1	Option Array

Element	Value	Content
1	0	Print diagnostics only
	1	Print retrieved records
2-10	0	Unused

Col 51-52	-	2X	Blank
Col 53-60	AQNAME	2A4	Study-unit name
Col 61-68	DBNAME	2A4	Data-base name
COL 69-72	PASSWD	A4	Data-base password
Col 73-76	-	4X	Blank
Col 77-79	USERID	A3	User initials
Col 80	KODE	A1	P

Record (2) - 005# - Read from user-selected unit

Col 1-4	TYPE	A4	005#
Col 5-10	-	6X	Blank
Col 11-18	AQNAME	2A4	Study-unit name
Col 19-48	PNAME	7A4,A2	Parameter name (left-justified)
Col 49	-	1X	Blank
Col 50	LEVEL	I1	Level of retrieval
Col 51-72	-	22X	Blank
Col 73-80	INSEQ	I8	Nondecreasing sequence number

Record (3) - 006# - Read from same unit as record (2)

Col 1-4	TYPE	A4	006#
Col 5-10	-	6X	Blank
Col 11-16	INLT1	I6	Minimum latitude, DDMMSS (whole degrees)
Col 17	-	1X	Blank
Col 18-24	INLN1	I7	Minimum longitude, DDDMMSS (whole degrees)
Col 25	-	1X	Blank
Col 26-31	INLT2	I6	Maximum latitude, DDMMSS (whole degrees)
Col 32	-	1X	Blank
Col 33-39	INLN2	I7	Maximum longitude, DDDMMSS (whole degrees)
Col 40-42	-	3X	Blank

Col 43	K599	I1	Temporary inclusions flag
Col 44-72	-	29X	Blank
Col 73-80	INSEQ	I8	Nondecreasing sequence number

Record (4) - 999# - Read from same unit as record (2)

Col 1-4	TYPE	A4	999#
Col 5-72	-	68X	Blank
Col 73-80	INSEQ	I8	Nondecreasing sequence number

There are several restrictions on the order of these records. Each 005# record must be followed by at least one 006# record. If this does not occur, the program will be aborted. A 006# record cannot exist without a corresponding 005# record. Retrieval requests can contain more than one 005# record, providing each 005# record is followed by one or more 006# records. This implies that any retrieval run can retrieve data for more than one parameter. (Note: If the Retrieval Program obtains input to the Data-Transformation Program, only retrieve one parameter at a time.) One 999# record is required; this should be the last record.

Output

The output consists of:

(1) Messages - written to unit 6 (standard print file);
(2) Retrieved data - written to unit defined on parameter record (usually a disk file); and
(3) Error messages - written to unit 21 (auxiliary print file).

In many cases, when an error occurs in the execution of this program, the program will print an error message following the standard messages. The error message will consist of an error code, a copy of the record being processed, and the error message. The error code consists of 8 digits. The first digit identifies the type of error. If set to 1, the error resulted from incorrect input. This type of error can be corrected by the program user. If set to 2, the error resulted from an attempt to execute a System 2000 (the DBMS) command. For this type of error, the user should contact the computer staff responsible for the program. If the first digit is 9, the error resulted from a logic fault within the program. When this occurs, the computer staff should be contacted. The second through fourth digits of the code identify the specific error. If the fifth digit is set to 0, the error was not severe enough to cause the run to be aborted. If set to 1, the run is aborted. If the first digit of the error code is 1, the seventh and eighth digits identify the column where the input error is located. If the first digit is 2, the sixth through eighth digits indicate the System 2000 completion code resulting from the execution of a System 2000 command. This completion code needs to be given to the computer staff when they are contacted about the error.

Sample Procedure

A procedure called DTRETV is used to execute the Retrieval Program. To execute the Retrieval Program, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB DD DSN=procedure.library,DISP=SHR
//  EXEC  DTRETV,NAME1=output
//GORT.SYSIN  DD  *
  parameter record
  retrieval request records
//
```

The procedure used to execute the Retrieval Program is as follows:

```
//DTRETV PROC TIMEG=2,REG=740K,VOL1=myvol,NAME1=NULLFILE,
//       SP1=1,SP2=10,PROG=DTRETV,UNIT1=3350
//*******************************************************************************
//*  DTRETV: A PROCEDURE TO RUN THE DATA BASE RETRIEVAL PROGRAM
//*
//*   USER MAY SUPPLY THE FOLLOWING PARAMETERS(DEFAULTS IN PARENTHESES)
//*
//*      TIMEG  TOTAL RUN TIME (2)
//*      REG    REGION SIZE (740K)
//*      UNIT1  RETRIEVAL PROGRAM OUTPUT DEVICE (3350)
//*      VOL1   RETRIEVAL PROGRAM OUTPUT VOLUME (myvol)
//*      NAME1  RETRIEVAL PROGRAM OUTPUT DSNAME (NULLFILE)
//*      PROG   NAME OF LOAD MODULE TO BE EXECUTED (DTRETV)
//*
//*******************************************************************************
//GORT EXEC PGM=&PROG,TIME=&TIMEG,REGION=&REG
//STEPLIB  DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//         DD  DSN=SYS1.FORTG.LINKLIBX,DISP=SHR
//*
//*  DEFINE THE STANDARD I/O DEVICES
//*
//FT05F001 DD  DDNAME=SYSIN
//FT06F001 DD  SYSOUT=A
//FT07F001 DD  SYSOUT=B
//*
//*  LOGICAL UNIT 11(DEFAULT) USED FOR OUTPUT OF RETRIEVED DATA
//*
//FT11F001 DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=(NEW,KEEP,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//             SPACE=(TRK,(6,6),RLSE),DSNAME=&NAME1
//*
//*  LOGICAL UNIT 20 USED FOR ERROR CODES AND MESSAGES FILE
//*  LOGICAL UNIT 21 USED FOR OUTPUT OF ERROR MESSAGES TO USER
//*
//FT20F001 DD  DSN=errorfile,DISP=SHR
//FT21F001 DD  SYSOUT=A
//*
```

```
//*  DEFINE FILES USED BY SYSTEM 2000
//*
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD DUMMY
//S2KCOMD  DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE01 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE02 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=SHR
//dbname2 DD DSN=databasefile2,DISP=SHR
//dbname3 DD DSN=databasefile3,DISP=SHR
//dbname4 DD DSN=databasefile4,DISP=SHR
//dbname5 DD DSN=databasefile5,DISP=SHR
//dbname6 DD DSN=databasefile6,DISP=SHR
```

The General-Update Program

The General-Update Program is used to make random changes to permanent data within the data base. The program uses the System 2000 (the DBMS) MODIFY command to replace the existing values with new values. Because a modify is used to update the data, the physical order of the data is not altered. This program is intended to be used primarily by the DBA. To perform the updates, the master password is required.

The program is designed to update 1-degree, 10-minute, 1-minute, and 6-second data. For 1-degree, 10-minute and 1-minute data, the program replaces all five statistical components (minimum, maximum, value, number of values, and standard deviation) for each of the specified 1-degree, 10-minute or 1-minute records. For 6-second data, the user will input new values only for the 6-second by 6-second blocks that are to receive new values. The program will update one parameter per program execution. For all data densities, the statistics are recomputed after all updates are completed. With this program a value that is equivalent to the missing-value indicator (MVI) can be changed or an existing value can be changed to the MVI.

Input

All input, except for the master password, is read from data set DATAIN.

Record 1: This is the 002# record. It describes the parameter that is to be updated.

Col	1-4	Record Type	A4	002#
Col	5-10	-	6X	Blank
Col	11-18	Unit	2A4	Study-unit name
Col	19-48	Parameter	7A4,A2	Parameter name
Col	49-72	-	24X	Blank
Col	73-80	Sequence	I8	Nondecreasing sequence number

Record 2: This record is the 007# record if updating 1-degree, 10-minute, or 1-minute data. The record contains the latitude and longitude of the 1-degree, 10-minute or 1-minute block and new values for the five statistics. The input will contain one 007# record for each block that is to be updated. The 007# record is formatted as follows:

Col	1-4	Record Type	A4	007#
Col	5-10	-	6X	Blank
Col	11-16	Lat	I6	Latitude (DDMMSS)
Col	17	-	1X	Blank
Col	18-24	Lon	I7	Longitude (DDDMMSS)
Col	25	-	1X	Blank
Col	26-30	Min	I5	Minimum
Col	31-35	Max	I5	Maximum
Col	36-40	Value	I5	New value
Col	41-45	Numpts	I5	Number of values
Col	46-53	Stndev	F8.2	Standard deviation
Col	54-72	-	19X	Blank
Col	73-80	Sequence	I8	Nondecreasing sequence number

Record 2 is the 008# record if updating 6-second data. This record contains the latitude and longitude of the 1-minute block and is formatted as follows:

Col	1-4	Record Type	A4	008#
Col	5-10	-	6X	Blank
Col	11-16	Lat	I6	Latitude (DDMMSS)
Col	17	-	1X	Blank
Col	18-24	Lon	I7	Longitude (DDDMMSS)
Col	25-72	-	48X	Blank
Col	73-80	Sequence	I8	Nondecreasing sequence number

Record 3: This record(s) is used when updating 6-second data. Each 008# record is followed by a series of records that contain new values for some or all of the 100 6-second by 6-second blocks contained within a 1-minute by 1-minute block. The records types are 801# through 810#. The 1-minute by

1-minute block can be thought of as a 10x10 matrix of 6-second values. The 801# record represents the first (north) row of the 10x10 matrix; the 804# record corresponds to the fourth row of the matrix; and so forth. Not all ten rows are required to have new values, nor or all 10 points within a row required. Hence, if the only rows that are to have new values are 5, 6, 7, and 8, then the 008# record is followed by 805#, 806#, 807#, and 808# records. Within these rows, only place values in the colums where a new value is to be inserted into the data base. The formats of records 801# through 810# are identical:

Col	1-4	Record Type	A4	801# through 810#
Col	5-10	-	6X	Blank
Col	11-60	Values	10I5	New values
Col	61-72	-	12X	Blank
Col	73-80	Sequence	I8	Nondecreasing sequence number

The series 008#, 801#-810# are repeated as many times as necessary.

Last record: The last record in the input must be a 999#

Col	1-4	Record Type	A4	999#
Col	5-72	-	68X	Blank
Col	73-80	Sequence	I8	Nondecreasing sequence number

The master password is also read into the program using a separate input unit.

Output

The output consists of two print files. The first file is the standard print file. It contains information such as: parameter being updated; number of updates; listing of update-request records; and error messages.

The second file is called the activity report. It contains specific information on each data block that is updated. For parameters stored as 1-degree, 10-minute or 1-minute data, the reports contains: latitude and longitude; the new statistical-values; the previous permanent-value; and a note if the old value was equivalent to the MVI or if the new value is equivalent to the MVI.

For parameters stored as 6-second data, the information supplied is more extensive. For each 1-minute by 1-minute block, the report contains the latitude and longitude of the 1-minute by 1-minute block and a 10x10 matrix of the 100 6-second values within the block. The new values are highlighted with asteriks. The remaining values represent the current permanent-values for those 6-second by 6-second blocks.

Sample Procedure

A procedure called DTGENUPD is used to execute the General-Update Program. To execute the General-Update Program, the following JCL should be used:

```
//... JOB ...
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTGENUPD
//GU.PASS  DD  *
  master password
//GU.DATAIN  DD  *
  update request records
```

The procedure used to execute the General-Update Program is as follows:

```
//DTGENUPD  PROC  TIM=2,REG=800K,PROG=DTGENUPD,SP1=1,SP2=10
//*********************************************************************
//*   DTGENUPD: A PROCEDURE TO RUN THE DATA BASE GENERAL-
//*             UPDATE PROGRAM
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*   TIM    TOTAL RUN TIME (2)
//*   REG    REGION SIZE (800K)
//*   PROG   LOAD MODULE NAME (DTGENUPD)
//*
//*********************************************************************
//*
//GU  EXEC  PGM=&PROG,TIME=&TIM,REGION=&REG,PARM='ISA(12K)'
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//*
//*   DEFINE STANDARD I/O DATA SETS
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//*
//*  DEFINE REPORT FILE
//*
//REPORT  DD  SYSOUT=A
//*
//*   DEFINE SYSTEM 2000 FILES
//*
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD DUMMY
//S2KCOMD  DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
```

```
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=SHR
//dbname2 DD DSN=databasefile2,DISP=SHR
//dbname3 DD DSN=databasefile3,DISP=SHR
//dbname4 DD DSN=databasefile4,DISP=SHR
//dbname5 DD DSN=databasefile5,DISP=SHR
//dbname6 DD DSN=databasefile6,DISP=SHR
```

The Instant-Update Program

The Instant-Update Program allows users to propose changes to data within the data base; these proposed changes are called test proposals. When a request is made to add a test proposal for a particular latitude-longitude block and parameter, the test proposal is loaded in the data base as part of the data belonging to the parameter, but the test proposal is kept separate from the other data. This separate area is temporary in nature; the accepted data reside in the permanent area of the data base. The Retrieval Program may optionally retrieve the test proposals.

An example of the use of test proposals would be in modeling. After test proposals are added to the data base, the data would be retrieved with the request that test proposals be included within the output. Then, for any permanent record within the retrieved area that has a corresponding test proposal, the proposed value would replace the current accepted-value in the retrieval output. After the retrieval, the data would be reformatted into model input data using the Data-Transformation Program. During the modeling process, the user may decide that some of the proposed values are better than the permanent values. The user may then want the proposed values to replace the corresponding permanent values. This is also done using the Instant-Update Program. When a test proposal is marked as an accepted proposal, the proposed value effectively becomes part of the permanent data for that parameter. Conversely, if the user determines that some of the test proposals are not satisfactory, the test proposals may be marked as rejected. This is the third function of the Instant-Update Program. Once rejected, the proposal effectively is no longer a part of the temporary data for that parameter.

Each test proposal is loaded into the data base as a separate record within the temporary area of that parameter. The test proposal is identified by its latitude, longitude, date and time added, and a flag. The flag indicates that the record is a test proposal, an accepted proposal, or a

rejected proposal. Only one test proposal should be added per latitude-longitude block per program execution. If more than one test proposal is added for a particular block in the same execution, the date and time of the proposal would be the same, and a later program execution would not be able to distinguish among the test proposals. After test proposals have been added, these proposals can be marked as rejected or accepted in subsequent executions of the program. A typical program execution may consist of a mixture of the three functions: adding test proposals, rejecting test proposals, and marking test proposals as accepted.

The Instant-Update Program updates one parameter per run. The program can be used to update 10-minute, 1-minute, and 6-second data. If more than one of the three program functions are to be performed by a particular run, the order must be:

1. Reject proposals
2. Accept proposals
3. Add test proposals

Only test proposals can be marked as rejected or accepted. A rejected proposal cannot be changed to test or accepted, because logically it no longer exists. An accepted proposal cannot be changed, because logically its proposed value is part of the permanent data.

The program limits the number of test proposals per latitude-longitude block to five. If five test proposals already exist, the user will not be allowed to add another. A message to this effect is printed and a listing of the five test proposals is provided. To add another test proposal, at least one of the existing test proposals should be rejected. If the value stored in the permanent data-area for the latitude-longitude block is the MVI, a proposal is not allowed. Also, the proposed value cannot be the MVI. Only the DBA can change the MVI to a valid value or a valid value to the MVI. A proposal can only be added for a record that already exists in the permanent area of the data base.

Input

Record 1: This is the 002# record. It describes the parameter that is to be updated.

Col 1-4	Record Type	A4	002#
Col 5-10	-	6X	Blank
Col 11-18	Unit	2A4	Study-unit name
Col 19-48	Parameter	7A4,A2	Parameter name
Col 49-72	-	24X	Blank
Col 73-80	Sequence	I8	Nondecreasing sequence number

Record 2: This is the 599# record. It contains the name of the person submitting the proposal changes. The person must be an authorized user of the program. The record also contains the action code: 1 indicates that a test proposal is to be marked rejected; 2 indicates that a test proposal is to be

marked accepted; and 3 indicates that a test proposal is to be added. If the action code is 1 or 2, the 599# record also contains the date and time when the proposal was added to the data base. This date and time can be found as part of the output of a previous Instant-Update Program execution. If the action code is 3, the date and time fields should be blank.

Col 1-4	Record Type	A4	599#
Col 5-10	-	6X	Blank
Col 11-35	Person	6A4,A1	Person submitting changes (last name)
Col 36	-	1X	Blank
Col 37	Action	I1	Action code
Col 38	-	1X	Blank
Col 39-44	Date	A4,A2	Date proposal added (MMDDYY)
Col 45-47	-	3X	Blank
Col 48-53	Time	I6	Time proposal added (HHMMSS)
Col 54-72	-	19X	Blank
Col 73-80	Sequence	I8	Nondecreasing sequence number

Record 3: This is the 600# record. This record can take several forms depending on the preceding 599# record. If the parameter being updated is 10-minute, 1-minute, or 6-second data and the preceding 599# record contained an action code of 1 or 2, the 600# record contains only the latitude and longitude of the data block where a test proposal exists. The record will appear as follows:

Version 1: 10-minute, 1-minute, or 6-second data - Action code 1 or 2

Col 1-4	Record Type	A4	600#
Col 5-10	-	6X	Blank
Col 11-16	Lat	I6	Latitude (DDMMSS)
Col 17	-	1X	Blank
Col 18-24	Lon	I7	Longitude (DDDMMSS)
Col 25-72	-	48X	Blank
Col 73-80	Sequence	I8	Nondecreasing sequence number

If the action code is 3 and the data are stored as 10-minute or 1-minute data, then the 600# record contains not only the latitude and longitude of the block but also the proposed value. In addition, the user may supply statistics that were used to compute the proposed value. These statistics are not required, but the proposed value must be present. The statistics are: minimum, maximum, number of values, and standard deviation.

Version 2: 10-minute or 1-minute data - Action code 3

Col 1-4	Record Type	A4	600#
Col 5-10	-	6X	Blank
Col 11-16	Lat	I6	Latitude (DDMMSS)
Col 17	-	1X	Blank
Col 18-24	Lon	I7	Longitude (DDDMMSS)

Col 25	-	1X	Blank
Col 26-30	Min	I5	Minimum
Col 31-35	Max	I5	Maximum
Col 36-40	Value	I5	Proposed value
Col 41-42	-	2X	Blank
Col 43-45	Numpts	I3	Number of values
Col 46-53	Stndev	F8.2	Standard deviation
Col 54-72	-	19X	Blank
Col 73-80	Sequence	I8	Nondecreasing sequence number

If the action code is 3 and the parameter is 6-second data, the 600# record contains only the latitude and longitude of the 1-minute block. It is identical in format to version 1 of the 600# record. The 600# record is then followed by a group of records that contain proposed values for some or all of the 100 6-second by 6-second blocks contained within the 1-minute by 1-minute block whose latitude and longitude is defined on the 600# record. These records are type 701# through 710#. The 1-minute by 1-minute block can be thought of as a 10x10 matrix of 6-second values. The 701# record represents the first (north) row of the 10x10 matrix; the 704# record corresponds to the fourth row of the matrix; and so forth. Not all ten rows are required to have proposed values, nor are all 10 points within a row required. Hence, if the only rows that are to have values proposed are 5, 6, 7, and 8, then the 600# record should be followed by 705#, 706#, 707#, and 708# records. Within these rows, only place values in the columns where a proposed value is to be inserted. If the points 2, 3, 4, and 5 within a row are to have proposed values, positions 1, 6, 7, 8, 9, and 10 should be blank. The formats of records 701# through 710# are identical:

Record 4: 6-second data - Action code 3

Col 1-4	Record Type	A4	701# through 710#
Col 5-10	-	6X	Blank
Col 11-60	Values	10I5	Proposed values
Col 61-72	-	12X	Blank
Col 73-80	Sequence	I8	Nondecreasing sequence number

The series 600#, 701#-710# are repeated as many times as necessary.

Last Record: The last record in the input must be a 999# record

Col 1-4	Record Type	A4	999#
Col 4-72	-	68X	Blank
Col 73-80	Sequence	I8	Nondecreasing sequence number

The following examples are given to make description of input more easily understood:

Example 1: Adding test proposals for parameters stored as 10-minute or 1-minute data. •

```
002#      NEBRASKAPUMPAGE - 1978                                    1
599#      LUCKEY             3                                      2
600#      400000  1012000    720                                    3
```

```
600#       400000  1012100    730                                          4
600#       400000  1012200    740                                          5
999#                                                                       6
```

Each of the 600# records represents a 1-minute by 1-minute block within the 1-minute parameter PUMPAGE - 1978. If the parameter was stored as 10-minute data, each 600# record would represent a 10-minute by 10-minute block.

Example 2: Adding test proposals for a parameter stored as 6-second data.

```
002#       NEBRASKAWATER TABLE - 1960                                      1
599#       LUCKEY                3                                         2
600#       405000  1013000                                                 3
701#        2520  2530  2549  2550                                         4
702#        2510  2515  2520  2560                                         5
703#        2570  2580  2590                                               6
600#       405000  1013100                                                 7
703#                    2770  2761  2752  2743                             8
704#                    2766  2757  2748  2739  2730                       9
705#              2770  2761  2753  2745  2736                            10
600#       400000  1013200                                                11
707#        2505  2495  2415  2477  2468                                  12
708#        2510  2500  2490  2480                                        13
709#        2517  2506  2495                                              14
710#        2520  2509                                                    15
999#                                                                      16
```

Each 600# record represents a 1-minute by 1-minute block; each value on the 701#-710# records is a proposed value for a 6-second by 6-second block within the 1-minute by 1-minute block. The position of the 6-second by 6-second block is determined by the record type and the position in the record. For example, a value in row 6 is placed in record type 706#, and the value found in columns 11-15 is the first value in that row.

Example 3: Mixture of actions.

```
002#       NEBRASKAPUMPAGE - 1978                                          1
599#       LUCKEY            1  083181  124536                           2
600#       400000  1012000                                                3
599#       LUCKEY            2  091581  073126                           4
600#       400000  1012100                                                5
599#       LUCKEY            3                                            6
600#       400000  1012200     650                                        7
999#                                                                      8
```

The first 599# record indicates that a test proposal that was added on the data 8/31/81 and time 12:45:36 is to be marked as rejected. The following 600# record gives the latitude and longitude of the test proposal that is to be rejected. If test proposals for other data blocks had been added in the same run as this one, other 600# records could follow. The second 599# record indicates that a test proposal added at the given date and time is to be

marked as accepted. The following 600# record gives its position. Other proposals added on that date and time could also be marked accepted with other 600# records. The third 599# record indicates that a test proposal is to be added. The position and test value is given on the following 600# record.

Output

The output consists of three print files. The first file is the standard print file. It contains information such as: person submitting proposal changes; parameter being updated; number of proposals rejected, accepted, or added; and a listing of the input request records.

The second file is called the activity report. It contains specific information on each proposal that is rejected, accepted, or added. For parameters stored as 10-minute or 1-minute data, this report gives:

A. For a rejected proposal:
   1. Latitude of rejected proposal
   2. Longitude of rejected proposal
   3. Date proposal was added to data base
   4. Time proposal was added to data base
   5. Rejected proposed-value
   6. Current permanent-value

B. For an accepted proposal:
   1. Latitude of accepted proposal
   2. Longitude of accepted proposal
   3. Date proposal added to data base
   4. Time proposal added to data base
   5. Previous permanent-value
   6. New accepted-value

C. For a test proposal:
   1. Latitude of test proposal
   2. Longitude of test proposal
   3. Date proposal added to data base
   4. Time proposal added to data base
   5. Proposed value
   6. Current permanent-value

For parameters stored as 6-second data, the information supplied in the activity report is more extensive. The report gives:

A. For a rejected proposal:
   1. Latitude of rejected proposal
   2. Longitude of rejected proposal
   3. Date proposal added to data base
   4. Time proposal added to data base
   5. A 10x10 matrix of the 100 values within the 1-minute by 1-minute block. The rejected proposed-values are highlighted with asterisks. The remaining values represent the current permanent-values for those 6-second by 6-second blocks.

B. For an accepted proposal:
   1. Latitude of accepted proposal
   2. Longitude of accepted proposal
   3. Date proposal added to data base
   4. Time proposal added to data base
   5. A 10x10 matrix of the 100 values within the 1-minute by 1-minute block. The new accepted-values are highlighted with asterisks. The remaining values represent the current permanent-values for those blocks.

C. For a test proposal:
   1. Latitude of test proposal
   2. Longitude of test proposal
   3. Date proposal added to data base
   4. Time proposal added to data base
   5. A 10x10 matrix of the 100 values within the 1-minute by 1-minute block. The new proposed-values are highlighted with asterisks. The remaining values represent the current permanent-values for those blocks.

The activity report also contains information on update requests that could not be accomplished. For example, if a proposed value is equal to the MVI, the proposal would not be added and the activity report would contain a message to this effect. If the proposal was for 6-second data, the message contains the position in the 100 6-second by 6-second blocks where the problem occurred. This location is a number between 1 and 100, where positions 1 to 10 describe the first row of data, positions 11-20, the second row of data, and so forth. Note that even if one of the proposed values is not allowable, the test proposal for the entire block will not be added.

The third file is for error messages. In the vast majority of cases, possible errors within the program are related to System 2000 (the DBMS) processing. These errors should not occur frequently; but, if an error does occur, a message is printed in the error file and the run is aborted. The DBA should be contacted, because some of these errors could result in the data base being damaged. When the data base is damaged, the Instant-Update Program cannot be run properly. If a severe error does occur, a warning is given in the standard print file that directs the user to look at the error message file.

Sample Procedure

The procedure used to execute the Instant-Update Program is called DTEDITIU. This procedure is a two-step process. The first step executes the Edit Program. The input to the Edit Program is a parameter record plus the update request records (that is, 002#, 599#, 600#, and so forth) for the Instant-Update Program. The Edit Program checks the syntax of these records as well as other types of checks. If a record is determined to be correct, the Edit Program writes this record to a file. This file is passed to the second step of the procedure, the execution of the Instant-Update Program.

Certain errors, such as incorrect person's name or parameter name, are considered to be fatal errors. If fatal errors are detected by the Edit Program, the second step of the procedure is not executed. Other errors may be severe enough to cause a particular group of records to be rejected, but not severe enough to prevent the second step from being executed. If, for instance, the rejected record is type 599#, then its corresponding 600# (and possibly 701# through 710#) records are also rejected, whether they are correct or incorrect. It is possible that only part of the update request records are passed to the second step of the procedure.

As part of its output, the Edit Program has a separate section of the printout where error messages are listed. These error messages are explained in the section on the Edit Program. The Edit Program output also contains the number of input records read, the number of errors detected, and the number of fatal errors. Fatal errors are errors that prohibit the second step from being executed.

To execute the Instant-Update Program, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTEDITIU,MEMBER=COCF266A
//GOED.SYSIN  DD  *
     Parameter record
     Update request records
//
```

On the EXEC statement (third record), the parameter MEMBER must be defined. This parameter is the name to be given to a member of a data set that will contain information about a particular execution of the Instant-Update Program. This information is used by the DBA to monitor the activity of the program. This parameter consists of 8 alphanumeric characters. Characters 1 and 2 are the state name abbreviation. Characters 3 and 4 are the user's initials. Characters 5 through 7 are the Julian day. Character 8 is a letter from A to Z to be used to differentiate among executions on the same day.

In the input, the Instant-Update Program request records are preceded by the Edit Program parameter record. This record contains information necessary to the proper execution of the Edit Program. The record is as follows:

Col 1-30	Title	7A4,A2	Title of run
Col 31-32	-	2X	Blank
Col 33-34	Input	I2	Should be set to 5
Col 35-36	-	2X	Blank
Col 37-38	Output	I2	Device where edited records are written, set to 11
Col 39-40	-	2X	Blank
Col 41-50	Options	10I1	Execution options:

Element	Value
1	0
2	0-print diagnostics 1-print all input records

			3	0
			4	4
			5-10	0 or blank
Col 51-52	-	2X	Blank	
Col 53-60	Study unit	2A4	Study-unit name	
Col 61-68	Data base	2A4	Data-base name	
Col 69-72	Password	A4	Password with update authority	
Col 73-76	-	4X	Blank	
Col 77-79	Userid	A3	User initials	
Col 80	Kode	A1	Set to P	

The procedure used to execute the Instant-Update Program is as follows:

```
//DTEDITIU PROC TIM1=1,TIM2=1,REG1=750K,REG2=770K,PROG1=DTEDIT3,
//            PROG2=DTIUPD,MEMBER=,SP1=1,SP2=10
//*********************************************************************
//*  DTEDITIU: A PROCEDURE TO RUN THE INSTANT-UPDATE PROGRAM
//*           IN CONJUNCTION WITH THE DATA BASE EDIT PROGRAM.
//*           STEP 1 TAKES THE INPUT TO THE INSTANT-UPDATE PROGRAM AND
//*           CHECKS THE CORRECTNESS OF THE DATA USING THE EDIT PROGRAM.
//*           IF THE EDIT IS SUCCESSFUL, STEP 2 IS EXECUTED. THIS IS
//*           THE INSTANT-UPDATE STEP.
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*    TIM1        TOTAL RUN TIME EDIT STEP (1)
//*    TIM2        TOTAL RUN TIME INSTANT-UPDATE STEP (1)
//*    REG1        REGION SIZE EDIT STEP (750K)
//*    REG2        REGION SIZE INSTANT-UPDATE STEP (770K)
//*    PROG1       LOAD MODULE NAME EDIT STEP (DTEDIT3)
//*    PROG2       LOAD MODULE NAME INSTANT-UPDATE STEP (DTIUPD)
//*    MEMBER      NAME OF PDS MEMBER FOR DBA INFORMATION FILE
//*
//*********************************************************************
//*
//*   STEP 1: EDIT
//*
//GOED EXEC PGM=&PROG1,TIME=&TIM1,REGION=&REG1
//STEPLIB  DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//         DD  DSN=SYS1.FORTG.LINKLIBX,DISP=SHR
//*
//*   DEFINE STANDARD I/O DEVICES
//*
//FT05F001  DD  DDNAME=SYSIN
//FT06F001  DD  SYSOUT=A
//FT07F001  DD  SYSOUT=B
//*
//*   DEFINE OUTPUT FILE
//*
//FT11F001  DD  DSN=&&EDOUT,UNIT=SYSDK,DISP=(,PASS),VOL=,
```

```
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120),SPACE=(TRK,(3,3),RLSE)
//*
//*  LOGICAL UNIT 20 USED FOR ERROR CODES AND MESSAGES FILE
//*  LOGICAL UNIT 21 USED FOR OUTPUT OF ERROR MESSAGES TO USER
//*
//FT20F001 DD  DSN=errorfile,DISP=SHR
//FT21F001 DD  SYSOUT=A
//*
//*  DEFINE FILES CONTAINING PARAMETER AND USERS' NAMES
//*
//FT22F001  DD  DSN=userfile,DISP=SHR
//FT23F001  DD  DSN=parmsfile,DISP=SHR
//*
//*  DEFINE FILES USED BY SYSTEM 2000
//*
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD DUMMY
//S2KCOMD  DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE01 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//LOCATE02 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=SHR
//dbname2 DD DSN=databasefile2,DISP=SHR
//dbname3 DD DSN=databasefile3,DISP=SHR
//dbname4 DD DSN=databasefile4,DISP=SHR
//dbname5 DD DSN=databasefile5,DISP=SHR
//dbname6 DD DSN=databasefile6,DISP=SHR
//*
//*
//IU EXEC PGM=&PROG2,TIME=&TIM2,REGION=&REG2,PARM='ISA(23K)',
//        COND=(13,LT,GOED)
//*
//*   STEP 2: INSTANT UPDATE
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//*
```

```
//*   DEFINE STANDARD I/O DATA SETS
//*
//DATAIN  DD  DSN=&&EDOUT,DISP=(OLD,DELETE)
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//*
//*   DEFINE REPORT AND ERROR FILES
//*
//REPORT  DD  SYSOUT=A
//REPORT2  DD  SYSOUT=A
//ERRFIL  DD  SYSOUT=A
//*
//*   DEFINE DBA INFORMATION FILE
//*
//DBAINFO  DD  DSN=dbainfofile(&MEMBER),DISP=OLD
//*
//*   DEFINE SYSTEM 2000 FILES
//*
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD DUMMY
//S2KCOMD  DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=OLD
//dbname2 DD DSN=databasefile2,DISP=OLD
//dbname3 DD DSN=databasefile3,DISP=OLD
//dbname4 DD DSN=databasefile4,DISP=OLD
//dbname5 DD DSN=databasefile5,DISP=OLD
//dbname6 DD DSN=databasefile6,DISP=OLD
```

The Move Program

The Instant-Update Program gives users of the DMS an opportunity to propose changes to values within the data base. Details of that process can be found in the documentation for the Instant-Update Program. With that program, the user can perform three tasks:

1. Add test proposals;
2. Accept test proposals; and
3. Reject test proposals.

When a test proposal is added to the data base, it is stored separately from the permanent data. A flag is set in the test proposal record to indicate that this is a test proposal. At some point, the user may want to mark a test proposal as rejected and have it removed from the data base. In this case, the Instant-Update Program changes the flag to indicate that the proposal is rejected; the program does not physically remove the proposal record from the data base. Alternatively, the user may want to mark a test proposal as an accepted proposal; the Instant-Update Program changes the flag to indicate that the proposal contains an accepted value that is to replace the current permanent-value. The program does not perform the replacement operation; the process of removing a rejected proposal from the data base and replacing a permanent value is a function of the Move Program. In all, the Move Program performs four tasks for each parameter:

1. Rejected proposals are removed from the data-base using the System 2000 (the DBMS) REMOVE TREE command.
2. Accepted proposals are located and the accepted values replace the present permanent-values, using the System 2000 MODIFY command.
3. The existing test proposals are examined and, if any proposals have existed longer than a set duration, they are removed.
4. If permanent values are changed, the parameter statistics are recomputed.

The Move Program has two modes of operation. The above tasks can be performed for all parameters in the data base or for selected parameters. The second mode of operation will usually be used because, at any one time, proposals likely will not exist for all the parameters in the data base. A guide to which parameters have proposals is found in the DBA information file produced by each execution of the Instand-Update Program.

The first step in processing a parameter is to retrieve information stored in the root-level record, including a flag that indicates whether proposals exist for the parameter. This flag is set by the Instant-Update Program and indicates precisely what types of proposals currently are within the data base. If the flag indicates that rejected proposals exist, then the program locates the rejected proposals and removes them from the data base. If the flag indicates that accepted proposals exist, then the program locates the accepted proposals and corresponding permanent data-records. Then, all five statistical components (minimum, maximum, value, number of values, and standard deviation) are replaced in the permanent data-record and the accepted proposal is removed from the data base. If the flag indicates that test proposals exist, they are examined, and if any of these proposals have existed longer than a set duration, they are removed from the data base.

Input

The first record contains a password having the proper update authority, the study-unit name, and a code indicating the mode of operation. If the code is 1, all the data-base parameters are processed and this record is the only input record. If the code is 2, only selected parameters are processed and the parameter names are input. The format of record 1 is as follows:

Col 1-4	Data-base password, left-justified
Col 5-12	Study-unit name
Col 13	Operation code

If the code is 2, then the parameter names are input, one per record, as follows:

Col 1-30	Parameter name, left-justified

If the code is 1, the parameter names are obtained from a file which contains the names of all parameters stored in the data base.

Output

The output consists of a printed report containing the data-base cycle information at the point when the data base is opened and at the point when all move operations have been performed. For each parameter processed, the report contains the following:

1. Name of parameter;
2. Root-level record statistics prior to the move operations;
3. Geographic coordinates of the blocks with new permanent-values;
4. Root-level record statistics after the move operations; these are listed only if the statistics are recomputed;
5. Number of rejected proposals removed;
6. Number of test proposals removed;
7. Number of accepted proposed-values moved to permanent part of the data base.

At the end of the report is a message that indicates whether the program terminated normally or abnormally. Abnormal termination would usually be the result of a failure in System 2000 (the DBMS) processing. If this message is printed, the computer staff responsible for the program should be contacted.

Sample Procedure

A procedure called DTMOVE is used to execute the Move Program. To execute the Move Program, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
```

```
//STEPNAME  EXEC  DTMOVE
//MOVE.DATAIN  DD  *
   input records
//
```

The procedure used to execute the Move Program is as follows:

```
//DTMOVE  PROC  TIM=2,REG=840K,PROG=DTMOVE,SP1=1,SP2=10
//*********************************************************************
//*  DTMOVE: A PROCEDURE TO RUN THE DATA BASE MOVE PROGRAM
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*   TIM      TOTAL RUN TIME (2)
//*   REG      REGION SIZE (840K)
//*   PROG     LOAD MODULE NAME (DTMOVE)
//*
//*********************************************************************
//MOVE  EXEC  PGM=&PROG,TIME=&TIM,REGION=&REG,PARM='ISA(20K)'
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//*
//*  DEFINE STANDARD I/O DATA SETS
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//*
//*  DEFINE FILE CONTAINING PARAMETER NAMES
//*
//PARMS  DD  DSN=parmsfile,DISP=SHR
//*
//*  DEFINE SYSTEM 2000 FILES
//*
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP  DD DUMMY
//S2KCOMD  DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
```

```
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=OLD
//dbname2 DD DSN=databasefile2,DISP=OLD
//dbname3 DD DSN=databasefile3,DISP=OLD
//dbname4 DD DSN=databasefile4,DISP=OLD
//dbname5 DD DSN=databasefile5,DISP=OLD
//dbname6 DD DSN=databasefile6,DISP=OLD
```

The Statistics Program

When a parameter is initially loaded into the data base, the statistical components (minimum, maximum, average, number of values, and standard deviation) are assigned values only at the base level of the data-base structure. Of the statistical components, the average is the only component that must be assigned a value. Calculating all the statistical components at the levels above the base level of the data-base structure is the function of the Statistics Program. This program is run after the Load Program, the General-Update program, and is usually run after the Move Program.

The Statistics Program calculates statistics for one parameter per program execution. The process by which these statistics are computed is basically the same for each data density except for a slight variation for 6-second data. Using the values at the base level, statistics are computed at each higher level. For example, to calculate statistics for a 10-minute statistical record at level 2 for 1-minute data, all values for 1-minute by 1-minute blocks within a 10-minute by 10-minute block are used. From these values, the minimum, maximum, average, number of values, and standard deviation are computed for the 10-minute statistical record. These five statistics are then placed in the data base in the record representing the 10-minute by 10-minute block. Similarly, values for all of the 1-minute by 1-minute blocks within a 1-degree by 1-degree block are used to calculate the same five statistics for the 1-degree statistical record at level 1. This process continues and a set of five statistics that reflect all the 1-minute by 1-minute blocks for a parameter are stored in the root-level record.

There is a slight variation in the procedure for 6-second data; because, when 6-second data are loaded, statistics for the 1-minute statistical records, at level 3, are calculated by the Load Program. The Statistics Program takes advantage of this fact when statistics are computed at higher levels of the data-base structure. For example, the sum of the 6-second values can be calculated by multiplying the number of values by the average found on the 1-minute statistical record. This sum can be used to calculate the overall average for the 10-minute by 10-minute and 1-degree by 1-degree blocks. Hence, the statistics for the 10-minute and 1-degree statistical records can be determined without actually retrieving the 6-second values. This procedure saves a great deal of processing. As with the other densities, the end result is statistics that reflect all the 6-second by 6-second blocks within the study area for that parameter. These final statistics are stored in the root-level record.

Input

The input is one record containing the following:

Col 1-4	Data-base password with the proper update authority
Col 5-12	Study-unit name
Col 13-42	Parameter name

Output

The output consists of a 1-page printed report providing basic information about the calculated statistics. The output includes the study unit and parameter names for which the statistics are being computed. In addition, data-base cycle information is provided for the points at which the data base is opened and at which the data base is closed after the statistics are successfully computed. Finally, the computed root-level statistics are printed.

Sample Procedure

A procedure called DTSTAT is used to execute the Statistics Program. To execute the Statistics Program, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//STEPNAME  EXEC  DTSTAT
//STAT.DATAIN  DD  *
       single input record
//
```

The procedure used to execute the Statistics Program is as follows:

```
//DTSTAT  PROC  TIM=2,REG=670K,PROG=DTSTAT,SP1=1,SP2=10
//*********************************************************************
//*  DTSTAT: A PROCEDURE TO RUN THE DATA BASE STATISTICS
//*          PROGRAM
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*   TIM      TOTAL RUN TIME (2)
//*   REG      REGION SIZE (670K)
//*   PROG     LOAD MODULE NAME (DTSTAT)
//*
//*********************************************************************
//STAT  EXEC  PGM=&PROG,TIME=&TIM,REGION=&REG,PARM='ISA(5K)'
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//*
//*  DEFINE STANDARD I/O DATA SETS
```

```
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP   DD  DUMMY
//*
//*  DEFINE SYSTEM 2000 FILES
//*
//S2KMSG    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP   DD DUMMY
//S2KCOMD   DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDK,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDK
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//S2KSYS07 DD DUMMY
//SF01      DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF02      DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF03      DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF04      DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//SF05      DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(S2KSYS03))
//SF06      DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDK,SEP=(SF01))
//dbname1 DD DSN=databasefile1,DISP=OLD
//dbname2 DD DSN=databasefile2,DISP=OLD
//dbname3 DD DSN=databasefile3,DISP=OLD
//dbname4 DD DSN=databasefile4,DISP=OLD
//dbname5 DD DSN=databasefile5,DISP=OLD
//dbname6 DD DSN=databasefile6,DISP=OLD
```

The Reload Program

The Reload Program is designed to unload the data from an existing data base (i.e. a data base as described by this report) and to then load the data into a new data base. For purposes of the following discussion, the existing data base will be called the 'old' data base. The two main purposes for using this program are to reorganize the old data base or to place the old data base on a storage device that has different efficient block sizes for files. In either case, the new data base is completely reorganized. All tables, such as the index and distinct-value tables, are reconstructed and compacted. The data table is also compacted and all reusable space is eliminated. For the new data base, the data-base cycle number, at the end of the program execution, is equivalent to the number of parameters stored in the old data base. The program is set up to reload 1-degree, 10-minute, 1-minute, and 6-second data.

Since the other programs in the data-management system are set up to work with the old data base, the next step in the process would be to save the new data base and to restore it to the old data base. When this has been done, the old data base will be available in its reorganized form.

It is possible to reload or reorganize a data base using the System 2000 (the DBMS) REORGANIZE and RELOAD commands. However, it has been determined that these methods are not as efficient as using the Reload Program. The Reload Program requires less processing time, computer memory, and scratch file usage.

Input

The only input to the program are the master passwords for the data bases. Both passwords are input on the same record. Columns 1-4 are for the new data-base password and columns 5-8 are used for the old data-base password. The passwords are read from data set DATAIN.

Output

The output consists of a list of the parameters that were reloaded into the new data base.

Sample Procedure

A procedure called RLDDTL is used to execute the Reload Program. To execute the Reload Program, the following JCL should be used:

```
//... JOB ...
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  RLDDTL
//GO.DATAIN  DD  *
   master passwords
```

The procedure used to execute the Reload Program is as follows:

```
//RLDDTL  PROC  REGG=1500K,TIMEG=50,SP1=1,SP2=10,PRMS=XPARMS
//*********************************************************************
//*   RLDDTL: A PROCEDURE TO RUN THE DATA BASE RELOAD PROGRAM
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*   TIMEG    TOTAL RUN TIME (50)
//*   REGG     REGION SIZE (1500K)
//*   PRMS     SYSTEM 2000 BUFFER POOLS (XPARMS)
//*
//*********************************************************************
//*
//GO  EXEC  PGM=RLDDTL,REGION=&REGG,TIME=&TIMEG,PARM='ISA(22K)'
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=SYS1.S2K,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//SYSLIB  DD  DSN=SYS1.PLIBASE,DISP=SHR
//        DD  DSN=SYS1.S2K,DISP=SHR
```

```
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP   DD  DUMMY
//S2KMSG    DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=132,BLKSIZE=1320)
//S2KPARMS DD DSN=s2kparmsfile,DISP=SHR
//S2KSNAP   DD DUMMY
//S2KCOMD   DD DUMMY
//S2KUDUMP DD DUMMY
//LOCATE00 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//LOCATE01 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//S2KSYS01 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDA
//S2KSYS02 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDA
//S2KSYS03 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS01))
//S2KSYS04 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS01))
//S2KSYS05 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS01))
//S2KSYS06 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS03))
//S2KSYS07 DD SPACE=(CYL,(&SP1,&SP2)),UNIT=SYSDA
//SF01     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS03))
//SF02     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(SF01))
//SF03     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS03))
//SF04     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(SF01))
//SF05     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(S2KSYS03))
//SF06     DD SPACE=(CYL,(&SP1,&SP2)),UNIT=(SYSDA,SEP=(SF01))
//olddb1   DD DSN=databasefile1,DISP=OLD
//olddb2   DD DSN=databasefile2,DISP=OLD
//olddb3   DD DSN=databasefile3,DISP=OLD
//olddb4   DD DSN=databasefile4,DISP=OLD
//olddb5   DD DSN=databasefile5,DISP=OLD
//olddb6   DD DSN=databasefile6,DISP=OLD
//newdb1   DD DSN=newdatabasefile1,DISP=OLD
//newdb2   DD DSN=newdatabasefile2,DISP=OLD
//newdb3   DD DSN=newdatabasefile3,DISP=OLD
//newdb4   DD DSN=newdatabasefile4,DISP=OLD
//newdb5   DD DSN=newdatabasefile5,DISP=OLD
//newdb6   DD DSN=newdatabasefile6,DISP=OLD
```

Note: In the above procedure, *olddb* is the first 7 characters of the old data-base name and *newdb* is the first 7 characters of the new data-base name.

The Data-Transformation Program

The Data-Transformation Program prepares the input for the finite-difference ground-water flow model (Trescott and others). The program is divided into two phases. The first phase converts data retrieved from the data base into (X, Y, value) triplets. The second phase uses the (X, Y, value) triplets to calculate a value for each block within a model.

In the first phase, latitude and longitude are converted to plane coordinates X and Y by Lambert's Conformal Projection, with standard parallels at 33 and 45 degrees, and with the central meridian at 96 degrees west of Greenwich. These results are then rotated by an angle specified by the user.

The second phase looks at each (X, Y, value) triplet that was determined in phase 1 and locates the model block which contains the point. After all the points have been properly located within the model, an interpolation is performed to determine a value for each model block. This interpolation may take the form of an average, weighted-average, or trend-surface analysis.

Prior to executing this program, the user must first retrieve data from the data base. Details of this retrieval can be found in the documentation for the Retrieval Program.

Criteria for Retrieving from Data Base

The amount of data retrieved depends on the selected interpolation technique. The average or weighted-average interpolation techniques will use only data that fall within the model to compute values for the model matrix. However, the program does detect if the data retrieved fall outside the model, and the program will eliminate these data from any further use. This is done because it is not likely that the user can retrieve only the data that fall within the model.

For trend-surface analysis, all retrieved data are used unless the analysis is done with 6-second data. If the user wants results in the model matrix to reflect the "local anomalies" within the model, then the retrieval should be limited to the area covered by the model. If the user is more interested in "regional trends," the retrieval can cover a larger area. The area retrieved to determine regional trends can be as large as the entire study area. However, if the retrieved data cover an area larger than the model, the resultant model matrix may not reflect the values within the model as they are stored in the data base. This is because the model matrix was determined using values outside the model area.

For 6-second data, the program limits the trend-surface analysis to data that falls within the model. There are two reasons for this limitation. First, in many cases, the trend-surface technique is not the best choice for use with 6-second data. There may be a sufficient number of 6-second values within the model to use another interpolation technique. Second, if all the retrieved 6-second data were used to compute the trend surface and the model matrix, the computer costs could possibly get very large because of the extensive amount of processing performed by the program. Because of this, prior to doing an extensive amount of processing, the program makes an estimate of the number of 6-second values within the model. If this estimate is greater than 1600 values, the run is aborted, and the user is asked to rerun the program using a different interpolation technique.

When using average or weighted-average interpolation, the user has three additional options for 6-second data. First, only 6-second values can be used to determine the model matrix. Second, only the 1-minute averages for the 6-second data can be used to determine the model matrix. Third, a mixture of 6-second values and 1-minute averages can be used. For the first and third options, 6-second data are retrieved from the data base; for the second

option, the 1-minute statistics for 6-second data are retrieved. This option is explained in the documentation for the Retrieval Program. For trend-surface analysis, the program does not allow a mixture of 6-second values and 1-minute averages.

Model-Core Determination

Model-core determination is a process that is used with 6-second data. The process divides the model into two regions. The first region is called the model core and is the area where 6-second values will be used to compute the model matrix. The second region, that portion of the model that borders the core, is the area where 1-minute averages for the 6-second data will be used to compute the model matrix.

The model core is used because of the possibility that, using 6-second data, the model could contain a very large amount of data. With that large amount of data, computer processing time and costs could become excessive. By using 1-minute averages in the outer rows and columns of the model, processing time and cost is greatly reduced.

The first step in automatic determination of the model core is to calculate the area of a 1-minute by 1-minute block at the latitude and longitude of the principal node. This area, the standard comparison area, contains exactly one 1-minute value. Then, starting at the block that contains the principal node and proceeding outward along the column that contains the principal node, areas of the model blocks are calculated. The ratio of the area of the block to the standard comparison area yields an estimate of the number of 1-minute values within that block. If that number is less than a predetermined number, usually five, then that block is considered to be within the model core. This process continues along the column of the principal node (in both directions) until the model core extent is determined. The first block at which the estimate of the number of 1-minute values equals or exceeds five is considered to be the first block outside the model core. This procedure is repeated in both directions along the row that contains the principal node. The end result of this process is the row and column extremes of the model core. The number five was chosen as the comparison value, because, with this number of 1-minute values within a block, a reasonable result can be computed for the model matrix by using an average or weighted-average interpolation.

This procedure does have limitations. For many model configurations, this process will not be able to determine a model core. This can occur if the blocks are very small in the central region of the model. This could lead to small blocks being found in the peripheral areas of the model. Under these circumstances, the outer blocks along the column and row of the principal node could easily contain less than five 1-minute values. To alleviate this problem, the program allows the user to optionally input the row and column extremes of the model core. In this case, the program calculates the model core as requested by the user. In addition, if the user wants the program to compute the model core, the comparison number of points can be changed. By changing this value, it is possible that the program could successfully compute the model core.

Interpolation Techniques

There are three interpolation techniques available for determining the model matrix: average, weighted-average, and trend-surface analysis (Davis, 1973). Use of each of these techniques depends on several factors. In general, average and weighted-average techniques are useful when the density of data within the model is large, while the trend-surface analysis is useful when the data are sparse.

There are two factors that will greatly effect the density of data within a model: (1) Density of the data as it is stored within the data base, and (2) distance between the nodes of the model. A general guideline for determining which technique to use is: if the distance between nodes in the model is less than the distance between data in the data base, then the trend-surface analysis would be the appropriate choice. In this case, it is likely that only one or possibly no data points fall within a model node and an average or weighted-average interpolation may not be appropriate.

With these facts in mind, we recommend the following. With 10-minute data, generally use the trend-surface technique. However, with a very coarse model grid it will be possible to get relevant results by using an average or weighted-average interpolation. With 1-minute and 6-second data, generally use an average or weighted-average interpolation unless the model grid is very fine.

The program allows the user to do a trend-surface analysis with 6-second data. However, if the program estimates that there are more than 1600 6-second values within the model, the run will be aborted and a message will be printed suggesting the user rerun the program with a request for a different interpolation technique. This is done, because, with more than 1600 values in the model, one will most likely obtain better results with an average or weighted-average interpolation.

Average

For each (X, Y, value) triplet, the averaging routine determines which model block contains the point. For each block within the model, a sum of the values within the block is calculated. After all the points have been located within the model, the sum of the values is divided by the number of points within the block and this result is assigned to that model block. These results are output as the model matrix.

Weighted Average

The weighted-average routine is very similar to the average routine. The difference occurs after a point is located within the model. The distance between the point and the center of the model block which contains the point is calculated. The program calculates: (1) The data-base value divided by the square of the distance and (2) the reciprocal of the square of the distance. Sums of each of these results are kept for each model block. After all the points have been properly located, these quantities are divided and the result is assigned as the value for that model block. The calculation for each block is as follows:

$$\text{Value} = \Sigma\ (\text{value}/D^2)\ /\ \Sigma\ (1/D^2)$$

where D^2 = square of the distance and the sums are over all the points in the model block.

For both of the above techniques, it is possible that some of the model blocks will not have any available data for computing a value. If the output format is F10.4, the block will be assigned a -999999999. If the format is F4.0, the block will be assigned a -999.

If a point is located on the line between two columns of the model, the point is used to determine the value for both blocks. The same thing is done if the point falls on the line between two rows of the model.

Trend Surface

This interpolation routine works quite differently from the others. The user must input the order of the polynomial that is to be determined by the program. If the order is one, the data will be fitted to a plane surface. Orders of two, three, and four fit the data to more complicated surfaces. A limitation of four has been placed on the order of the polynomial. If a number larger than four is input, the program will be aborted.

With the order of the polynomial, the number of coefficients to be determined and the number of points required are:

Order	Coefficients	Number of points required
1	3	3
2	6	4
3	10	5
4	15	6

The trend-surface technique uses all of the data to determine the polynomial. Values for each model block are obtained by solving the polynomial using the X and Y of the center of the block. This value is then placed in the model matrix. All blocks within the model are assigned a value.

As the trend-surface routine is determining the polynomial, it keeps count of the number of points being used. If this number is larger than 500, the program prints a message indicating the number of points used. This is done because with more than 500 points within the model, it is possible that the average or weighted-average technique will give better results. The routine does not abort and it will use all the points to compute the polynomial.

General Information

All three interpolation programs allow two different output formats for the model matrix--8F10.4 and 20F4.0. The first output format poses no problem in that all significant digits of the result can fit within that format. The

second output format poses a problem in that many of the values have more than four significant digits. The program outputs the four most-significant digits. A message is printed indicating the factor by which each value was multiplied to output these four most-significant digits.

Program Flow

Flow of the program depends on the density of the data retrieved from the data base:

1. Ten-minute Data: With 10-minute data, the model grid may be regular or irregular. Because the amount of data-base input is usually small, no model-core determination is necessary nor is it an allowable option. If the user requests average or weighted-average interpolation, the program will eliminate any data that does not fall into the model area. If trend-surface analysis is requested, all the data are used for determining the model matrix.

2. One-Minute Data: With 1-minute data, the model grid may be regular or irregular. Because the amount of data-base input is usually small, no model-core determination is necessary nor is it an allowable option. If the user requests average or weighted-average interpolation, the program will eliminate any data that does not fall within the model area. If trend-surface interpolation is requested, all the data are used.

3. Six-Second Data: With 6-second data, the model grid may be regular or irregular. The program flow depends on the value of the second execution option.

If this option is set to 0, then the 1-minute averages are to be used to determine the model matrix. The user must have retrieved the 1-minute statistics for the 6-second data from the data base. If this option is chosen and the retrieved data contain the 6-second data records, the run will be aborted. If the user requests average or weighted-average interpolation, the program will eliminate any data outside the model area. For trend-surface analysis, all the data are used to compute the model matrix.

If the option is set to 1, then only the 6-second values will be used. If average or weighted-average interpolation is requested, the program will eliminate any data outside the model. If trend-surface analysis is chosen, the program first makes an estimate of the number of 6-second values within the model. If this estimate exceeds 1600, the run is aborted, with the suggestion that the run be resubmitted for a different interpolation technique. If the estimate is less than 1600 values, the program will continue processing, but will use only the data that fall within the model to compute the model matrix.

If the option is set to 2, then a mixture of 6-second values and 1-minute averages will be used. With this option, a model core is determined. This can be done in two ways. The user can request that the program compute the model core, or the user can input the row and column extremes of the model core. If average or weighted-average interpolation is requested, then the program eliminates data that fall outside the model area. For a mixture of 6-second values and 1-minute averages, trend-surface analysis is not allowed.

Special Considerations

This program is divided into two steps. To prevent the second step from being run if the first step is aborted, certain condition codes are set if a severe error occurs during execution. The condition code is composed of four digits. The leftmost digit is reserved for system use. The two steps set the other three digits to 777 or 888 if an error occurs and the run is to be aborted.

A condition code of 888 implies that one or more of the input values was incorrect. This condition code is also set if the user requests something that the program cannot perform. For example, if the user requests trend-surface analysis with 6-second data and the program estimates that there are more than 1600 values within the model, the run will be aborted and a condition code of 888 is set. With a condition code of 888, the program can be rerun after the input is corrected.

A condition code of 777 indicates an error beyond the control of the user. It usually indicates a logic fault or a system problem has occured. If this condition code is raised, the user should contact the computer staff responsible for the program.

Input and Output

The input and output is divided between two steps.

Step 1

Input: Read from data set DATAIN

Record (1) - Execution options

Col 1	-	0	Regular grid pattern
		1	Irregular grid pattern
Col 2	-	blank	If the parameter is stored as 1-minute or 10-minute data
		0	1-minute averages used
		1	6-second values used
		2	Mixture of 6-second values and 1-minute averages
Col 3	-	blank	Model-core determination not selected
		0	Program will generate model core
		1	User inputting model core
Col 4	-	0	Model-matrix output format of 8F10.4
		1	Model-matrix output format of 20F4.0
Col 5	-	1	Average interpolation
		2	Weighted-Average interpolation
		3	Trend-Surface analysis

Record (2) - Description of principal node. The principal node is any arbitrary node in the model generally near the center.

Col 1-6	Latitude of principal node (DDMMSS)
Col 7-13	Longitude of principal node (DDDMMSS)
Col 14-16	Row of model for principal node (I3)
Col 17-19	Column of model for principal node (I3)
Col 20-26	Rotation angle of the model columns from north in decimal degrees, positive for counterclockwise rotation (F7.4)

Record (3) - Number of rows and columns in grid

Col 1-3	Number of rows in grid (I3)
Col 4-6	Number of columns in grid (I3)

The fourth input record depends on the value of the first execution option. If regular grid option set:

Record (4) - Row and column spacings

Col 1-10	Column spacing-value in feet (F10.0)
Col 11-20	Row spacing-value in feet (F10.0)

If irregular grid option set:

Record (4) - Row and column spacings

Col 1-80	Column spacing-values in feet (8F10.0)
Col 1-80	Row spacing-values in feet (8F10.0)

The fifth record is used if a mixture of 6-second values and 1-minute averages are used to compute the model matrix. If the program determines the model core, the user must input the number of 1-minute values needed in a model block to use the 1-minute averages to compute a value for a block:

Record (5) - Comparison number of one-minute values

Col 1-2	Number of 1-minute values (I2)--usually set to 5

If the user is inputting the model core description, the record is as follows:

Record (5) - Description of model core

Col 1-3	Top row of model core (I3)
Col 4-6	Bottom row of model core (I3)
Col 7-9	Right column of model core (I3)
Col 10-12	Left column of model core (I3)

Data-Base Input: read from data set DBINPT

Step 2

Input: Read from data set DATAIN

Record (1) - Same as record (3) of step 1
Record (2) - Same as record (4) of step 1

If trend-surface analysis is requested a third record is added containing the order of the polynomial:

Record (3) - Order of polynomial

Col 1	Order of polynomial (I1)

Transformed Data-Base Data (output from step 1):

If execution option two is not set to 2, read from data set DBINPT.
If execution option two is set to 2, then 1-minute statistics for 6-second data are read from DBINPT and 6-second data read from DBINPT2.

Output:

Messages - written to SYSPRINT
Model Matrix - written to MODDATA

Sample Procedure

A procedure called DTDTIP is used to execute the Data-Transformation Program. To execute the Data-Transformation Program, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTDTIP,NAME1=input
//DTP.DATAIN  DD *
     input for the first step
//INTERP.DATAIN  DD  *
     input for the second step
//
```

The procedure used to execute the Data-Transformation Program is as follows:

```
//DTDTIP  PROC  TIME1=2,TIME2=1,REG1=2200K,REG2=190K,PROG1=DTDTP,
//           PROG2=DTINTERP,UNIT1=3350,VOL1=myvol,NAME1=NULLFILE,
//           UNIT2=SYSDK,VOL2=,NAME2=,UNIT3=SYSDK,VOL3=,NAME3=
//*******************************************************************************
//*  DTDTIP: A PROCEDURE TO RUN THE DATA-TRANSFORMATION AND
//*        INTERPOLATION PROGRAM
```

```
//*        PROCEDURE DIVIDED INTO TWO STEPS
//*        STEP 1 TRANSFORMS DATA-BASE INPUT TO MODEL COORDINATES
//*        STEP 1 CALLED DTP
//*        STEP 2 USES AN INTERPOLATION TECHNIQUE TO COMPUTE THE MODEL
//*        MATRIX. STEP 2 CALLED INTERP.
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*  TIME1    TOTAL RUN TIME FOR DTP STEP (2)
//*  TIME2    TOTAL RUN TIME FOR INTERP STEP (1)
//*  REG1     REGION SIZE FOR DTP STEP (2200K)
//*  REG2     REGION SIZE FOR INTERP STEP (190K)
//*  PROG1    LOAD MODULE FOR DTP STEP (DTDTP)
//*  PROG2    LOAD MODULE FOR INTERP STEP (DTINTERP)
//*  UNIT1    RETRIEVED DATA-BASE DATA DEVICE (3350)
//*  VOL1     RETRIEVED DATA-BASE DATA VOLUME (myvol)
//*  NAME1    RETRIEVED DATA-BASE DATA DSNAME (NULLFILE)
//*  UNIT2    DTP STEP FIRST OUTPUT FILE DEVICE (SYSDK)
//*  UNIT2    INTERP STEP FIRST INPUT FILE DEVICE (SYSDK)
//*  VOL2     DTP STEP FIRST OUTPUT FILE VOLUME (BLANK)
//*  VOL2     INTERP STEP FIRST INPUT FILE VOLUME (BLANK)
//*  NAME2    DTP STEP FIRST OUTPUT FILE DSNAME (BLANK)
//*  NAME2    INTERP STEP FIRST INPUT FILE DSNAME (BLANK)
//*  UNIT3    DTP STEP SECOND OUTPUT FILE DEVICE (SYSDK)
//*  UNIT3    INTERP STEP SECOND INPUT FILE DEVICE (SYSDK)
//*  VOL3     DTP STEP SECOND OUTPUT FILE VOLUME (BLANK)
//*  VOL3     INTERP STEP SECOND INPUT FILE VOLUME (BLANK)
//*  NAME3    DTP STEP SECOND OUTPUT FILE DSNAME (BLANK)
//*  NAME3    INTERP STEP SECOND INPUT FILE DSNAME (BLANK)
//*
//*******************************************************************************
//DTP  EXEC  PGM=&PROG1,TIME=&TIME1,REGION=&REG1,PARM='ISA(1882K)'
//*
//*  STEP 1:  DATA TRANSFORMATION
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//*
//*   DEFINE STANDARD I/O DATA SETS
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//FT06F001  DD  SYSOUT=A
//*
//*   DEFINE DATA-BASE INPUT DATA SET
//*
//DBINPT  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*
//*   DEFINE OUTPUT DATA SETS FOR TRANSFORMED DATA
//*
//DBOTPT  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DSNAME=&NAME2,
//            DISP=(NEW,PASS,DELETE),
//            DCB=(RECFM=FB,LRECL=20,BLKSIZE=9440),
```

```
//             SPACE=(TRK,(5,5))
//DBOTPT2  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DSNAME=&NAME3,
//             DISP=(NEW,PASS,DELETE),
//             DCB=(RECFM=FB,LRECL=20,BLKSIZE=9440),
//             SPACE=(TRK,(5,5))
//INTERP   EXEC  PGM=&PROG2,TIME=&TIME2,REGION=&REG2,PARM='ISA(52K)',
//         COND=(8,LT,DTP)
//*
//*   STEP 2: INTERPOLATION
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//*
//*   DEFINE STANDARD I/O DATA SETS
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP   DD  DUMMY
//*
//*   DEFINE INPUT DATA SETS FOR TRANSFORMED DATA
//*
//DBINPT   DD  UNIT=&UNIT2,VOL=SER=&VOL2,DSNAME=*.DTP.DBOTPT,
//             DISP=(OLD,PASS)
//DBINPT2  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DSNAME=*.DTP.DBOTPT2,
//             DISP=(OLD,PASS)
//*
//*   DEFINE OUTPUT DATA SET FOR MODEL MATRIX
//*
//MODDATA  DD  SYSOUT=B,DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
```

The Data-Manipulation Program

The Data-Manipulation program mathematically manipulates data. There are two main uses for the program: (1) It can be used to manipulate one or two existing parameters for use in other application programs, and (2) it can be used to create new parameters for input into the data base. The program provides five standard functions for manipulating data:

LINEAR Function: A*X1 + B*X2 + C
MULTIPLICATION Function: A*X1*X2 + C
DIVISION Function: A*(X1/X2) + C
LOGARITHM Function: $A*LOG_{10}(X1) + C$
ANTILOGARITHM Function: $A*ANTILOG_{10}(X1) + C$

where X1 and X2 represent data-base format data and A, B, and C are constants. Additional standard functions can be added to the program as the need arises.

With these five standard functions, many different algebraic combinations of two data sets can be performed. An example would be to divide two data sets and raise the result to a power N:

$$Y = (X1/X2)^N .$$

The result for this algebraic combination is computed in 4 steps:

1. Divide X1 by X2;
2. Compute the logarithm of the results of step 1;
3. Multiply the results of step 2 by N; and
4. Compute the antilogarithm of the results of step 3.

An alternative would be to express the above equation as:

$$LOG_{10}(Y) = N*LOG_{10}(X1) - N*LOG_{10}(X2) .$$

The result for this algebraic combination is also computed in 4 steps:

1. Compute the logarithm of X1 and multiply result by N;
2. Compute the logarithm of X2 and multiply result by N;
3. Subtract the results of step 2 from step 1; and
4. Compute the antilogarithm of the result of step 3.

Each of the above 4 steps is a separate execution of the Data-Manipulation Program.

With the Data-Manipulation Program, the user can bypass the standard functions and input an arithmetic expression that is to be used to manipulate the data. This expression must be a valid PL/1 arithmetic expression. This expression is limited to the manipulation of 1 or 2 data-base format data sets. The variables X1 and X2 must be used to represent the data. The expression can contain any number of constants. Of these constants, three of them may be represented by the variables A, B, and C. The values of these three constants are input separate from the arithmetic expression.

The arithmetic expression can incorporate any of the standard arithmetic operations which include addition, subtraction, multiplication, division, and exponentiation. In addition, the expression can contain references to many of the standard mathematical functions such as the trigonometric and hyperbolic functions.

By allowing the use of standard mathematical functions within the optional arithmetic expression, the possibility exists that computational errors, such as attempting to compute the square root of a negative number or division by zero, may occur. The program is designed to detect these types of errors. The program specifically detects the following error types:

1. Division by zero;
2. Underflow;
3. Overflow; and
4. Computational errors.

An underflow error results from computing a number that is too small to be stored by the computer and overflow results from computing a number that is too large to be stored in the computer. The Data-Manipulation Program allows the user to accumulate 10 errors of each type. After 10 errors of any type are detected, the program execution is aborted.

Input

The mathematical function to be performed and all other parameters are input on one or, optionally, two input records. The first input record is always required and it contains a maximum of six fields. One field describes the function to be performed. This is the FNCTION field and can contain one of six values:

LINEAR	(A*X1 + B*X2 + C)
MULTIPLY	(A*X1*X2 + C)
DIVIDE	(A*(X1/X2) + C)
LOG	(A*LOG_{10}(X1) + C)
ANTILOG	(A*$ANTILOG_{10}$(X1) + C)
USERDEF	

The value, USERDEF, will be explained later. If the FNCTION field does not contain one of the above values, the run is aborted. The next three fields contain the values of the constants A, B, and C. There are no default values for these constants. For each function, the required constants must be assigned values. For the functions LOG, ANTILOG, MULTIPLY and DIVIDE, the constants A and C must be defined. For the LINEAR function, A, B, and C must be assigned values. Failure to define one or more of the required constants will cause the run to be aborted. The next field is called PARM. This is a field of up to 30 alphanumeric characters that define the parameter name for the output of the program. The sixth field is called NUMBER and it indicates the number of data-base format data sets to be manipulated. It can take the values of 1 or 2 only.

These six fields are input in a free-field format. They can be placed on one or more records. The fields are separated by one or more blanks or by commas. The last field must be followed by a semicolon. There is no specific order in which the fields must be defined. An example is as follows:

```
FNCTION='LINEAR' A=1 B=1 C=0 PARM='PUMPAGE - 1978' NUMBER=2;
```

If the value USERDEF is specified for the FNCTION field then, as well as defining the appropriate fields from among A, B, C, PARM, and NUMBER, an additional record needs to be input that specifies the arithmetic expression that is to be used to manipulate the data-base data. This expression may contain up to 61 characters (including blanks) and it must be a valid PL/1 arithmetic expression. The data-base data must be represented by the variables X1 and X2 where X1 is a value from data set 1 and X2 is a value from set 2. The expression can contain any number of constants. Of these constants, only three of them may be represented by variables. The variable names for these three constants must be A, B, and C. The following symbols must be used for arithmetic operations:

Addition:	+
Subtraction:	-
Multiplication:	*
Division:	/
Exponentiation:	**

The arithmetic expression may contain references to mathematical functions. The most common functions are:

ACOS	- Inverse (arc) cosine in radians
ASIN	- Inverse sine in radians
ATAN	- Inverse tangent in radians
ATAND	- Inverse tangent in degrees
ATANH	- Inverse hyperbolic tangent
COS	- Cosine of angle expressed in radians
COSD	- Cosine of angle expressed in degrees
COSH	- Hyperbolic cosine
ERF	- Error function
ERFC	- Complement of error function
EXP	- The base, e, raised to a power
LOG	- Natural logarithm
LOG2	- Binary logarithm
LOG10	- Common logarithm
SIN	- Sine of angle expressed in radians
SIND	- Sine of angle expressed in degrees
SINH	- Hyperbolic sine
SQRT	- Square root
TAN	- Tangent of angle expressed in radians
TAND	- Tangent of angle expressed in degrees
TANH	- Hyperbolic tangent

The arithmetic expression can be used to manipulate 1 or 2 data sets. The actual expression is input in the same free-field format as the six fields previously described. It can be placed on the same record following the semicolon that is found at the end of the description of the first six fields or it can be placed on a separate record. The expression is input as follows:

```
EXPRESSION='ARITHMETIC EXPRESSION';
```

Examples are as follows:

```
EXPRESSION='A*SQRT(X1) + C';
EXPRESSION='5.0*SQRT(X1) + 2.6';
EXPRESSION='A*COS(X1) + B*SIN(X2) + C';
```

The other major input to the Data-Manipulation Program is the data file(s) upon which the mathematical manipulations are to be performed. These data must be in the standard data-base format, which means that the file must begin with the 001#, 002#, and 003# records. These records are then followed by the proper combination of 100#, 200#, 300#, 401#-410# records, depending on the density of the data. These data may be derived in three ways. The data can be retrieved from the data base and, by using this method, the data are automatically in the proper format. The input data could also be the output from a previous run of the Data-Manipulation Program. In this case, the data are also in the proper format. Finally, the user can create the data set. If this method is used, the data should be edited prior to input into the Data-Manipulation Program, using the Edit Program.

For the functions LOG and ANTILOG, only one data set is input. The program extracts the parameter name, data density, scale factor, and the missing-value indicator (MVI) from the 001#, 002#, and 003# records. By using the data-density value, the program determines which of the records contains the values that are to be used in computing the log or antilog.

For the LINEAR, MULTIPLY, and DIVIDE functions, two distinct data sets are input. These data sets need not exactly match; one of the data sets can contain more or less data than the other. The program only performs the data manipulations for the data that are common to both data sets.

If two data sets are being processed, the user must specify the intersection of the two data sets using geographic coordinates. The geographic coordinates are given as whole degrees and specified in the form DDMMSS for latitude and DDDMMSS for longitude. The values are placed on one input record in the order: minimum latitude (columns 1-6), maximum latitude (columns 7-12), minimum longitude (columns 13-19), and maximum longitude (columns 20-26). These values are read from a data set called LLLIMIT. The data manipulations will only be performed in the area covered by the intersection of the data sets.

Output

The output is in two forms. The first is the standard print file that contains information such as the selected function, the new scale factor and the new MVI for the output data. Error messages can also be found with this output. If the user is supplying the arithmetic expression and it is syntactically an incorrect PL/1 expression, then the PL/1 optimizing compiler will print an error message and the program will not process the data. The compiler messages consist of a number of the form IELXXXXA and a message. These error messages can be found in the IBM publication called 'OS PL/1 Optimizing Compiler: Messages'.

As stated previously, errors may occur while evaluating the selected function. When the program detects an error, a warning message is printed. In the cases of division by zero, underflow, and overflow, the message will specifically state the error that was detected. Due to the numerous possible computational errors, the message provides a code, called an 'oncode', that indicates the precise computational error. A complete explanation of the 'oncodes' may be found in the IBM publication called 'OS and DOS PL/1 Language Reference Manual'.

After 10 errors of any type are detected, a message is printed and the run is aborted. In addition to the error message provided by this program, the operating system supplies an error message that can be found just prior to the program error message in the printout. The system error message may be helpful in understanding the error that has occurred. Explanations of these system error messages may be found in the IBM publication titled 'OS PL/1 Optimizing Compiler: Messages'.

The second output is a data set, that is usually written to disk, and contains the results of applying the requested function to the input data. The format of the output is the same as for the input. For the LOG and ANTILOG functions, if the parameter value is equal to the MVI, the log or antilog is not applied to that value. The value is output as the new MVI. For the MULTIPLY, DIVIDE, or LINEAR function, when a match of data is found, both of the values must not be equal to the MVI before the operation is performed. If one or both of the values are equal to the MVI, then no output record is generated for that block. This output can be used in three different ways. First, it can be the input to another Data-Manipulation Program run. Second, it can be a new parameter that is to be loaded into the data base. Third, the output can be the input to another application program.

Sample Procedure

The Data-Manipulation Program is run by executing a procedure called DTMATH. If the LOG or ANTILOG function is selected, only one data-base format input file is used and this file is defined by the parameter NAME1. If the LINEAR, MULTIPLY or DIVIDE function is selected, two data-base format input files are used and the file names are defined using the parameters NAME1 and NAME2. The output file name is defined using the parameter NAME3. To execute the Data-Manipulation Program, the following JCL should be used:

```
//. . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTMATH,NAME1=input1,NAME2=input2,NAME3=output
//GENFNCT.CMDIN  DD  *
  one or more records
//GO.LLLIMIT  DD  *
  latitude-longitude intersection
 //
```

The procedure used to execute the Data-Manipulation Program is as follows:

```
//DTMATH  PROC  TIM1=1,TIM2=1,TIM3=1,REG1=150K,REG2=350K,REG3=1100K,
//          PROG1=GENFNCT,PROG2=IEL0AA,PROG3=LOADER,UNIT1=3350,
//          VOL1=myvol,NAME1=NULLFILE,UNIT2=3350,VOL2=myvol,
//          NAME2=NULLFILE,UNIT3=3350,VOL3=myvol,NAME3=NULLFILE,
//          GOPARM='ISA(346K)'
//*****************************************************************************
//*   DTMATH: A PROCEDURE TO RUN THE DATA-MANIPULATION PROGRAM
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*  TIM1       TOTAL RUN TIME - STEP 1 (1)
//*  TIM2       TOTAL RUN TIME - STEP 2 (1)
//*  TIM3       TOTAL RUN TIME - STEP 3 (1)
//*  REG1       REGION SIZE - STEP 1 (150K)
//*  REG2       REGION SIZE - STEP 2 (350K)
//*  REG3       REGION SIZE - STEP 3 (1100K)
```

```
//*            CHANGE TO 3500K WHEN PROCESSING 2 SETS OF 6" DATA
//*  PROG1     LOAD MODULE NAME - STEP 1 (GENFNCT)
//*  PROG2     LOAD MODULE NAME - STEP 2 (IEL0AA)
//*  PROG3     LOAD MODULE NAME - STEP 3 (LOADER)
//*  UNIT1     FIRST DATA-BASE INPUT FILE DEVICE (3350)
//*  VOL1      FIRST DATA-BASE INPUT FILE VOLUME (myvol)
//*  NAME1     FIRST DATA-BASE INPUT FILE DSNAME (NULLFILE)
//*  UNIT2     SECOND DATA-BASE INPUT FILE DEVICE (3350)
//*  VOL2      SECOND DATA-BASE INPUT FILE VOLUME (myvol)
//*  NAME2     SECOND DATA-BASE INPUT FILE DSNAME (NULLFILE)
//*  UNIT3     OUTPUT FILE DEVICE (3350)
//*  VOL3      OUTPUT FILE VOLUME (myvol)
//*  NAME3     OUTPUT FILE DSNAME (NULLFILE)
//*
//*******************************************************************************
//*
//GENFNCT  EXEC  PGM=&PROG1,TIME=&TIM1,REGION=&REG1,PARM='ISA(4K)'
//*
//*  STEP 1: GENERATE THE SOURCE CODE FOR THE FUNCTION SUBPROGRAM
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//*
//*  DEFINE STANDARD DATA SETS
//*
//SYSPRINT DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//*
//*  DEFINE FILE CONTAINING PARTIAL SOURCE CODE FOR FUNCTION
//*
//DATAFCT  DD  DSN=sourcelib(COMPUTE),DISP=SHR
//*
//*  DEFINE TEMPORARY FILE FOR COMPLETE SOURCE CODE FOR FUNCTION
//*
//DATAOUT  DD  DSN=&&TEMP,UNIT=SYSDA,VOL=,DISP=(NEW,PASS),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3600),SPACE=(TRK,(1,1),RLSE)
//*
//*  DEFINE TEMPORARY FILE FOR INPUT TO STEP THREE
//*
//CMDOUT   DD  DSN=&&INPUT,UNIT=SYSDA,VOL=,DISP=(NEW,PASS),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3600),SPACE=(TRK,(1,1),RLSE)
//*
//*
//COMPILE  EXEC  PGM=&PROG2,TIME=&TIM2,REGION=&REG2,
//       PARM='NOOPTIONS,NOSOURCE,NOSTORAGE',COND=(9,LT,GENFNCT)
//*
//*  STEP 2: COMPILE THE FUNCTION SUBPROGRAM
//*
//STEPLIB  DD  DSN=SYS1.PLIX.LINKLIB,DISP=SHR
//SYSPRINT DD  SYSOUT=A
//SYSLIN   DD  DSN=&&LOADSET,DISP=(NEW,PASS),UNIT=SYSDA,
//   SPACE=(3120,(35,25),,,ROUND),
//   DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```

```
//SYSUT1  DD  DSN=&&SYSUT1,UNIT=VIO,SPACE=(CYL,(30,15)),
//    DCB=(BLKSIZE=1024,BUFNO=1)
//SYSIN  DD  DSN=&&TEMP,DISP=(OLD,DELETE)
//*
//*
//GO  EXEC  PGM=&PROG3,TIME=&TIM3,REGION=&REG3,PARM='/&GOPARM',
//    COND=((9,LT,GENFNCT),(9,LT,COMPILE))
//*
//*  STEP 3:  LOAD AND EXECUTE OF DATA-MANIPULATION PROGRAM
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//SYSLIB  DD  DSN=SYS1.PLIBASE,DISP=SHR
//        DD  DSN=mylib,DISP=SHR
//SYSLIN  DD  DSN=mylib(DTMATH),DISP=SHR
//        DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//SYSLOUT  DD  DUMMY
//SYSPRINT DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//*
//*  DEFINE DATA-BASE INPUT DATA SETS
//*
//DATAX1  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DSNAME=&NAME1,DISP=SHR
//DATAX2  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DSNAME=&NAME2,DISP=SHR
//*
//*  DEFINE OUTPUT DATA SET
//*
//DATAOUT  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DSNAME=&NAME3,
//             DISP=(NEW,KEEP,DELETE),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),
//             SPACE=(TRK,(5,5))
//*
//*  DEFINE TEMPORARY DATA SETS
//*
//TEMPIO  DD  DSN=&&TEMP,UNIT=SYSDA,
//            DCB=(RECFM=FB,LRECL=29,BLKSIZE=6206),SPACE=(TRK,(5,5))
//TEMPIO2  DD  DSN=&&TEMP2,UNIT=SYSDA,
//            DCB=(RECFM=FB,LRECL=52,BLKSIZE=6188),SPACE=(TRK,(5,5))
//CMDIN  DD  DSN=&&INPUT,DISP=(OLD,DELETE)
```

The Graphics Program

The Graphics Program produces contour maps and three-dimensional perspective drawings (3-D plots) of data. The program will process 10-minute, 1-minute, and 6-second data. The data are usually obtained using the Retrieval Program; the details of the retrieval can be found in the documentation for the Retrieval Program.

The contour maps are generated using Calcomp's General Purpose Contouring Program (GPCP) Version I. The program can plot data only, contours only, or both. The input includes map and plotter specifications and the interval between bold contours. The user specifies a title for the plot. Only one

map can be generated per program execution. The map consists of data points and (or) contours. The bold contours are labelled and the plot is surrounded by a border. The title is located outside the plot area along the positive Y axis.

The 3-D plots are generated using Calcomp's THREE-D Program Version 2. The user selects the size of the plot, the smoothness of the plot, the relative vertical exaggeration, and the position of the observer. The user can produce from one to nine plots of the same data from different locations. The position of the observer is specified by an azimuth angle, an elevation angle, and zoom factor. The zoom factor controls the distance between the observer and the surface. The plot is oriented so that the title is along the top.

The Graphics Program automatically saves the gridded data that are needed to produce the contour maps and the 3-D plots. Hence, in subsequent executions of the program, the user can use the saved data to produce additional plots without going through the process of regridding the data required for producing the plots. Even with this feature, the program can be very costly to use because of the extensive processing. The computer processing time increases as the number of values increase and doubling the number of values more than doubles the processing time.

Input

Input: Read from data set DATAIN. The first set of records is always required.

Record (1) - Execution Options

Col 1	blank	No map projection selected (using previously gridded data)
	1	Lambert map projection selected
	2	Albers map projection selected
	3	Polyconic map projection selected
Col 2	1	Produce a contour map or a data-point plot
	2	Produce 3-D plot(s)
	3	Produce both a contour map and 3-D plot(s)
Col 3	1	Generate data and plot(s)
	2	Generate plot(s) using previously generated data

The second record depends on the value placed in column 3 of record 1. If generating data and plot(s):

Record (2) - Definition of plotting area in geographic coordinates

Col 1-6	Minimum latitude of plotting area (DDMMSS)
Col 7-12	Maximum latitude of plotting area (DDMMSS)
Col 13-19	Minimum longitude of plotting area (DDDMMSS)

Col 20-26	Maximum longitude of plotting area (DDDMMSS)
Col 27-32	Size of buffer area around plot (DDMMSS). Useful for proper overlapping of contour maps.

If generating plots using previously generated data:

Record (2) - Definition of plotting area in cartesian coordinates

Col 1-10	Minimum X coordinate of plotting area (F10.4)
Col 11-20	Maximum X coordinate of plotting area (F10.4)
Col 21-30	Minimum Y coordinate of plotting area (F10.4)
Col 31-40	Maximum Y coordinate of plotting area (F10.4)
Col 41-52	Minimum Z value (E12.6)
Col 53-64	Maximum Z value (E12.6)

Note: Values for the above 6 quantities can be found as part of the print-out of a previous execution of the Graphics Program.

If generating data and plot(s), a third record is used:

Record (3) - Scale of map

Col 1-9	Denominator of map scale (e.g. 250000 for 1:250,000 map) (I9)

The second set of records are required only for contour plots.

Record (1) - Plot options and title

Col 1	1	Plot data points only
	2	Plot contours only
	3	Plot data points and contours
Col 2		Blank
Col 3-77		Title of plot (left-justified)

Record (2) - Plot specifications

Col 1-5	Rotation angle (decimal degrees). Measured counterclockwise from east. One decimal digit of precision allowed (F5.0)
Col 6-7	Blank
Col 8-10	Width of plot, in inches (I3)
Col 11-13	Blank
Col 14-25	Interval between bold contours (E12.6)

The third set of records is required only for 3-D plots.

Record (1) - 3-D program execution options

Col 1	1	Coarse gridding (no smoothing)
	2	Medium gridding (some smoothing)
	3	Fine gridding (most smoothing)

Col 2	1	Normal vertical exaggeration
	2	Less than normal vertical exaggeration
	3	Minimum vertical exaggeration
	4	More than normal vertical exaggeration
	5	Maximum vertical exaggeration
Col 3	1-9	Number of 3-D plots

Record (2) - Plot specifications (one for each 3-D plot)

Col 1-5	Azimuth angle of observer (decimal degrees), measured counterclockwise from east. A value of 0 indicates viewing from due east, a value of 90 indicates viewing from due north, and so forth. One decimal digit of precision allowed (F5.0)
Col 6-10	Elevation angle of observer (decimal degrees), measured counterclockwise from directly above the plot. A value of 0 indicates viewing from directly above the plot and a value of 90 indicates viewing from the same level as the plot. One decimal digit of precision allowed (F5.0)
Col 11-13	Blank
Col 14-15	Zoom factor (I2). Varies from 0 to 10 with 0 placing the observer at the plot and 10 placing the observer very far from the plot.
Col 16	Blank
Col 17-20	Length of side of plotting area, in inches. One decimal digit of precision allowed (F4.0)
Col 21-70	Title of plot (left-justified)

Output

The first output is a standard print file containing basic information about the execution of the Graphics Program. This file contains: (1) Selected execution options; (2) basic information on data-base parameter being plotted; (3) if generating data, the calculated minimum and maximum X, Y, and Z values; (4) if producing a contour map, the contour plot specifications, the contour interval, and the value of the first contour plotted; and (5) if producing 3-D plots, the specifications for each requested plot. If any errors occur, the error message can be found in this file.

The second output is a punch file containing the Calcomp plotter commands. Additional output is generated by Calcomp's GPCP and THREE-D programs. GPCP prints one line for each value showing the value and its corresponding plotter coordinates. The THREE-D program prints the Z matrix. The output generated by these two programs can be extensive. There is no practical way to eliminate this information.

Sample Procedure

A procedure called DTGRAPH is used to execute the Graphics Program. To run the Graphics Program, several variations of JCL should be used. If generating data and plotting contour maps and (or) 3-D plots, the following JCL should be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTGRAPH,NAME1=databasedata,
//     NAME2=data,NAME3=savefile1,NAME4=savefile2
//GRAPH.DATAIN  DD  *
   input records
//
```

If generating contour maps using previously generated data:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTGRAPH,NAME2=data,DSP2=OLD,
//     NAME3=savefile1,DSP3=OLD
//GRAPH.DATAIN  DD  *
   input records
//
```

If generating 3-D plots using previously generated data:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTGRAPH,NAME4=savefile2,DSP4=OLD
//GRAPH.DATAIN  DD  *
   input records
//
```

If generating both contour map and 3-D plots using previously generated data:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DTGRAPH,NAME2=data,DSP2=OLD,
//     NAME3=savefile1,DSP3=OLD,NAME4=savefile2,
//     DSP4=OLD
//GRAPH.DATAIN  DD  *
  input records
//
```

In the above examples, *databasedata* is the data-set name of the file containing the retrieved data, *data* is the data-set name of the file containing the transformed data-base data, *savefile1* is the data-set name of the file where the data needed to produce contour maps are saved, and *savefile2* is the data-set name of the file where the data needed to produce 3-D plots are saved.

The procedure used to execute the Graphics Program is as follows:

```
//DTGRAPH  PROC  PROG1=DTGRAPH,PROG2=GPCPGRID,PROG3=MDFILE,
//         PROG4=THREE906,TIME1=1,TIME2=1,TIME3=1,TIME4=1,
//         REG1=2200K,REG2=450K,REG3=110K,REG4=250K,
//         NAME1=NULLFILE,UNIT1=3350,VOL1=myvol,
//         NAME2=NULLFILE,UNIT2=3350,VOL2=myvol,
//         DSP2='(NEW,KEEP,DELETE)',NAME3=NULLFILE,UNIT3=3350,
//         VOL3=myvol,DSP3='(NEW,KEEP,DELETE)',NAME4=NULLFILE,
//         UNIT4=3350,VOL4=myvol,DSP4='(NEW,KEEP,DELETE)'
//*******************************************************************************
//*
//*   DTGRAPH: A PROCEDURE TO RUN THE GRAPHICS PROGRAM
//*            PROCEDURE DIVIDED INTO FOUR STEPS
//*            STEP 1 (GRAPH): TRANSFORMS DATA-BASE INPUT TO (X,Y,Z)
//*                 AND GENERATES CONTROL CARDS FOR CALCOMP'S GPCP AND
//*                 THREE-D PROGRAMS.
//*            STEP 2 (GPCP) EXECUTES CALCOMP'S GPCP PROGRAM
//*            STEP 3 (MDFILE) MODIFIES FILE CONTAINING GRIDDED DATA,
//*                 THAT WAS PRODUCED BY GPCP USING THE 'PNCH' COMMAND,
//*                 FOR INPUT TO THE THREE-D PROGRAM.
//*            STEP 4 (THREED) EXECUTES CALCOMP'S THREE-D PROGRAM
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS):
//*
//*       PROG1      LOAD MODULE NAME FOR STEP 1 (DTGRAPH)
//*       PROG2      LOAD MODULE NAME FOR STEP 2 (GPCPGRID)
//*                  DEFAULT MODULE PRODUCES CONTOUR PLOTS FOR CALCOMP
//*                  PLOTTERS WITH 906 TYPE CONTROLLERS
//*       PROG3      LOAD MODULE NAME FOR STEP 3 (MDFILE)
//*       PROG4      LOAD MODULE NAME FOR STEP 4 (THREE906)
//*                  DEFAULT MODULE PRODUCES 3-D PLOTS FOR CALCOMP
//*                  PLOTTERS WITH 906 TYPE CONTROLLERS
//*       TIME1      TOTAL RUN TIME FOR STEP 1 (1)
//*       TIME2      TOTAL RUN TIME FOR STEP 2 (1)
//*       TIME3      TOTAL RUN TIME FOR STEP 3 (1)
//*       TIME4      TOTAL RUN TIME FOR STEP 4 (1)
//*       REG1       REGION SIZE FOR STEP 1 (2200K)
//*       REG2       REGION SIZE FOR STEP 2 (450K)
//*       REG3       REGION SIZE FOR STEP 3 (110K)
//*       REG4       REGION SIZE FOR STEP 4 (250K)
//*       NAME1      DATA-BASE DATA INPUT FILE DSNAME (NULLFILE)
//*       UNIT1      DATA-BASE DATA INPUT FILE UNIT (3350)
//*       VOL1       DATA-BASE DATA INPUT FILE VOLUME (myvol)
//*       NAME2      CONTROL-POINT DATA FILE DSNAME (NULLFILE)
//*       UNIT2      CONTROL-POINT DATA FILE UNIT (3350)
//*       VOL2       CONTROL-POINT DATA FILE VOLUME (myvol)
//*       DSP2       CONTROL-POINT DATA FILE DISPOSITION
//*                  ((NEW,KEEP,DELETE))
//*       NAME3      GPCP SAVE FILE DSNAME (NULLFILE)
//*       UNIT3      GPCP SAVE FILE UNIT (3350)
//*       VOL3       GPCP SAVE FILE VOLUME (myvol)
//*       DSP3       GPCP SAVE FILE DISPOSITION ((NEW,KEEP,DELETE))
```

```
//*       NAME4      GPCP PNCH FILE DSNAME (NULLFILE)
//*       UNIT4      GPCP PNCH FILE UNIT (3350)
//*       VOL4       GPCP PNCH FILE VOLUME (myvol)
//*       DSP4       GPCP PNCH FILE DISPOSITION ((NEW,KEEP,DELETE))
//*
//*********************************************************************
//*
//GRAPH  EXEC  PGM=&PROG1,TIME=&TIME1,REGION=&REG1,PARM='ISA(1864K)'
//*
//*         STEP 1: DATA TRANSFORMATION AND CONTROL CARD GENERATION
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//*
//*   DEFINE STANDARD I/O DATA SETS
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//FT06F001  DD  SYSOUT=A
//*
//*  DEFINE DATA-BASE DATA INPUT FILE
//*
//DBINPT  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*
//*  DEFINE DATA SET FOR TRANSFORMED DATA (CONTROL-POINT FILE)
//*
//DBOTPT  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=&DSP2,DSN=&NAME2,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=12960),SPACE=(TRK,(3,25),RLSE)
//*
//*  DEFINE TEMPORARY FILES FOR CONTROL CARDS
//*
//CNTCRD1  DD  UNIT=SYSDK,VOL=,DISP=(NEW,PASS),DSN=&&CARD1,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3600),SPACE=(TRK,(1,1),RLSE)
//CNTCRD2  DD  UNIT=SYSDK,VOL=,DISP=(NEW,PASS),DSN=&&CARD2,
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3600),SPACE=(TRK,(1,1),RLSE)
//*
//*
//GPCP  EXEC  PGM=&PROG2,TIME=&TIME2,REGION=&REG2,
//            COND=((333,EQ,GRAPH),(776,LT,GRAPH))
//*
//*     STEP 2: CALCOMP'S GPCP PROGRAM - VERSION 1
//*
//STEPLIB  DD  DSN=SYS1.CALCOMP,DISP=SHR
//         DD  DSN=SYS1.MVT.LINKLIB,DISP=SHR
//*  DEFINE TEMPORARY WORK FILE
//FT01F001  DD  UNIT=SYSDK,VOL=,DISP=NEW,DSN=&&TEMP1,DCB=BLKSIZE=9444,
//          SPACE=(TRK,(5,2),RLSE)
//*
//*  DEFINE CONTROL-POINT FILE - TRANSFORMED DATA FILE FROM STEP 1
//*
//FT03F001  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(OLD,KEEP),DSN=&NAME2
//*
```

```
//*  DEFINE SAVE FILE
//*
//FT04F001  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DISP=&DSP3,DSN=&NAME3,
//          DCB=(RECFM=VBS,BLKSIZE=6447),SPACE=(TRK,(5,5),RLSE)
//*  DEFINE CONTROL CARD FILE - SAME AS FILE CNTCRD1 FORM STEP 1
//FT05F001  DD  UNIT=SYSDK,VOL=,DISP=(OLD,DELETE),DSN=&&CARD1
//*  DEFINE STANDARD PRINT FILE
//FT06F001  DD  SYSOUT=A
//*
//*  DEFINE PNCH FILE
//*
//FT07F001  DD  UNIT=&UNIT4,VOL=SER=&VOL4,DISP=&DSP4,DSNAME=&NAME4,
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160),SPACE=(TRK,(5,5),RLSE)
//*
//*  DEFINE PLOTTER COMMAND FILE
//*
//FT09F001  DD  SYSOUT=B
//*
//*
//MDFILE  EXEC  PGM=&PROG3,TIME=&TIME3,REGION=&REG3,PARM='ISA(4K)',
//      COND=((111,EQ,GRAPH),(776,LT,GRAPH))
//*
//*      STEP 3: MODIFICATION OF PNCH FILE, PRODUCED BY STEP 2, FOR USE
//*              BY STEP 4
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,DISP=SHR
//*
//*  DEFINE STANDARD DATA SETS
//*
//SYSPRINT  DD  SYSOUT=A
//PLIDUMP  DD  DUMMY
//*
//*  DEFINE INPUT FILE - PNCH FILE FROM STEP 2
//*
//DATAIN  DD  UNIT=&UNIT4,VOL=SER=&VOL4,DISP=(OLD,KEEP),DSNAME=&NAME4
//*  DEFINE A TEMPORARY DATA SET
//TEMP  DD  UNIT=SYSDK,VOL=,DISP=(NEW,PASS),DSN=&&TEMP,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=9440),SPACE=(TRK,(5,2),RLSE)
//*
//THREED  EXEC  PGM=&PROG4,TIME=&TIME4,REGION=&REG4,
//     COND=((111,EQ,GRAPH),(776,LT,GRAPH))
//*
//*      STEP 4: CALCOMP'S THREE-D PROGRAM
//*
//STEPLIB  DD  DSN=SYS1.CALCOMP,DISP=SHR
//         DD  DSN=SYS1.MVT.LINKLIB,DISP=SHR
//*  DEFINE TEMPORARY WORK FILES
//FT01F001  DD  UNIT=SYSDK,VOL=,DISP=NEW,DSN=&&TEMP1,
//          DCB=BLKSIZE=9444,SPACE=(TRK,(5,2),RLSE)
//FT04F001  DD  UNIT=SYSDK,VOL=,DISP=NEW,DSN=&&TEMP2,
```

```
//          DCB=BLKSIZE=9444,SPACE=(TRK,(5,2),RLSE)
//*  DEFINE GRIDDED DATA INPUT FILE - TEMP FILE FROM STEP 3
//*
//FT03F001  DD  UNIT=SYSDK,VOL=,DISP=(OLD,DELETE),DSN=&&TEMP
//*
//*
//*  DEFINE CONTROL CARD FILE - FILE CNTCRD2 FROM STEP 1
//FT05F001  DD  UNIT=SYSDK,VOL=,DISP=(OLD,DELETE),DSN=&&CARD2
//*  DEFINE STANDARD OUTPUT FILES
//FT06F001  DD  SYSOUT=A
//FT07F001  DD  SYSOUT=B
//*  DEFINE PLOTTER COMMAND FILE
//FT09F001  DD  SYSOUT=B
```

SUPPLEMENT III: PROGRAM-MAINTENANCE MANUAL

The purpose of this section is to highlight the changes needed to implement the DMS for another study. It is not a complete reference for debugging any possible future problems with the DMS. The information provided is based upon several assumptions concerning the environment in which the DMS is to be implemented. The assumptions are:

1. Software is implemented by using an IBM or IBM-compatible mainframe;
2. Software is implemented by using the System 2000 DBMS and the same date-base definition; and
3. The computer system has the Fortran IV and PL/1 programming languages.

These criteria narrow the type of implementation for which information is provided. It would not be practical to give information pertaining to the implementation of this system using a different DBMS. There are several available packages that could provide the resources of System 2000. However, regardless of DBMS and programming languages chosen, the logic behind the present software would still be useful. If the system is used in a different environment, it is advisable to retain the same input and output formats. This would ease the implementation of the data-generation and application programs.

A section is provided for each program. Each section begins with an introduction and then lists changes needed for each subroutine in the program requiring changes. In most instances, needed changes are to the initial values of variables and to array dimensions. For example, some initialized variables reflect the location of the High Plains aquifer. These need to be changed to reflect the location of the area to be studied. Changes in the program logic are not likely to be necessary. Major coding changes are required only for the Edit and Retrieval Programs. These changes will be discussed in the section for each of these programs.

Edit Program

The Edit Program provides a means of checking correctness of the input to other programs in the DMS. Because of this extensive purpose, the Edit Program has become very large. This program will require more implementation changes than any other program in the system.

Because the program is written in IBM Fortran, which does not provide a means of encoding and decoding data, an unsupported and undocumented subroutine, called CORE, is used to provide this function. It is thus advisable to rewrite the code that performs the decoding and encoding. This can be easily done if the available version of Fortran has an encode/decode feature. If not, the Edit Program would have to be rewritten in a language that provides this feature--the use of CORE is very prevalent within this program.

There are three versions of the Edit Program. Version 1, called DTEDIT1, is used to check the input to the Load Program. Version 2, called DTEDIT2, is used to check the input to the Retrieval Program. Version 3, called DTEDIT3, is used to check the input to the Instant-Update Program. All three versions of the Edit Program are identically structured. Each program has a block-data subroutine, a program-initialization subroutine, and one subroutine to guide the checking of the records. Each program has only one subroutine that accesses the data base. The following subroutines will require changes. After each subroutine name, a name is given in parentheses that identifies the version of the Edit Program to which the subroutine belongs.

BDAT (DTEDIT1)

This is a block-data subroutine used to set up common areas and initialize variables. The LEVEL1 array is used for duplication checking. It covers the entire United States by 1-degree by 1-degree blocks. These dimensions could be changed to cover a different sized area such as part of the United States. If the area covered by this array is changed, initial values of the variables LTBASE and LNBASE would need to be changed. These variables are the base values for latitude and longitude and they are used to compute positions within the LEVEL1 array.

INIT (All 3 versions)

Two system subroutines are used to determine the date and time of the run. These subroutines may be different on another system. No other changes are required.

CHK100 (DTEDIT1)

The variables LATMIN, LATMAX, LONMIN, and LONMAX are used to describe the extent of the study area in terms of latitude and longitude. These variables are initialized to appropriate values for the High Plains aquifer. These values would have to be changed for another study.

DUPL (DTEDIT1)

This subroutine uses the LEVEL1 array which is used for duplication checking of data prior to loading into the data base. The LEVEL1 array is initially declared in the BDAT block-data subroutine and the changes to this array are discussed in the section on that subroutine.

DTSTK1 (DTEDIT1)

This is the only subroutine within the DTEDIT1 program that accesses the data base. Hence, it is the only subroutine that contains System 2000 (the DBMS) PLEX commands. The data-base name that is used in some of the PLEX statements should be changed. This subroutine uses the LEVEL1 array that is initially declared in the BDAT block-data subroutine and the changes to this array are discussed in the section on that subroutine.

LTCHK2 (DTEDIT2)

This subroutine checks to see if the latitude on a 006# record falls within the study area. The latitude extremes are now set to values appropriate to the High Plains aquifer. These values need to be changed to reflect the area under study.

LNCHK2 (DTEDIT2)

This subroutine checks to see if the longitude on a 006# record falls within the study area. The longitude extremes are now set to values appropriate to the High Plains aquifer. These values need to be changed to reflect the area under study.

DTSTK2 (DTEDIT2)

This is the only subroutine within the DTEDIT2 program that accesses the data base. Hence, it is the only subroutine that contains System 2000 (the DBMS) PLEX commands. The data-base name that is used in some of the PLEX statements should be changed.

BDAT (DTEDIT3)

This is the block-data subroutine used to set up common areas and initialize variables. The VALNAM array is used to store the names of authorized users of the Instant-Update program; this array will presently store 20 names. This may need to be changed, depending on the number of authorized users.

NAMECK (DTEDIT3)

This subroutine uses the VALNAM array to store the names of authorized users of the Instant-Update Program; the dimension may be changed to accommodate more names.

LTCHK (DTEDIT3)

This subroutine determines if the latitude on a 600# record falls within the study area. The constants defining the latitude extremes are now set to values that are appropriate for the High Plains aquifer. These values need to be changed to reflect the area under study.

LNCHK (DTEDIT3)

This subroutine determines if the longitude on a 600# record falls within the study area. The constants defining the longitude extremes are now set to values that are appropriate for the High Plains aquifer. These values need to be changed to reflect the area under study.

DTSTK3 (DTEDIT3)

This is the only subroutine within the DTEDIT3 program that accesses the data base. Hence, it is the only subroutine that contains System 2000 (the DBMS) PLEX commands. The data-base name that is used in some of the PLEX statements should be changed.

Load Program

The Load Program consists of a main procedure, containing internal subroutine-procedures, and three external subroutine-procedures. The program loads 1-degree, 10-minute, 1-minute, and 6-second data; because of its modular design, it can be easily expanded to accommodate other densities. One of the only limitations on the program is that it expects to load entire 1-degree by 1-degree blocks of data at one time. The main procedure and its internal subroutine-procedures will be discussed as one unit under the heading of DTLOAD; the external subroutine-procedures will be discussed separately.

DTLOAD

The main portion of the program should work essentially unchanged. The data-base name used in the System 2000 (the DBMS) PLEX commands should be changed.

An array, called GRAPH, is declared in the main procedure. It is used to produce graphics output for the report that is produced by each execution of the program. It is dimensioned to reflect the latitude and longitude extremes of the High Plains aquifer and has a position for each 1-degree by 1-degree block within and on the boundary of the aquifer. As a 1-degree by 1-degree block is loaded in the data base, an asterisk is placed in the GRAPH array position corresponding to the 1-degree by 1-degree block. Dimensions of this array would have to be changed for another study.

INITGR

This procedure initializes positions within the GRAPH array that correspond to the boundary of the High Plains aquifer. The positions are set to the letter 'O'. This procedure would have to be completely changed for a different study.

REPORT

This procedure outputs several lines of print that use the name of the study unit; this would have to be changed for another study. The main output are two figures which resemble the shape of the High Plains aquifer. The first figure is produced using an array called EX-GRAPH which is dimensioned exactly like GRAPH. It is initialized with the letters 'O', 'P', and 'C'. The 'O' means that the 1-degree by 1-degree block corresponding to that position is outside of the study area and the 'P' means that the block is partially within the study area. The second figure is produced with the GRAPH array. Each figure is surrounded by geographic coordinates that will have to be changed for another study.

Retrieval Program

The Retrieval Program is written in IBM Fortran. As with the Edit Program, the Retrieval Program requires a method of encoding and decoding data. Because IBM Fortran does not have this feature, a routine called CORE is used to provide this function. The sections of the code that use the CORE subroutine should be changed. The program can remain written in Fortran, if the available version has an encode/decode feature. If this feature is not available, the program should be rewritten in a language that can provide the feature. The program retrieves 1-degree, 10-minute, 1-minute, and 6-second data. It can be expanded for other densities if desired.

MAIN

A password is required to open the data base. The password is initialized within the program. This password should be changed and read from the input. The data-base name used in some of the System 2000 (the DBMS) PLEX commands should be changed. The program uses system subroutines to compute date and time of the run; these subroutines may be different on another system. An array, called DENTAB, relates the data density to the base level of the data. It also relates the data density to the output record types associated with the base level. This array would have to be modified if more data densities are built into the data-base structure.

General-Update Program

The General-Update Program consists of a main procedure and eight external subroutine-procedures. The program processes updates for 1-degree, 10-minute, 1-minute, and 6-second data. Because of it modular design, it can be expanded for more data densities. The program should work with little or no modification. In all cases, the main procedure and the subroutine procedures use a data-base name in some of the System 2000 (the DBMS) PLEX commands. This data-base name should be changed.

The program allows the user, usually the DBA, to change an existing value to the MVI or to change an MVI-valued block to another value. This rule can be changed if necessary.

Instant-Update Program

The Instant-Update Program consists of a main procedure and two external subroutine-procedures. The program processes proposal records for 10-minute, 1-minute, and 6-second data. Because of its modular design, it can be easily expanded for more densities. The program should work with little or no modification.

There are several rules about adding test proposals that are incorporated into the subroutine procedures UPDATE1 and UPDATE2. First, a test proposal cannot be added for any block that already has five existing test proposals; this number can be changed. Second, a test proposal cannot be added for a

block in which the permanent stored-value is the MVI. Conversely, the MVI cannot be proposed as a test value. These rules were adapted for the High Plains study; they are subject to change and the procedures easily can be modified to incorporate any changes to these rules.

DTIUPD

A password is required to open the data base. This password is presently initialized and should be changed. The data-base name used in some of the System 2000 (the DBMS) PLEX commands should be changed.

UPDATE1

The data-base name used in some of the System 2000 PLEX commands should be changed.

UPDATE2

The data-base name used in some of the System 2000 PLEX commands should be changed.

Move Program

The Move Program consists of a main procedure and six external subroutine-procedures. It manipulates the proposal records for 10-minute, 1-minute, and 6-second data. Because of its modular design, it easily can be expanded if more densities are required. In the three subroutine procedures that actually process the proposals (MOVE2, MOVE3, and MOVE4), a rule is implemented that states that any test proposal over 2 months old is removed from the data base; this value of 2 months can be changed. The data-base name used in some of the System 2000 (the DBMS) PLEX commands should be changed.

Statistics Program

This program computes statistics for 1-degree, 10-minute, 1-minute, and 6-second data. Because of its modular design, the program can be easily expanded to compute statistics for other data densities. The data-base name used in the System 2000 (the DBMS) PLEX commands should be changed. No other modifications are required.

Reload Program

The Reload Program consists of a main procedure and three external subroutine-procedures. The program processes 10-minute, 1-minute, and 6-second data. Because of its modular design, it can be easily expanded for more data densities. The program should work with little or no modification. There are two data-base names used in some of the System 2000 (the DBMS) PLEX commands. These names should be changed.

Data-Transformation Program

The Data-Transformation Program is two separate computer programs; the first program is called DTDTP and the second is called DTINTERP.

DTDTP

The DTDTP program consists of a main procedure, called DTDTP, and 18 subroutine procedures. The function of this program is to transform data from the data base, which is in the form of (longitude, latitude, value) triplets, into (X, Y, value) triplets. The program transforms 10-minute, 1-minute, and 6-second data. The program can be expanded to accommodate other data densities. Two procedures within the DTDTP program require changes.

PREP4

The array LONDIS is used to estimate the number of 6-second values within a model area. This array is dimensioned to reflect latitude extremes of the High Plains aquifer. The array contains the distance (in miles) between degrees of longitude as a function of latitude. This array will have to be changed for a different study area. The variable MAXPOINTS is initialized to 1600 and it represents the maximum number of 6-second values within a model area for which trend-surface analysis is a useful technique for determining the model matrix. The initial value of this variable can be changed.

CORE

The array LONDIS is used in this subroutine procedure. It is the same array as used by the PREP4 subroutine procedure. The same comments apply to LONDIS as discussed for PREP4.

DTINTERP

The DTINTERP program consists of a main procedure and 5 subroutine procedures. This program takes the transformed data from the first step and generates the model matrix. There are three techniques available to generate the matrix; because of its modular design, more techniques can be added if desired. Only one of the procedures may require changes.

TREND

The order of the polynomial used to compute the trend surface cannot exceed 4. This procedure would have to be rewritten to allow larger order polynomials.

Data-Manipulation Program

The Data-Manipulation Program contains a pre-processor program called GENFNCT. This program generates the source code for a function subroutine that is used in the Data-Manipulation program to perform the mathematical manipulations of the data-base formatted data. The function subroutine is

called COMPUTE and its partial source code is stored in a disk file which is read by the GENFNCT program. After the source code for COMPUTE is completed, the function subroutine is compiled and linked with the remainder of the Data-Manipulation Program.

The Data-Manipulation Program consists of a main procedure, 6 subroutine procedures, and 1 function subroutine (COMPUTE). The program manipulates 10-minute, 1-minute, and 6-second data. Because of its modular design, it can be easily expanded to accommodate other data densities. The program should work with little or no modifications.

Graphics Program

The Graphics Program consists of a main procedure and 11 subroutine procedures. Although this program is written in PL/1, it calls 3 Fortran subroutines for converting (longitude, latitude, value) triplets to (X, Y, value) triplets using one of three map projections. This program is linked to Calcomp software for producing contour maps and 3-D perspective plots. If Calcomp software is not available, then two of the subroutine procedures will have to be completely rewritten. The subroutine procedure GPCP1 generates the commands for executing Calcomp's General Purpose Contouring Program Version 1. The subroutine procedure THREED, generates the commands to execute Calcomps THREED program Version 2. The Graphics Program will process 10-minute, 1-minute, and 6-second data. Because of its modular design, it can be easily expanded to accommodate other data densities.

SUPPLEMENT IV: COMPUTER SOFTWARE TAPE

The computer software tape consists of 25 files of programs and supporting files that are described and documented in this report. Each computer program was placed in a separate file; the supporting files were combined into related groups.

The magnetic tape is stored at the U.S. Geological Survey's National Computer Center, Reston, Va.; additional information and copies of individual files or the entire tape may be obtained from: Office of the Chief Hydrologist, Water Resources Division, U.S. Geological Survey, Mail Stop 409, National Center, 12201 Sunrise Valley Drive, Reston, VA 22092. The magnetic tape can be identified by the number 224161 which is a number assigned by the tape librarian for the Information Systems Division in Reston, Va.

Physical Characteristics of the Magnetic Tape

The physical characteristics of the magnetic tape were chosen to make the tape compatible with a wide variety of computer systems. For example, the density and block-size values are small so that the tape can be read using a minicomputer or a microcomputer.

The magnetic tape has the following physical characteristics:

Tracks: 9
Density: 1600 BPI (bits per inch)
Labels: None
Record Length (Fixed): 80 bytes
Block Size: 2,000 bytes
Code: ASCII

The magnetic tape was processed on an Amdahl 470V/7 computer located at the U.S. Geological Survey's National Computer Center in Reston, Va. Each computer program or supporting file was initially stored as a sequential file or member of a partitioned-data set on a disk-storage device. The files were copied to the magnetic tape using an IBM utility program called IEBGENER.

Tape Contents

The files that are stored on magnetic-tape number 224161 can be divided into four categories: (1) Source code for computer programs, (2) IBM Job Control Language (JCL) for executing the programs, (3) files containing variable declarations used by some of the computer programs, and (4) miscellaneous information. Listed in table 2 are the files in the order in which they were stored on the magnetic tape.

Table 2.--*Data-base programs and supporting files*

File number	Contents
1	Edit Program for Load-Program input
2	Edit Program for Retrieval-Program input
3	Edit Program for Instant-Update Program input
4	Load Program
5	Load-Program subroutines
6	Retrieval Program
7	Retrieval-Program subroutines
8	Instant-Update Program
9	Instant-Update Program subroutines
10	Move Program
11	Statistics Program
12	General-Update Program
13	Reload Program
14	Data-Transformation Program
15	Data-Transformation Program subroutines
16	Interpolation Program
17	Data-Manipulation Program
18	Data-Manipulation Program subroutines and supporting programs
19	Graphics Program
20	Graphics Program subroutines and supporting programs
21	IBM procedures for programs that access the data base and the DBMS
22	IBM procedures for the data-application programs
23	Variable declarations for Fortran programs that access the data base and the DBMS
24	Variable declarations for PL/1 programs that access the data base and the DBMS
25	Miscellaneous

The computer programs (files 1 through 20 in table 2) can be divided into two categories: (1) Programs that interact with the data base and the DBMS, and (2) data-application programs. These programs were written in either Fortran IV or PL/1.

Files 1 through 13 on magnetic-tape number 224161 contain the programs that interact with the data base and the DBMS. These programs were linked to System 2000 (the DBMS) through subroutines provided with System 2000. The Edit and Retrieval Programs were written in Fortran IV. All other programs in this group were written in PL/1.

Files 14 through 20 on magnetic-tape number 224161 contain the programs that use data retrieved from the data base. All of these programs were written in PL/1.

Files 21 and 22 on magnetic-tape number 224161 contain the procedures, written in IBM JCL, to execute the computer programs. These procedures were written to be used on the Amdahl 470V/7 computer in Reston, Va., and make reference to files that are stored on a 3350-type disk-storage device. File 21 contains the procedures for executing the programs that access the data base and the DBMS. File 22 contains the procedures for executing the data-application programs.

Files 23 and 24 on magnetic-tape number 224161 contain the variable declarations needed to link the programs in files 1 through 13 with the System 2000 DBMS.

File 25 on magnetic-tape number 224161 contains the following miscellaneous information: (1) A list of parameter names, (2) the data-base definition in the form required by System 2000 (the DBMS), (3) the list of error messages used by the Edit and Retrieval Programs, and (4) several short files containing information needed by the IBM procedures for the programs that access the data base and the DBMS. Most of the information in this file would be needed only if the System 2000 DBMS was used to manage the data base.

*U.S. GOVERNMENT PRINTING OFFICE: 1986-0-672-826/45298