UNITED STATES
DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

A DATA-MANAGEMENT SYSTEM FOR AREAL INTERPRETIVE DATA FOR THE HIGH PLAINS IN PARTS OF COLORADO, KANSAS, NEBRASKA, NEW MEXICO, OKLAHOMA, SOUTH DAKOTA, TEXAS, AND WYOMING

By Richard R. Luckey and Carmelo F. Ferrigno

U.S. GEOLOGICAL SURVEY

Water-Resources Investigations 82-4072

December 1982

P. III FOLLOWS

UNITED STATES DEPARTMENT OF THE INTERIOR

JAMES G. WATT, Secretary

GEOLOGICAL SURVEY

Dallas L. Peck, Director

For additional information write:

Project Chief, High Plains RASA
U.S. Geological Survey
Water Resources Division
Mail Stop 412, Box 25046
Denver Federal Center
Denver, Colorado 80225

For purchase write to:

Open-File Services Section
Western Distribution Branch
U.S. Geological Survey
Box 25425
Denver Federal Center
Denver, Colorado 80225
(303) 234-5888

CONTENTS

CONTENTS--Continued

CONTENTS--Continued

ILLUSTRATIONS

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
acre	4047	square meters
feet (ft)	0.3048	meter
feet per day (ft/d)	0.3048	meters/day
inch (in.)	25.40	millimeter
mile (mi)	1.609	kilometer
square mile (mi^2)	2.590	square kilometer
mile per inch (mi/in.)	6.334×10^{-2}	kilometer per millimeter

A DATA-MANAGEMENT SYSTEM FOR AREAL INTERPRETIVE DATA FOR THE HIGH PLAINS IN PARTS OF COLORADO, KANSAS, NEBRASKA, NEW MEXICO, OKLAHOMA, SOUTH DAKOTA, TEXAS, AND WYOMING

By Richard R. Luckey and Carmelo F. Ferrigno

ABSTRACT

The High Plains Regional Aquifer System Analysis study has developed a regional water-resources (and related) data storage and retrieval system to organize and preserve areal interpretive data. The system is general and can easily be adapted for other studies. This report documents the High Plains data base as well as the general system that is independent of the High Plains area.

The system is built around a hierarchically structured data base consisting of related latitude-longitude blocks. Various parameters in the data base are stored at different degrees of detail, with the finest detail being a block of 1 minute of latitude by 1 minute of longitude (approximately 1 square mile).

Five general classes of data have been stored in the High Plains data base: (1) Aquifer-geometry data; (2) aquifer and water characteristics; (3) water levels; (4) climatological data; and (5) land- and water-use data. These data were compiled from maps, using both automated and manual data-generation procedures. The data-compilation procedures and computer programs are documented in this report, including information on 44 parameters that have been stored in the data base, which represent more than 10.9 million characters of information. Supplemental sections of the report contain user documentation of the data-system software and the changes that would need to be made to use this system for other studies.

INTRODUCTION

The U.S. Geological Survey began a series of hydrologic investigations in 1978, the Regional Aquifer System Analysis (RASA) programs, to provide water-management information on a regional scale. This program represents a systematic effort to study regional ground-water systems throughout much of the country and represent a significant part of the national water supply. The objective of the program is to assemble hydrologic information to create predictive capabilities for effective management of the Nation's ground water.

The general purpose of the 5-year High Plains RASA study that began in 1978 is to provide: (1) Hydrologic information needed to evaluate the effects of continued ground-water development; and (2) computer models to predict aquifer response to changes in ground-water development (Weeks, 1978). To meet these objectives, a regional water-resources (and related) data storage and retrieval system was developed. The purpose of this Data-Management System (DMS) is to organize the large volumes of data needed to develop both a regional finite-difference model of ground-water flow, and local models in selected areas. It is intended that the system and the data be maintained beyond the duration of the High Plains RASA for future water-resources management.

While only data for the High Plains aquifer (fig. 1) have been stored in the data base, 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 block of latitude and longitude. These records are then related or grouped in a heirarchical, or inverted tree, structure. A hierarchical data-base structure means that all data blocks are related to one and only one block at each higher level, and there is one specially designed block, called the entry block or root, 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 heirarchical 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. 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. Data prepared by various independent studies are not always consistent with each other, and the DMS can be used to identify the areas of inconsistency. Even various maps within a single study may have points of inconsistency: for example, maps may indicate a water level below the base of the aquifer. The system easily can be used to check for these kinds of inconsistencies.

The major use of the DMS is to provide data for use in ground-water flow models; by properly using the system, the data for the models will be much

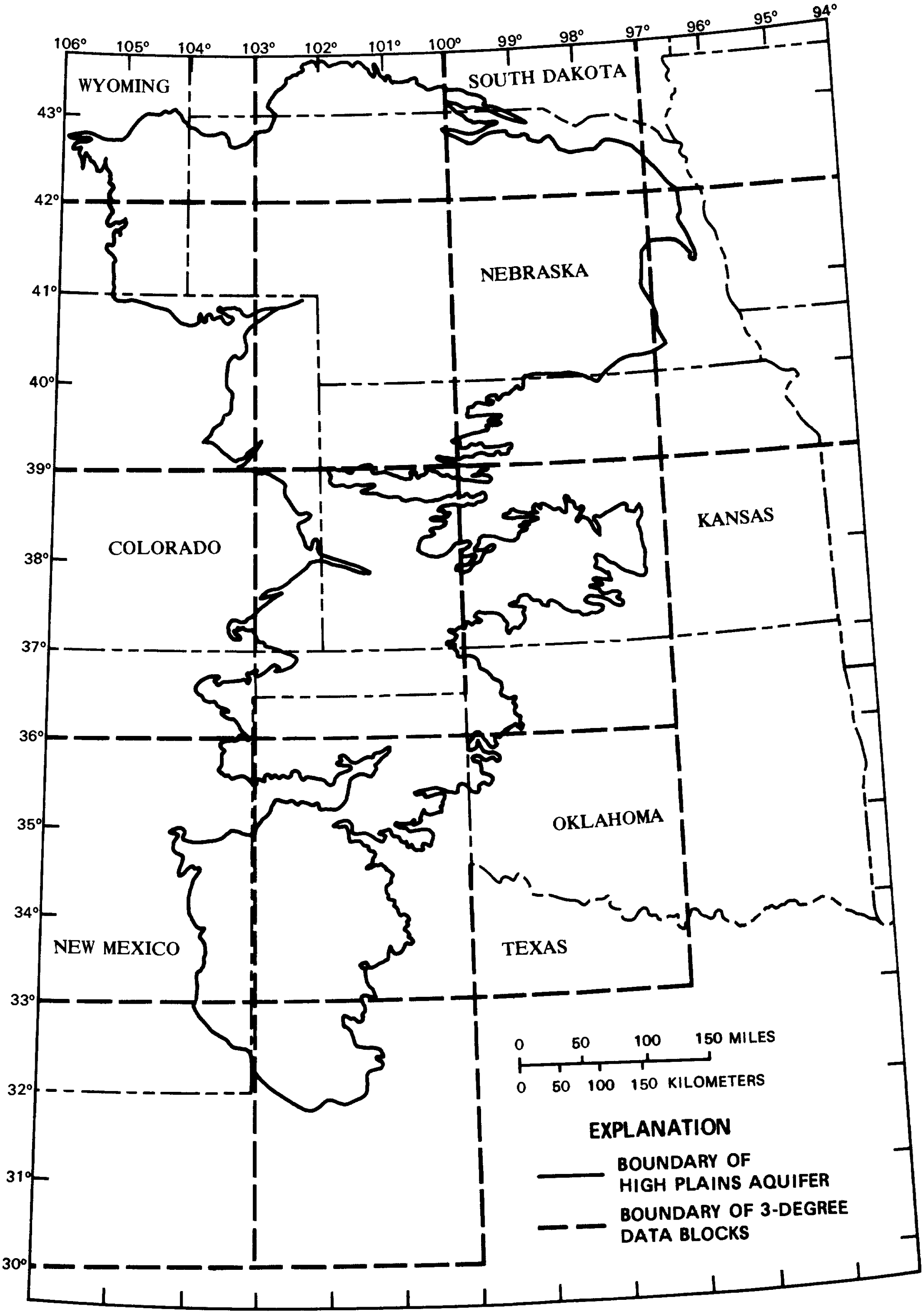


Figure 1.--High Plains study area with 3-degree data blocks.

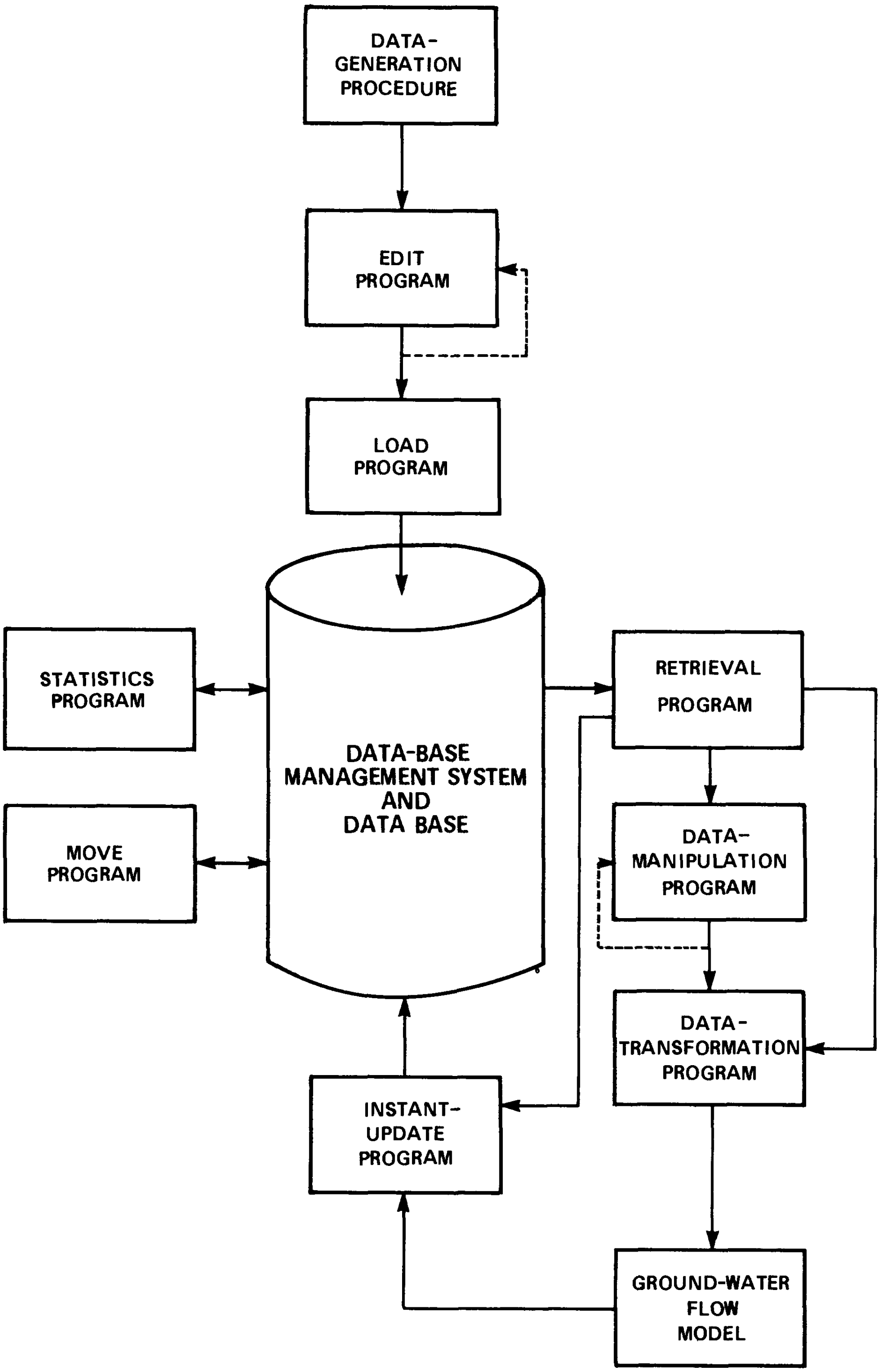

DATA-GENERATION PROCEDURE
EDIT PROGRAM
LOAD PROGRAM
STATISTICS PROGRAM
MOVE PROGRAM
DATA-BASE MANAGEMENT SYSTEM AND DATA BASE
RETRIEVAL PROGRAM
DATA-MANIPULATION PROGRAM
DATA-TRANSFORMATION PROGRAM
INSTANT-UPDATE PROGRAM
GROUND-WATER FLOW MODEL

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

more consistent. If the data in the data base does need revision, the DMS can be used to temporarily update the data and check the validity of the update before a permanent revision is made.

The DMS has been linked to two computer programs produced by Calcomp Computer Products, Inc.[1] The computer programs can be used to generate contour maps and three-dimensional views using data retrieved from the data base. The DMS also could be linked to other machine-graphics systems.

PURPOSE AND SCOPE

The purpose of this report is to provide a description of the complete DMS and the data that has been stored in the High Plains RASA data base. The report gives users who have access to the U.S. Geological Survey computer system and the High Plains RASA data base enough information to retrieve and use the data, and, if authorized, update the data base. In addition, the report outlines the data-base structure and all major computer programs of the complete DMS. This information is needed by persons who will adapt this system for use by other studies that need to store areal map information.

The report is in two parts: a main body and a supplemental section. The main body gives a general description of the complete 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 section. The supplemental section is designed to give users of the existing DMS complete information needed to use it. In addition, that section 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 is used to manage diverse geographic information for a multistate area and provide data for regional ground-water models. The geographic information 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.

[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.

Data-Base Structure

In order 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 1-minute by 1-minute block of latitude and longitude, the root-level record could consist of components describing the latitude, longitude, and aquifer of the node. Such a structure would need 27 million characters of storage for 44 parameters. An alternative data-base structure would require less storage. The alternative structure consists of root-level records that contain the aquifer and parameter names. Using this structure, the data for each parameter would be stored at the levels below the root level. Each tree or logical data set would then consist of the data for one parameter and one aquifer. With this structure the size of the data base is greatly decreased and for the High Plains study, the data-base size is approximately 10.9 million characters of information for 44 parameters.

One factor contributing to the decreased size of the data base is that the latitude and longitude is not stored for each 1-minute block. Only one coordinate for each 10-minute block is stored and from this coordinate, the latitude and longitude of each 1-minute 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 High Plains study, four degrees of detail were chosen to store the required data. In all instances, the root level of data (level 0) contains, at a minimum, the parameter and aquifer 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 components that describe the units of measurement for the stored data 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 data value is needed for each 3-degree by 3-degree block (latitude-longitude extent) within the aquifer, then the data can be stored at a density of 1. Density-1 data are referred to as 3-degree data. A density-1 parameter consists of one record at level 0, and a series of level-1 records describing the 3-degree blocks within the aquifer. The 3-degree records contain the parameter value and the latitude and longitude of the southeast corner of the 3-degree block. The records also contain four statistical components: the minimum, the maximum, the number of values, and the standard deviation. For 3-degree data, these statistics reflect the data used to compute the single data value for the entire 3-degree block. These same statistical components can be found at the lower levels of the data structure.

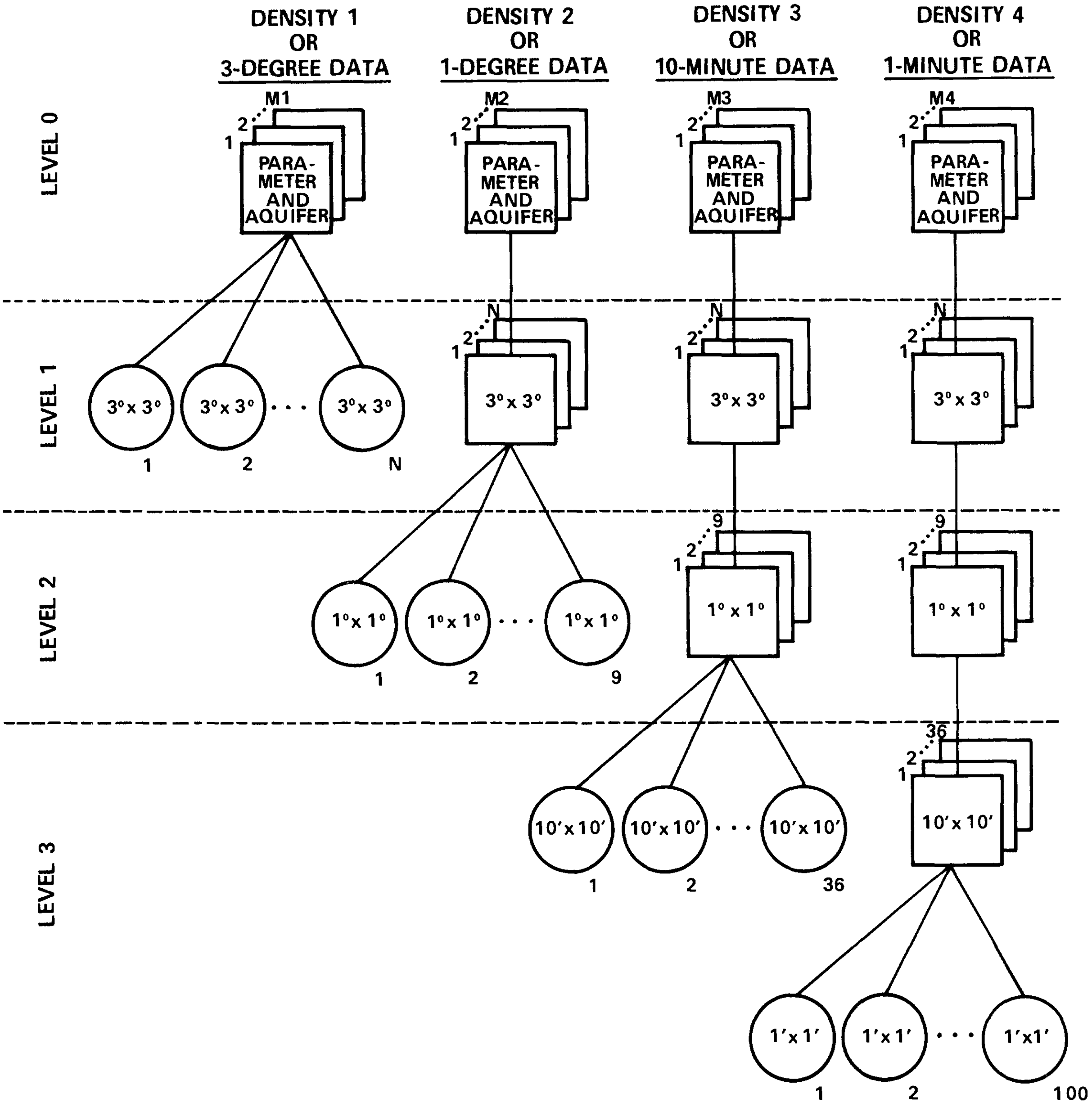


Figure 3.--Structure of the data base for the four possible densities.

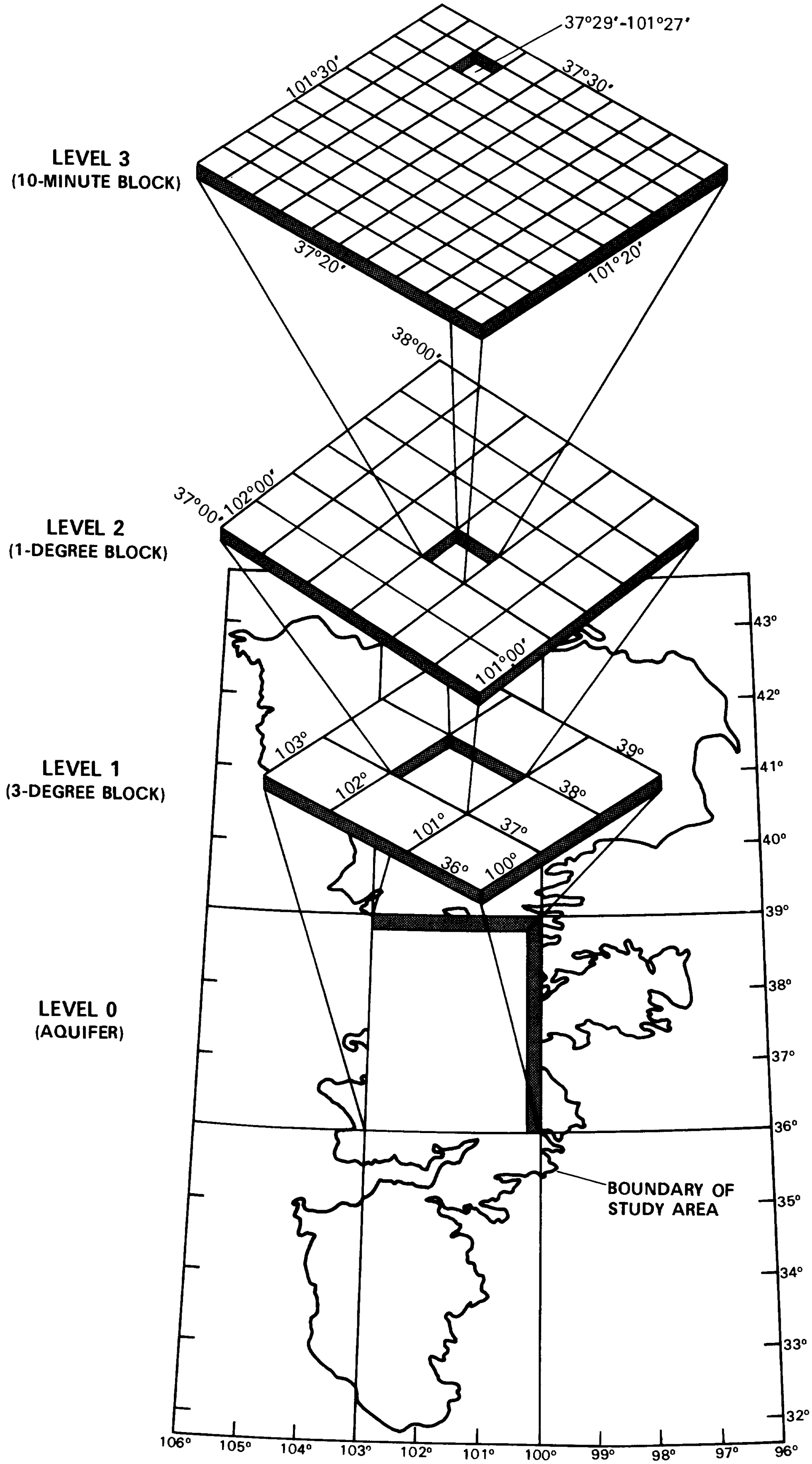


Figure 4.--Geographic area of the data-base levels.

A density-2 parameter has one data value for each 1-degree by 1-degree block within the aquifer. Density-2 data are called 1-degree data. The records for each 1-degree block are stored at level 2 of the data-base structure. The 1-degree blocks are logically connected to the appropriate 3-degree block record, which is at level 1. Level 1 contains a record for each 3-degree block within the aquifer. The 1-degree records contain the latitude and longitude of the southeast corner of the block, the data value, and the four statistical components. The four statistical components summarize the data used to compute the single data value for the 1-degree block.

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

A density-4 parameter has one data value for each 1-minute by 1-minute block within the aquifer. Density-4 data are referred to as 1-minute data. The 1-minute blocks are grouped with the appropriate 10-minute block. A record for each of the 10-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 data values for the corresponding 1-minute blocks. The record also contains five statistical components, which, in this case, are determined from the 100 data values.

Logically, the 1-minute data values are at a level below the 10-minute data, because they are of greater detail. However, the 1-minute blocks 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 data at levels 3 and 4.

As discussed previously, the data base has a hierarchical structure of four levels with the root level containing information that uniquely identifies a particular parameter with an aquifer. The lowest level or base level at which a parameter's data resides depends on the density; the actual data 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.

Several factors control the density at which a parameter is stored. One factor is 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. The 3-degree by

3-degree block is approximately 36,000 mi^2 at 32 degrees latitude, and approximately 31,400 mi^2 at 42 degrees latitude. A 1-degree block is approximately 4,000 mi^2 at 32 degrees latitude, and 3,500 mi^2 at 42 degrees latitude. A 10-minute block is approximately 111 mi^2 at 32 degrees latitude, and 97 mi^2 at 42 degrees latitude. Finally, a 1-minute block is approximately 1.1 mi^2 at 32 degrees latitude, and 1.0 mi^2 at 42 degrees latitude.

Three other major controlling factors for determining data density are: (1) The detail of the data needed for modeling; (2) the detail of the source material (usually a map); and (3) the distribution of variance of the data over the map. In the case of water-level data, maximum detail is needed for modeling and the source maps are very detailed. Hence, the water levels were stored as 1-minute data. Land-use and water-use data are required in as much detail as possible. For the High Plains study, detailed data were not available; in most instances, data were only available for counties. Using this data and other source material, the data were extrapolated and stored as 10-minute data. The only exceptions were the 1978 and 1980 land-use data that were available in detail; data for these years were stored as 1-minute data. Although detailed climatic data are available, they were not required. These data were stored as 1-degree data.

Statistics

Actual data 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 data 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, a 10-minute parameter has its actual data value 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 1-degree statistics (level 2) are computed by using the parameter values for all of the 10-minute blocks within the 1-degree block. The 3-degree level statistics (level 1) are computed by using parameter values for all of the 10-minute blocks within the 3-degree block. Finally, a set of statistics are stored at level 0, that are a reflection of all of the 10-minute blocks within the aquifer. Therefore, the value component can have two meanings: (1) At the base level of a parameter, the value component is the actual data value for that block; and (2) at higher levels within the structure, the value component is the mean of the data 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.

Proposals

An important aspect of a DMS is to provide a method of updating. The method chosen for the High Plains study is called the proposal concept. A proposal is a separate record that can be added to the data structure for a parameter. It consists primarily of a proposed data value and the latitude and longitude of the data block. This proposed parameter value is for a block at the base level at which a parameter is stored. For example, a proposed value for a 10-minute parameter would correspond to a 10-minute block. A user can propose a new value for any valid block within the aquifer.

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 Data Base Administrator (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

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 directly usable. Only the programs that interact with the data base and the DBMS would have to be changed if the DBMS were changed. The program changes should be accomplished easily if a DBMS based on a heirarchical data structure were selected. If another type of DBMS were selected, these program changes would most likely be very extensive.

TYPES OF DATA STORED

The general philosophy of the High Plains RASA is to store only primary data in the data base 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 a modification 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 modification becomes even more complex.

Five general classes of data are stored in the High Plains RASA data base: (1) Aquifer geometry; (2) aquifer and water characteristics; (3) water levels;

(4) climatic data; and (5) land- and water-use information. All data stored in the data base is listed in table 1, with a brief description of the source of the data. In December 1982, the data base contained 44 parameters and more than 10.9 million characters of information.

The general data class of aquifer geometry consists of data on altitude and geology of the base of the aquifer (Weeks and Gutentag, 1981), altitude of land surface, and boundary of the aquifer. Altitude of land surface comes from 1:250,000 scale, U.S. Geological Survey topographic maps. All other data was generated by the High Plains RASA project, using all available data sources.

Aquifer characteristics include hydraulic conductivity and specific yield of the High Plains aquifer. These characteristics were mapped by the High Plains RASA project by analyzing lithologic logs from numerous wells. Water characteristics include dissolved-solids and sodium concentrations (Krothe and others, 1982).

Water level information for various years is stored in the data base (Gutentag and Weeks, 1980). Water-level maps were generated by the High Plains RASA project at 5-year intervals for 1960 through 1980. A map showing water levels prior to any extensive development of the aquifer also was prepared. Water levels are used directly in the modeling, and also are combined with each other and with other data to generate saturated-thickness maps, transmissivity maps, and water-level change maps (Luckey and others, 1981).

Generalized climatological information stored in the data base includes annual lake evaporation and annual normal precipitation. These were obtained from small-scale National Weather Service maps (U.S. Department of Commerce, 1977); they are stored because they are related in a general way to the recharge to the aquifer.

The general class of land- and water-use information contains a number of varied parameters. Crop requirements, by crop type, were calculated by the High Plains RASA project for normal precipitation and temperature conditions. A composite irrigation demand for various years was then calculated, using the mix of crops reported by the U.S. Bureau of Census. Census data also were used to define irrigated acreage for various years (Heimes and Luckey, 1982). By combining composite irrigation demand and irrigated acreage, and assuming an irrigation efficiency, ground-water pumpage can be estimated. Much more detailed information on irrigated land, as well as range land and dryland, was obtained from Landsat-satellite imagery (Donovan and others, 1982). These data can be used to generate pumpage data and to help estimate location of recharge from precipitation. Data on soil type, obtained from State soil maps, also are stored in the data base and could be used to estimate recharge.

Table 1.--*Parameters in the High Plains data base*

[in, inch; ft, foot; ft/d, feet per day; pct, percent; mg/L, milligrams per liter]

Parameter name	Mean value	Density[1]	Number of points	Source
ANNUAL LAKE EVAPORATION	54.5 in	1°	77	National Weather Service maps.
ANNUAL NORMAL PRECIPITATION	19.1 in	1°	77	Do.
BASE OF AQUIFER - ALTITUDE	2,933 ft	1'	166,639	High Plains RASA project.
BASE OF AQUIFER - GEOLOGY	[2]	10'	2,001	Do.
BOUNDARY	[2]	1'	518,400	Do.
CROP REQUIREMENT - ALFALFA	24.5 in	1°	77	Do.
CROP REQUIREMENT - CORN	15.5 in	1°	77	Do.
CROP REQUIREMENT - COTTON	9.4 in	1°	77	Do.
CROP REQUIREMENT - DRY BEANS	13.8 in	1°	77	Do.
CROP REQUIREMENT - SORGHUM	12.0 in	1°	77	Do.
CROP REQUIREMENT - SOYBEANS	9.5 in	1°	77	Do.
CROP REQUIREMENT - SUGAR BEETS	20.7 in	1°	77	Do.
CROP REQUIREMENT - WHEAT	8.1 in	1°	77	Do.
DISSOLVED SOLIDS	426 mg/L	10'	2,001	Do.
DRYLAND - 1978 - LANDSAT	26.11 pct	1'	174,871	Landsat satellite.
HYDRAULIC CONDUCTIVITY	55.68 ft/d	10'	1,781	High Plains RASA project.
IRRIGATED LAND - 1949 - CENSUS	1,163 acres	10'	1,781	U.S. Census of Agriculture.
IRRIGATED LAND - 1954 - CENSUS	2,217 acres	10'	1,781	Do.
IRRIGATED LAND - 1959 - CENSUS	3,530 acres	10'	1,781	Do.
IRRIGATED LAND - 1964 - CENSUS	3,923 acres	10'	1,781	Do.
IRRIGATED LAND - 1969 - CENSUS	5,019 acres	10'	1,781	Do.
IRRIGATED LAND - 1974 - CENSUS	5,918 acres	10'	1,781	Do.
IRRIGATED LAND - 1978 - CENSUS	7,262 acres	10'	1,781	Do.
IRRIGATED LAND - 1978 -LANDSAT	13.15 pct	1'	174,871	Landsat satellite.
IRRIGATED LAND - 1980 -LANDSAT	13.07 pct	1'	168,619	Do.
IRRIGATION DEMAND - 1949	16.5 in	10'	1,781	High Plains RASA project.
IRRIGATION DEMAND - 1954	16.4 in	10'	1,781	Do.
IRRIGATION DEMAND - 1959	15.0 in	10'	1,781	Do.
IRRIGATION DEMAND - 1964	14.7 in	10'	1,781	Do.

Table 1.--*Parameters in the High Plains data base*--Continued

Parameter name	Mean value	Density[1]	Number of points	Source
IRRIGATION DEMAND - 1969	14.7 in	10'	1,781	High Plains RASA project.
IRRIGATION DEMAND - 1974	14.6 in	10'	1,781	Do.
LAND SURFACE	3,244 ft	10'	1,781	U.S.G.S. topographic maps.
RANGELAND - 1978 - LANDSAT	60.05 pct	1'	174,871	Landsat satellite.
SODIUM CONCENTRATION	34 mg/L	10'	2,001	High Plains RASA project.
SOIL TYPE	[2]	10'	1,781	Do.
SPECIFIC YIELD	14.14 pct	10'	1,781	Do.
S.W. IRR. LAND - 1959 - CENSUS	383 acres	10'	1,781	U.S. Census of Agriculture.
WATER TABLE - PRED	3,135 ft	1'	166,639	High Plains RASA project.
WATER TABLE - 1960	3,139 ft	1'	166,639	Do.
WATER TABLE - 1965	3,134 ft	1'	166,639	Do.
WATER TABLE - 1970	3,132 ft	1'	166,639	Do.
WATER TABLE - 1975	3,131 ft	1'	166,639	Do.
WATER TABLE - 1978	3,126 ft	10'	1,781	Do.
WATER TABLE - 1980	3,127 ft	1'	166,639	Do.

[1] One value per block of indicated size.

[2] Item assigned arbitrary code value so mean value is meaningless.

DATA GENERATION

Most of the data stored in the High Plains RASA data base originated as data portrayed on some type of map. These maps are prepared at various scales and at various degrees of detail. The scale of the original maps ranged from 1:250,000 (about 4 mi/in.) to about 1:19,000,000 (about 300 mi/in.). While no control is possible over the degree of detail of maps from published sources, the maps prepared for the High Plains RASA were made at the degree of detail needed for both regional and local ground-water management models. After maps were obtained or constructed, average values for data-base blocks were obtained by using either automated or manual procedures.

Automated Procedure

The automated procedure is used when 10- or 1-minute data are required, and original data are in the form of a contour map. If less detail is required, a manual procedure is more efficient. The automated procedure consists of four steps: (1) The contours are digitized to (X, Y, Z) triplets; (2) the (X, Y, Z) triplets are converted to (longitude, latitude, Z) triplets; (3) the (longitude, latitude, Z) triplets are converted to one value for each 10-minute or 1-minute block; and (4) all blocks outside the limit of the aquifer are eliminated (fig. 5).

The first step in the automated procedure is to digitize the contours. This is done using a Numonics 1224 digitizer operating in a mapping incremental mode and linked to a Datapoint 5500 minicomputer. Generally, a 1-degree by 1-degree or 2-degree by 2-degree block is digitized at one time.

The second step of the procedure is to convert the (X, Y, Z) triplet from the first step to a (longitude, latitude, Z) triplet. The coordinates are adjusted for map translation, map rotation, and convergence of longitude lines. The coordinates are not corrected for the curvature of the latitude lines; however, this curvature is insignificant for small blocks.

The third step of the procedure is to convert the (longitude, latitude, Z) triplets into average values for the 10-minute or 1-minute block. First, the data are gridded in either the north-south direction or the east-west direction; that selection is governed by the general trend of the contour lines. For gridding in a north-south direction for 10-minute blocks, the (longitude, latitude, Z) triplets are scanned to find successive triplets that span a 5-minute, 15-minute, 25-minute, 35-minute, 45-minute, or 55-minute latitude line. A linear interpolation is then done on the longitude, and a new triplet is generated with the latitude of the desired line. After this is done for all possible latitude lines of interest, and the new triplets are sorted, a data set is generated that defines the position and value of all contours that cross the latitude line. An identical procedure is followed for gridding in the east-west direction with the longitude value fixed, and the interpolation done on the latitude line. If data are desired for 1-minute blocks, the data are gridded at the 0.5-minute lines rather than the 5-minute

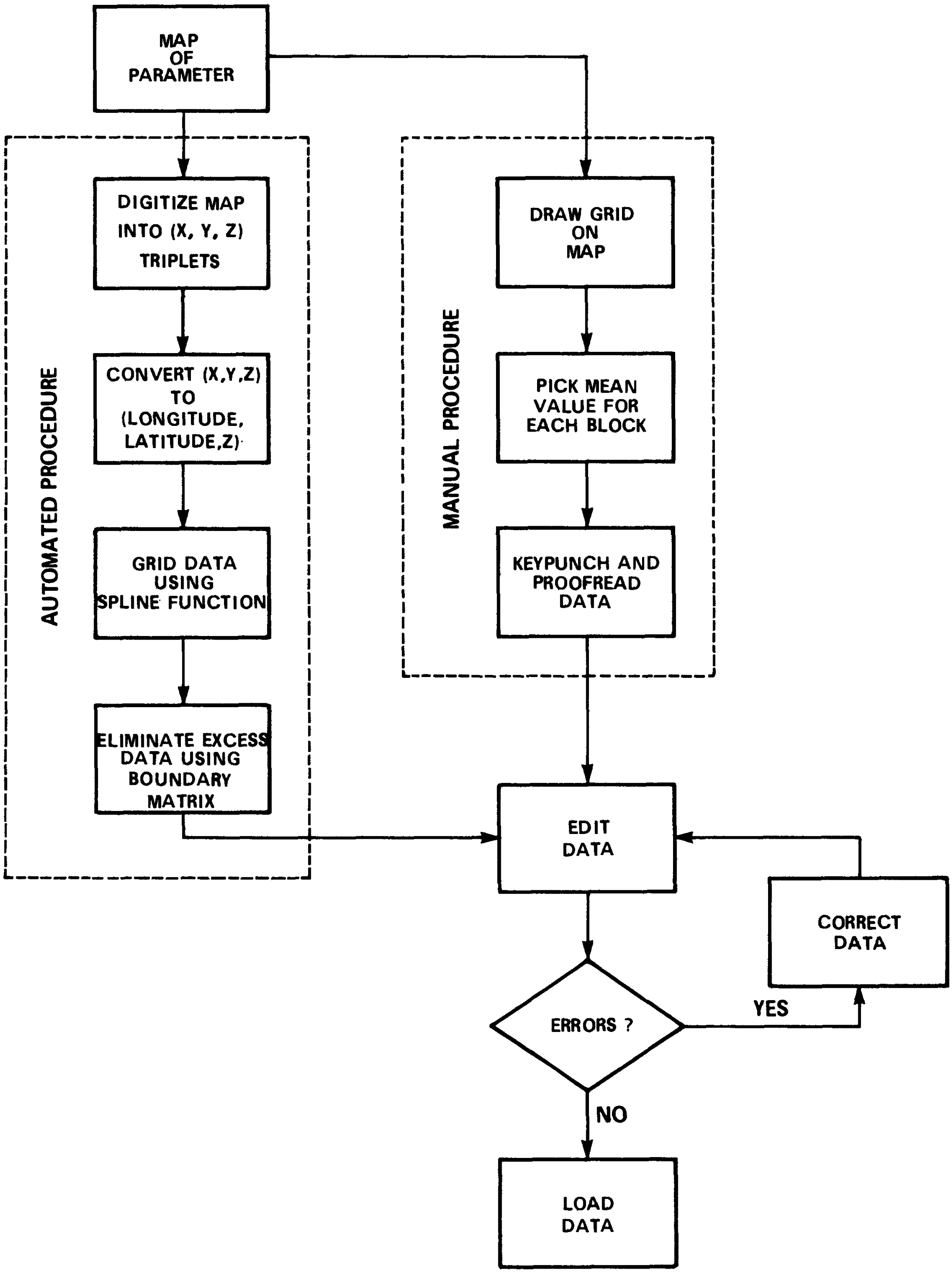

MAP
OF
PARAMETER
AUTOMATED PROCEDURE
DIGITIZE MAP INTO (X, Y, Z) TRIPLETS
CONVERT (X,Y,Z) TO (LONGITUDE, LATITUDE,Z)
GRID DATA USING SPLINE FUNCTION
ELIMINATE EXCESS DATA USING BOUNDARY MATRIX
MANUAL PROCEDURE
DRAW GRID ON MAP
PICK MEAN VALUE FOR EACH BLOCK
KEYPUNCH AND PROOFREAD DATA
EDIT DATA
ERRORS ?
YES
CORRECT DATA
NO
LOAD DATA

Figure 5.--Flowchart illustrating steps in data-generation procedure.

lines. The data are then gridded along each latitude (or longitude) line by first fitting a cubic-spline function through all the (longitude, Z) values along each line. A cubic-spline function is the smoothest interpolation function that fits all of the contour crossing points exactly. The cubic-spline function is used to generate Z values at 5-minute or 0.5-minute longitude (or latitude) intervals. At this point, a data value has been generated for the center of each 10-minute or 1-minute block.

The final step is to trim the data back to the boundary of the aquifer. Prior to digitizing, the original contours are extended beyond the boundary of the aquifer to increase accuracy of the interpolation process in the vicinity of the boundary. This results in data being generated beyond the limits of the aquifer. These generated data are compared to a control data set that defines the limits of the aquifer, and all generated points outside of the area are eliminated.

Manual Procedure

A manual procedure is used for some 10-minute data and for all 1-degree data. This procedure is preferable when the number of points to be generated is small or the mapped surface is highly irregular.

A template showing the 10-minute blocks within a 1-degree block, or the 1-degree blocks within a 3-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 keypunching 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. During the automated data-generation process, computer programs check correctness of the generated data against the original contour maps to insure that the generated data are consistent with the source material. The data are then again carefully machine-edited prior to loading them into the data base.

There are two main reasons for carefully editing generated data:

1. Data should be accurate. To insure accuracy, a manual-editing procedure is used. The data is 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 be sure that they can be loaded properly into the data base. The Edit Program is used to prepare generated 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 data records follow standard formats for use in the data-base Loading 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 3-degree or 1-degree block. Finally, the coordinates are examined to determine if proper 1-degree records are grouped with the appropriate 3-degree record, and that 10-minute records are grouped with the appropriate 1-degree record.

The end 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.

DATA LOADING

The data are loaded into a hierarchical data base for storage, update, and retrieval. The loading procedure is dependent on the DBMS. The DBMS stores the data and creates appropriate indices to the data. These indices are useful for updating or retrieving the data.

DATA UPDATING

Updating the data base is a two-step process. In the first step, accomplished through the use of the Instant-Update Program, users propose changes to data values. In the second step, the DBA replaces the appropriate permanent-data values with the acceptable 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 indicates that it is a test value. 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 time. Second, the user may decide that the test value 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 indicates the test proposal is acceptable, and it 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. In the retrieval, the user can request that test proposals be included in the retrieved data. For each block retrieved that has a corresponding test proposal, the test value will replace the permanent value. This data 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.

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 finds the corresponding permanent data 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 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 DBA 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.

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 aquifer 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 10-minute data is level 3. Hence, if the actual data values are required for a 10-minute parameter, 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 1-degree statistics for a 10-minute parameter 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 data block. If a test proposal is found, its value replaces the permanent data value in the retrieved data. If, for a particular data block, 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 available 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 water-level data, where the water level is 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 compatable with the application programs.

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 data. The second program is the interface between the data base and the model. This program transforms data-base data into a format for 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 available functions. The first function takes the logarithm of a data value in the form:

$$A*\log(X) + C$$

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

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

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

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

where A, B, and C are constants, and X1 and X2 are data 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 data values.

The main entry data to this program is a set of data in standard retrieval format; manipulated data also are in that format. The entry data can be data retrieved from the data base. It also can be an intermediate result from one of the steps in the process of calculating secondary data.

The major use of this program is to generate secondary data. Secondary data are data that are produced from parameters already stored in the data base. These stored data are called primary data.

Examples of secondary data are saturated thickness, water-level changes, and transmissivity. To illustrate use of the program, the steps to generate transmissivity will be discussed. Transmissivity is the product of saturated thickness and hydraulic conductivity. The required steps are:

1. Subtract altitude of base of aquifer from the water level (saturated thickness); and

2. Multiply saturated thickness by hydraulic conductivity.

These two steps result in secondary data, called transmissivity, in standard entry format. In fact, the transmissivity data could be loaded into the data base at this point.

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

Another use of the program is to check consistency of primary data. For example, by calculating secondary data, depth to water, the relationship between water level and land surface can be examined.

Data Transformation

The Data-Transformation Program is the interface between the data base and the model. This program produces a model matrix for the finite-difference model used for aquifer simulation. The program processes data retrieved from the data base. It will process 1-degree, 10-minute, and 1-minute data.

The program is divided into two major steps. In step 1, data 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 program user. The result of this step is a group of (X, Y, Z) 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. For average and weighted-average techniques, converted data are trimmed to fit the model area. For each (X, Y, Z) triplet, the block in which the point is located is determined. Then, for each model block, a value is calculated by using the data that are located within the block. It is quite possible that some of the model blocks cannot be assigned a value, because of a lack of data. In the instance of the trend-surface technique, all converted data are used to calculate the node values and all blocks are assigned a value. 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 6. There are no restrictions on the 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.

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.

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.

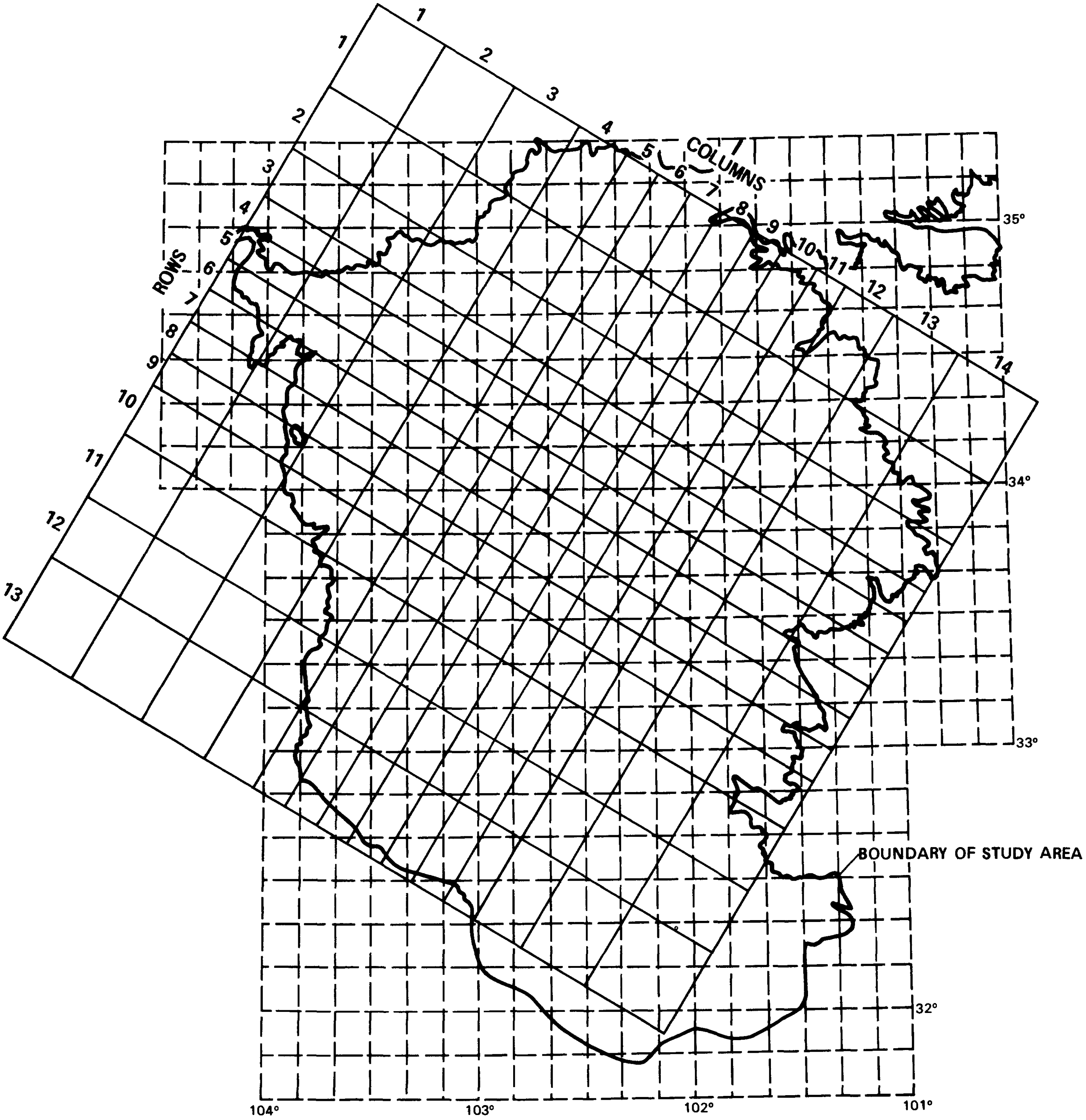
To generate three-dimensional views, the user specifies the following: (1) Title of plot; (2) position of observer; (3) distance of observer from surface; (4) smoothness of plot; and (5) exaggeration of plot in the Z-direction. The Graphics Program then generates the data and commands needed by the Calcomp 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 system because the computer programs were available and usable. The data base can be linked to other available commercial graphics program packages as needed.

SUMMARY

The High Plains RASA study, which began in 1978, has developed a regional water-resources (and related) data storage, retrieval, and update system to organize and manipulate areal interpretive data needed to develop regional and local ground-water flow models. The data storage and retrieval system is general and can easily be adapted for other studies.

The system is built around a hierarchically structured data base consisting of related latitude-longitude blocks. Various parameters in the data base are stored at different degrees of detail, with the finest detail being a block of 1 minute of latitude by 1 minute of longitude (approximately 1 mi^2).



POSITION OF PRINCIPAL NODE: ROW 6, COLUMN 8
LATITUDE OF PRINCIPAL NODE: 34°00'00''
LONGITUDE OF PRINCIPAL NODE: 102°30'00''
ROTATION ANGLE: 30°

EXPLANATION

-+- DATA-BASE GRID

╳ MODEL GRID

Figure 6.--Hypothetical-model grid overlaid on the data-base grid.

Statistics about the parameter are stored at all higher levels in the data base, including 10-minute blocks, 1-degree blocks, and 3-degree blocks, so the parameter is available at all levels above the base level. The system allows immediate updates to the data base that are virtually transient in nature. The method of updating the data provides immediate updates, while preserving the integrity of the permanent part of the data base.

Five general classes of data have been stored in the data base: (1) Aquifer-geometry data; (2) aquifer and water characteristics; (3) water levels; (4) climatological data; and (5) land- and water-use data. These data were compiled from maps, using both automated and manual data-generation procedures.

Data can be retrieved from the data base, combined with other data in the same format, and automatically regridded into any finite-difference, flow-model grid. The system thus accomplishes the objective of organizing data for use in the flow models.

REFERENCES

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

Donovan, W., Koch, C., Morse, A., Jones, H., and Heimes, F., 1982, One billion pixels; analysis and reduction of Landsat data for the High Plains hydrological model (abs.): American Society of Photogrammetry, Proceedings, 48th Annual Meeting, Denver, Colorado, March, 1982, p. 562.

Gutentag, E. D., and Weeks, J. B., 1980, Water table in the High Plains aquifer in 1978 in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Hydrologic Investigations Atlas HA-642.

Heimes, F. J., and Luckey, R. R., 1982, Method for estimating historical irrigation requirements from ground water in the High Plains in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Water-Resources Investigations 82-40, 64 p.

Krothe, N. C., Oliver, J. W., and Weeks, J. B., 1982, Dissolved solids and sodium in water from the High Plains aquifer in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Hydrologic Investigations Atlas HA-658.

Luckey, R. R., Gutentag, E. D., and Weeks, J. B., 1981, Water-level and saturated-thickness changes, predevelopment to 1980, in the High Plains aquifer, in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Hydrologic Investigations Atlas HA-652.

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, Book 7, Chapter C1, 115 p.

U.S. Department of Commerce, 1977, Climatic atlas of the United States: Environmental Sciences Services Administration, Environmental Data Service, 80 p.

Weeks, J. B., 1978, Plan of study for the High Plains regional aquifer-system analysis in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Water-Resources Investigations 78-70, 28 p.

Weeks, J. B., and Gutentag, E. D., 1981, Bedrock geology, altitude of base, and 1980 saturated thickness of the High Plains aquifer in parts of Colorado, Kansas, Nebraska, New Mexico, Oklahoma, South Dakota, Texas, and Wyoming: U.S. Geological Survey Hydrologic Investigations Atlas HA-648.

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 that uniquely define the data stored at the lower levels of the structure. The aquifer unit and parameter name are part of the entry record. At level 1, data for a 3-degree block are stored. At level 2, data for a 1-degree block are stored. At level 3, data for 10-minute blocks are stored. Logically, 1-minute data should be found 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 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 2. 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 and PROP contain 100 schema items that correspond to the 100 1-minute data blocks for 1-minute data within a 10-minute block. All these schema items are not listed, because the items are identically formatted. Arrangement of the 1-minute schema items within the 10-minute block is shown in figure 7.

Table 2.--*Data-base dictionary*

Schema or component		Format		
Number	Name	Type	Length	Remarks
				LEVEL 0 - ENTRY RECORD
0	ENTRY	-	-	Level 0 schema record
1	AQPARM	Character	38	Concatenation of aquifer unit and parameter name
2	UNIT	Character	8	Aquifer unit
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	MNO	Integer	5	Minimum parameter value
12	MXO	Integer	5	Maximum parameter value
13	AVGO	Integer	5	Average parameter value
14	NUMO	Integer	7	Number of data values used to compute statistics
15	SDVO	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 2.--*Data-base dictionary*--Continued

Schema or component		Format		
Number	Name	Type	Length	Remarks
				LEVEL 1 - 3-DEGREE DATA RECORD
100	LEV1	-	-	Level 1 schema record - child of entry record
101	LAT1	Integer	6	Latitude of southeast corner of 3-degree block (DDMMSS)
102	LONG1	Integer	7	Longitude of southeast corner of 3-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 data values used to compute statistics
115	SDV1	Decimal	7	Standard deviation - 2 decimal places
				LEVEL 2 - 1-DEGREE DATA RECORD
200	LEV2	-	-	Level 2 schema record - child of 3-degree data record
201	LAT2	Integer	6	Latitude of southeast corner of 1-degree block (DDMMSS)
202	LONG2	Integer	7	Longitude of southeast corner of 1-degree 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 data values used to compute statistics
215	SDV2	Decimal	7	Standard deviation - 2 decimal places

Table 2.--*Data-base dictionary*--Continued

Schema or component		Format		
Number	Name	Type	Length	Remarks
				LEVEL 3 - 10-MINUTE DATA RECORD
350	LEV3B	-	-	Level 3 schema record - child of 1-degree data record
351	LAT3B	Integer	6	Latitude of southeast corner of 10-minute block (DDMMSS)
352	LONG3B	Integer	7	Longitude of southeast corner of 10-minute block (DDDMMSS)
353	LAT3BD	Decimal	7	Decimal equivalent of latitude, in degrees - 5 decimal places
354	LONG3BD	Decimal	8	Decimal equivalent of longitude, in degrees - 5 decimal places
361	MN3B	Integer	5	Minimum parameter value
362	MX3B	Integer	5	Maximum parameter value
363	AVG3B	Integer	5	Average parameter value
364	NUM3B	Integer	5	Number of data values used to compute statistics
365	SDV3B	Decimal	7	Standard deviation - 2 decimal places
				LEVEL 3 - 1-MINUTE DATA RECORD
300	LEV3	-	-	Level 3 schema record - child of 1-degree data record
301	LAT3	Integer	6	Latitude of southeast corner of 10-minute block (DDMMSS)
302	LONG3	Integer	7	Longitude of southeast corner of 10-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 data values used to compute statistics
315	SDV3	Decimal	7	Standard deviation - 2 decimal places
401- 500	AA- JJ	Integer	5	1-minute parameter matrix - 10 rows by 10 columns - see figure 7 for arrangement

Table 2.--*Data-base dictionary*--Continued

Schema or component		Format		
Number	Name	Type	Length	Remarks
				LEVEL 1 - MULTI-VALUE PROPOSAL RECORD
599	PROP	-	-	Level 1 schema record - used to propose changes to parameter values for 1-minute data - child of entry record
601	LATP	Integer	6	Latitude of southeast corner of 10-minute block (DDMMSS)
602	LONGP	Integer	7	Longitude of southeast corner of 10-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	1-minute parameter proposal matrix - 10 rows by 10 columns - see figure 7 for arrangement
				LEVEL 1 - SINGLE-VALUE PROPOSAL RECORD
899	PROP2	-	-	Level 1 schema record - used to propose changes to parameter values for 3-degree data, 1-degree data and 10-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	Average parameter value
914	NUM9	Integer	3	Number of data 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

PEA — COMPONENT NAME
741 — COMPONENT NUMBER

Figure 7.--Arrangement of 1-minute block schema items in a 10-minute block for the LEV3 and PROP schema records.

SUPPLEMENT II: PROGRAM-USER DOCUMENTATION

This supplement contains documentation for the major computer programs used for the 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 14 sets of documentation contained within this supplement. The first five sets of documentation are for data-generation programs. The next six sets of documentation are for programs that interact with data base and the DBMS. The final three sets of documentation are for application programs that use data retrieved from the data base.

The Digitizer Data-Conversion Program

This program converts digitizer-data output in an X-Y coordinate system to the latitude-longitude coordinate system. The program accounts for translation of the origin, skew of the coordinate system, and convergence of the longitude lines. It does not account for curvature of the latitude lines that occurs with some map projections. The program requires that north on the map be at the top of the sheet on the digitizer, but it allows as much as a 90-degree rotation.

The primary input file consists of four control points, a check point, and the digitizer input. The four control points are tne lower-left corner, the lower-right corner, the upper-right corner, and the upper-left corner of the area being digitized. These points need to be entered in this order. The check point is the same as the first control point, and the X-Y coordinates of these two points need to match within 2 percent. The digitizer coordinates need to be reset to (0,0) before entering the first control point.

An output record consists of the input X-Y coordinates, corrected X-Y coordinates, and latitude-longitude coordinates. A number of error and informational messages may be produced; these should all be self-explanatory.

Mathematics

The program first does a rotation of the coordinate system using the formulas

$$X = X' \cos \theta + Y' \sin \theta$$

$$Y = Y' \cos \theta - X' \sin \theta$$

where

$$\sin \theta = \frac{Y'_2}{\sqrt{(X_2')^2 + (Y_2')^2}}$$

$$\cos \theta = \frac{X_2'}{\sqrt{(X_2')^2 + (Y_2')^2}}$$

where X' and Y' denote the coordinate system prior to rotation; X and Y denote the coordinate system after rotation; and the subscript denotes the control-point number.

The points are scaled using the scale factors

$$\text{Scale}(X) = \frac{\text{longitude}_3}{X_3}$$

$$\text{Scale}(Y) = \frac{\text{latitude}_3}{Y_3}$$

with the same notation as above.

Correction for convergence of the longitude lines is accomplished using the formula

$$X_{\text{CORRECTED}} = X + [(\tfrac{1}{2}X_2 - X)/\tfrac{1}{2}X_2] \cdot (Y/Y_4) \cdot X_4$$

The Y coordinate of the point does not need to be corrected for convergence.

The translation is accomplished by

$$\text{longitude} = \text{longitude}_1 - Y \cdot \text{Scale}(Y)$$

$$\text{latitude} = \text{latitude}_1 + X_{\text{CORRECTED}} \cdot \text{Scale}(X)$$

These final values are converted to degrees-minutes-seconds notation prior to output.

Input

One parameter card, read on logical unit 5, is required. The format of the card is:

Col 1-10	Logical-unit number for input (right justified)
Col 11-20	Logical-unit number for output (right justified)

The primary input file consists of four control points, a check point, and any number of digitizer data points. This file generally is a tape or disk file and is read from the logical unit defined on the parameter card. The control points occupy the first four records (cards) under the following format:

Col 1-8	X - coordinate (decimal)
Col 9-18	Y - coordinate (decimal)
Col 19-20	Blank
Col 21-23	Control-point latitude (degrees)
Col 24-25	Control-point latitude (minutes)
Col 26-27	Control-point latitude (seconds)
Col 28-29	Blank
Col 30-32	Control-point longitude (degrees)
Col 33-34	Control-point longitude (minutes)
Col 35-36	Control-point longitude (seconds)

The check point occupies the fifth record (card) and the digitizer data points occupy from the sixth and subsequent records of the file. All are under the following format:

Col 1-8	X - coordinate (decimal)
Col 9-18	Y - coordinate (decimal)
Col 19-20	Blank
Col 21-27	Z - coordinate (character)

Output

The output from the program, generally a tape or disk file, consists of one record for each digitizer data point under the following format:

Col 1-4	Blank
Col 5-12	Input X-coordinate
Col 13-20	Input Y-coordinate
Col 21-28	X-coordinate; rotated, corrected, and scaled
Col 29-36	Y-coordinate; rotated, corrected, and scaled
Col 37-40	Blank
Col 41-47	Latitude, in degrees, minutes, and seconds
Col 48-51	Blank
Col 52-58	Longitude, in degrees, minutes, and seconds
Col 59-60	Blank
Col 61-68	Input Z-coordinate

The output file is written to the logical unit defined on the parameter card.

Sample Procedure

A procedure can be used to execute the program with the procedure name DIGLL. To execute the procedure, using disk input and disk output, the following Job Control Language (JCL) could be used:

```
// . . . JOB . . .
//PROCLIB DD DSN=procedure.library,DISP=SHR
//  EXEC  DIGLL,NAME1=input,NAME2=output
//SYSIN  DD *
   parameter card
//
```

where *input* is the data-set name of the primary input file and *output* is the data-set name of the output file. The procedure DIGLL could appear as follows:

```
//DIGLL PROC UNIT1='3330-1',VOL1=myvol,NAME1=NULLFILE,UNIT2='3330-1',
//       VOL2=myvol,NAME2=NULLFILE
//*********************************************************************
//*DIGLL -- A PROCEDURE TO RUN PROGRAM DIGLL
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES):
//*
//*         UNIT1      INPUT POINT DATA DEVICE (3330-1)
//*         VOL1       INPUT POINT DATA VOLUME (myvol)
//*         NAME1      INPUT POINT DATA DSNAME (NULLFILE)
//*         UNIT2      CONVERTED DATA OUTPUT DEVICE (3330-1)
//*         VOL2       CONVERTED DATA OUTPUT VOLUME (myvol)
//*         NAME2      CONVERTED DATA OUTPUT DSNAME (NULLFILE)
//*
//*********************************************************************
//DIGLL  EXEC  PGM=DIGLL,TIME=2,REGION=52K
//*  LIBRARY IDENTIFICATION
//STEPLIB  DD  UNIT=3330-1,VOL=SER=myvol,DISP=SHR,
//         DSNAME=mylib
//*  CARD READER IS LOGICAL UNIT 5
//FT05F001  DD  DDNAME=SYSIN
//* PRINTER IS LOGICAL UNIT 6
//FT06F001  DD  SYSOUT=A
//*  PUNCH IS LOGICAL UNIT 7
//FT07F001  DD  SYSOUT=B
//*  INPUT UNIT FOR POINT DATA -- LOGICAL UNIT 10 ASSUMED
//FT10F001  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*  OUTPUT UNIT FOR CONVERTED DATA -- LOGICAL UNIT 11 ASSUMED
//FT11F001  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(NEW,KEEP,DELETE),
//  DCB=(RECFM=FB,LRECL=68,BLKSIZE=6800),SPACE=(TRK,(5,5),RLSE),
//  DSNAME=&NAME2
//*  END OF PROCEDURE
```

The First Data-Gridding Program

Data gridding is a two-step process with the data gridded in only one direction in the first step. This program takes corrected digitizer data that defines a curved line and replaces the data in one direction, so that one coordinate falls along each 30-second (5'30", 6'30", and so forth) line of either latitude or longitude. This respacing is done by using linear interpolation between input data points. The user specifies gridding in the latitude or the longitude direction.

Input from this program is compatible with output from the Digitizer Data-Conversion Program, and output from this program is compatible with the Second Data-Gridding Program. A method of obtaining data-base input from a contour map of a parameter would be:

1. Digitize the map of the parameter, using a digitizer operating in an incremented-mapping mode.
2. Process the data from step 1 with the Digitizer Data-Conversion Program to convert X-Y coordinates to latitude-longitude coordinates.
3. Process the output from step 2 with this program to aline the data along 30-second latitude lines.
4. Sort the output from this program and process with the Second Data-Gridding Program to produce data-base input cards.

Input

Parameter card (read from logical unit 5):

Col 1-10	Logical-unit number for input data (right justified)
Col 11-20	Logical-unit number for output data (right justified)
Col 21-26	Blank
Col 27-30	Grid-direction indicator: LAT (even spacing in the latitude direction) or LONG (even spacing in the longitude direction). This parameter needs to be left justified.

Input data (read from the logical unit defined on the parameter card):

Col 1-40	Blank
Col 41-47	Latitude, in degrees, minutes, and seconds (right justified)
Col 48-51	Blank
Col 52-58	Longitude, in degrees, minutes, and seconds (right justified)
Col 59-60	Blank
Col 61-68	Z-coordinate (alphanumeric)

Output

Output data (written to unit defined on the parameter card):

Col 1-7	Latitude, in degrees, minutes, and seconds
Col 8-11	Blank
Col 12-18	Longitude, in degrees, minutes, and seconds
Col 19-20	Blank
Col 21-28	Z-coordinate (alphanumeric)

Sample Procedure

A procedure can be used to execute this program with the procedure name LINDAT1. To execute the procedure, the following JCL would be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//   EXEC  LINDAT1,NAME1=input,NAME2=output
//SYSIN  DD  *
     parameter card
//
```

where *input* is the data-set name of the input-data file; *output* is the data-set name of the output-data file; and the procedure LINDAT1 would appear as follows:

```
//LINDAT1 PROC UNIT1='3330-1',VOL1=myvol,NAME1=NULLFILE,
//         VOL2=myvol,NAME2=NULLFILE,UNIT2='3330-1'
//**************************************************************************
//*LINDAT1 -- A PROCEDURE TO RUN THE FIRST DATA GRIDDING PROGRAM
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES):
//*
//*         UNIT1      INPUT POINT DATA DEVICE (3330-1)
//*         VOL1       INPUT POINT DATA VOLUME (myvol)
//*         NAME1      INPUT POINT DATA DSNAME (NULLFILE)
//*         UNIT2      CONVERTED DATA OUTPUT DEVICE (3330-1)
//*         VOL2       CONVERTED DATA OUTPUT VOLUME (myvol)
//*         NAME2      CONVERTED DATA OUTPUT DSNAME (NULLFILE)
//*
//***************************************************************************
//*           THE REGION PARAMETER MAY NOT INCLUDE ENOUGH I/O BUFFERS
//LINDAT1  EXEC  PGM=LINDAT1,TIME=2,REGION=52K
//*    LIBRARY IDENTIFICATION
//STEPLIB  DD  UNIT=3330-1,VOL=SER=myvol,DISP=SHR,
//          DSNAME=mylib
//*    CARD READER IS LOGICAL UNIT 5
//FT05F001  DD  DDNAME=SYSIN
//*    PRINTER IS LOGICAL UNIT 6
//FT06F001  DD  SYSOUT=A
//*    PUNCH IS LOGICAL UNIT 7
//FT07F001  DD  SYSOUT=B
//*    INPUT UNIT FOR POINT DATA -- LOGICAL UNIT 10 ASSUMED
//FT10F001  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*    OUTPUT UNIT FOR CONVERTED DATA -- LOGICAL UNIT 11 ASSUMED
//FT11F001  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(NEW,KEEP,DELETE),
//  DCB=(RECFM=FB,LRECL=28,BLKSIZE=2800),SPACE=(TRK,(5,5),RLSE),
//  DSNAME=&NAME2
//*   END OF PROCEDURE
```

The Second Data-Gridding Program

This program processes data produced by the First Data-Gridding Program by gridding the data in the second direction and producing card images compatible with the Load Program. This interpolation is done by fitting a cubic-spline function to each 30-second line of data produced by the First Data Gridding Program. A spline function is a mathematical analogy to the draftsman's mechanical spline. This function fits all the original data exactly, has continuous first and second derivatives, and is the smoothest function that interpolates the data. This technique works best when the 30-second lines are approximately perpendicular to the general trend of the original contours. Hence, the First Data-Gridding Program should grid the data in the direction approximately parallel to the contours; this program will complete the process by gridding the data in the direction perpendicular to the contours.

Input

Parameter card (read from logical unit 5):

Col 1-10	Logical-unit number for data-base format cards
Col 11-20	Logical-unit number for data produced by the First Data-Gridding Program and subsequently sorted
Col 21-30	Logical-unit number for messages
Col 31-40	Logical-unit number for card-image output compatible with the Load Program

Data-base input cards (read from logical unit defined on parameter card): The format of these cards is described in detail in the Load Program section. Those formats need to be followed exactly.

The first three cards in this input set need to be a 001* card, a 002* card, and a 003* card. These may be followed optionally by as many sets of 050* cards as desired. The last 050* card or the 003* card needs to be followed by a 100* card. This first 100* card defines a strip that is 3-degrees high and as wide as the study area. All subsequent processing needs to use only this 3-degree strip.

The 100* card is followed by several 200* cards that define 1-degree blocks for which data are to be produced. Each 200* card is in turn followed by a series of 300* cards that define 10-minute blocks for which data are to be produced. For each 300* card, a set of 401*-410* cards are produced that define the 1-minute values. This entire sequence may then be followed by a 100* card, and the process begins again. A 999* card is the last set in the series. A 999* card terminates the series of data-base input. A similar program is used to generate 10-minute values. The input to this program is identical, except that 300* cards are not placed in the input file.

Half-gridded data input (read from logical unit defined above): These data are in the output format of the First Data-Gridding Program. This data set needs to be sorted in order by latitude and longitude.

Output

Data base output (placed on the logical unit specified on parameter card): This file consists of a copy of the data base input cards with the 401*-410* cards inserted in the proper place. This file should then be ready to pass to the Edit Program and the Load Program. If the version of the program used to generate 10-minute data is used, the 350* cards are inserted.

The message file output (placed on logical unit defined on parameter card): This file consists of a listing of the parameter cards plus any diagnostic or informative messages. All these messages should be self-explanatory.

Contour violation file (placed on logical unit 9): This program checks generated values against adjacent contour values. If the generated values violate these contours, a diagnostic message is generated. A dump of pertinent auxiliary information also is generated at the end of the run. On a slope, a 5-foot violation is allowed without any diagnostic messages. On hills and in valleys, no such allowance is made. For hills and valleys, the contour interval for this checking is estimated from a nearby slope.

Return codes: The program will generate a non-zero completion code if an abnormal termination occurs. These are listed below:

Completion code	Meaning
1900	Error on I/O units card
1905	Empty parameter file
1910	Illegal data-base card
1915	Missing 100* card
1920	Illegal density value on 002* card
1940	Missing 200* card
1950	No 300* card after a 200* card
1999	Tried to enter part of code not written

Sample Procedure

A procedure can be used to execute this program with the procedure name SPLINT. To execute the procedure, the following JCL would be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  SPLINT,NAME1=input,NAME2=output,NAME3=data,
//  NAME4=dump
//SYSIN  DD  *
        parameter card
//
```

where *input* is the data-set name of the disk file containing data-base input cards; *output* is the data-set name of the data-base output cards disk file; *data* is the data-set name of the sorted digitizer data-disk file; and *dump* is the data-set name of contour-violation disk file. The procedure SPLINT could appear as follows:

```
//SPLINT  PROC  UNIT1='3330-1',VOL1=myvol,NAME1=NULLFILE,UNIT2='3330-1',
//        VOL2=myvol,NAME2=NULLFILE,UNIT3='3330-1',VOL3=myvol,
//        NAME3=NULLFILE,NAME4=NULLFILE
//*********************************************************************
//*SPLINT -- A PROCEDURE TO RUN THE SECOND DATA GRIDDING PROGRAM
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES):
//*
//*         UNIT1     DATA BASE INPUT CARDS DEVICE (3330-1)
//*         VOL1      DATA BASE INPUT CARDS VOLUME (myvol)
//*         NAME1     DATA BASE INPUT CARDS DSNAME (NULLFILE)
//*         UNIT2     DATA BASE OUTPUT CARDS DEVICE (3330-1)
//*         VOL2      DATA BASE OUTPUT CARDS VOLUME (myvol)
//*         NAME2     DATA BASE OUTPUT CARDS DSNAME (NULLFILE)
//*         UNIT3     ORDERED DIGITIZER DATA INPUT FILE UNIT (3330-1)
//*         VOL3      ORDERED DIGITIZER DATA INPUT FILE VOLUME (myvol)
//*         NAME3     ORDERED DIGITIZER DATA INPUT FILE DSNAME (NULLFILE)
//*         NAME4     INTERPOLATION ERROR DUMP FILE DSNAME (NULLFILE)
//*
//*********************************************************************
//*           THE REGION PARAMETER MAY NOT INCLUDE ENOUGH I/O BUFFERS
//    EXEC  PGM=SPLINT,TIME=1,REGION=430K
//*   LIBRARY IDENTIFICATION
//STEPLIB  DD  UNIT=3330-1,VOL=SER=myvol,DISP=SHR,
//         DSNAME=mylib
//*   CARD READER IS LOGICAL UNIT 5
//FT05F001  DD  DDNAME=SYSIN
//*   PRINTER IS LOGICAL UNIT 6
//FT06F001  DD  SYSOUT=A
//*   PUNCH IS LOGICAL UNIT 7
//FT07F001  DD  SYSOUT=B
//*   DUMP FILE FOR INTERPOLATION ERRORS IS ON LOGICAL UNIT 9
//*     PUT ON A TEMPORARY DISK FILE
//FT09F001  DD  UNIT=SYSDK,DISP=(NEW,PASS),SPACE=(CYL,(5,5),RLSE),
//    DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400),DSNAME=&NAME4
//*   INPUT UNIT FOR DATA BASE CARDS -- LOGICAL UNIT 10 ASSUMED
//FT10F001  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*   OUTPUT UNIT FOR DATA BASE CARDS - LOGICAL UNIT 11 ASSUMED
//FT11F001  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(NEW,KEEP,DELETE),
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400),SPACE=(TRK,(5,5),RLSE),
//  DSNAME=&NAME2
//*  INPUT UNIT FOR ORDERED DIGITIZER DATA -- LOGICAL UNIT 12 ASSUMED
//FT12F001  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DISP=SHR,DSNAME=&NAME3
//*   END OF PROCEDURE
```

The Boundary-Matrix Generation Program

This program is a special purpose program that generates a data-base parameter that defines the extent of the study area. This parameter is required by the Excess-Data Elimination Program. This data-base parameter is given the value 0 if the grid point is outside the boundary; it is given the value 1 if the grid point is inside the boundary.

This program will operate on a band of latitude that is 3 degrees high, and a band of longitude that is as much as 12 degrees wide. The program assumes that input data are alined along 30-second lines of latitude and are sorted in ascending order by latitude and longitude. The program further assumes that the east and west limits of the area of consideration are not in the study area, and issues a warning if the boundary does not cross each 30-second line an even number of times. The program processes the data from east to west, and keeps track of the number of times the 30-second line crosses the boundary. If at any point, the boundary has been crossed an odd number of times, the point is inside the study area.

Input

Parameter card (read on logical unit 5):

Col 1-10	Latitude-longitude limit card and sequence-number card logical unit (right jusitified)
Col 11-20	Output-messages logical unit (right justified)
Col 21-30	Input-data logical unit (right justified)
Col 31-40	Output-data logical unit (right justified)

Latitude-longitude limit card (read on the logical unit defined on the parameter card):

Col 1-10	Minimum latitude to be considered - degrees only (right justified)
Col 11-20	Minimum longitude to be considered - degrees only (right justified)
Col 21-30	Maximum latitude to be considered - degrees only (right justified)
Col 31-40	Maximum longitude to be considered - degrees only (right justified)

Sequence-number card (read on the logical unit defined on the parameter card):

Col 1-2	Blank
Col 3-10	Last sequence number used

Input data set (read on the logical unit defined on the parameter card): This data set would be produced by digitizing the boundary for the strip of interest, processing the data through the Digitizer Data-Conversion Program and the First Data-Gridding Program (operating in the latitude direction) and sorting the resulting data set by latitude and longitude using any available sorting program. The final data set describes the location of the boundary at 30-second intervals of latitude under the format:

Col 1-7	Latitude, in degrees, minutes, and seconds
Col 8-11	Blank
Col 12-18	Longitude, in degrees, minutes, and seconds

Output

A number of informative messages are produced on the logical unit defined on the parameter card. These messages, which need to be examined carefully, are fully self-explanatory.

The output data set, which is produced on the logical unit defined on the parameter card, consists of 100*, 200*, 300*, and 401*-410* cards. The format of these cards is defined in the documentation of the Load Program. The data values on the 401*-410* cards are set to 1 if the 1-minute block is in the aquifer; it is set to 0 if the block is outside of the aquifer.

Sample Procedure

A procedure can be used to execute this program with the procedure name BNDGEN. To execute the procedure, the following JCL would be used:

```
// . . . JOB . . .
//PROCLIB DD DSN=procedure.library,DISP=SHR
//   EXEC  BNDGEN,NAME1=input,NAME2=output
//SYSIN  DD *
     parameter card
//
```

where *input* is the data-set name of the input disk file; and *output* is the data-set name of the output disk file. The procedure BNDGEN could appear as follows:

```
//BNDGEN   PROC   UNIT1='3330-1',VOL1=myvol,NAME1=NULLFILE,UNIT2='3330-1',
//       VOL2=myvol,NAME2=NULLFILE
//**************************************************************************
//*BNDGEN -- A PROCEDURE TO RUN THE BOUNDARY MATRIX GENERATION PROGRAM
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES):
//*
//*         UNIT1      BOUNDARY INPUT DATA DEVICE (3330-1)
//*         VOL1       BOUNDARY INPUT DATA VOLUME (myvol)
//*         NAME1      BOUNDARY INPUT DATA DSNAME (NULLFILE)
//*         UNIT2      BOUNDARY OUTPUT DATA DEVICE (3330-1)
//*         VOL2       BOUNDARY OUTPUT DATA VOLUME (myvol)
//*         NAME2      BOUNDARY OUTPUT DATA DSNAME (NULLFILE)
//*
//**************************************************************************
//*            THE REGION PARAMETER MAY NOT INCLUDE ENOUGH I/O BUFFERS
//BNDGEN  EXEC  PGM=BNDGEN,TIME=1,REGION=300K
//*  LIBRARY IDENTIFICATION
//STEPLIB  DD  UNIT1=3330-1,VOL=SER=myvol,DISP=SHR,
//        DSNAME=mylib
//*  CARD READER IS LOGICAL UNIT 5
//FT05F001  DD  DDNAME=SYSIN
//*  PRINTER IS LOGICAL UNIT 6
//FT06F001  DD  SYSOUT=A
//*  PUNCH IS LOGICAL UNIT 7
//FT07F001  DD  SYSOUT=B
//*  INPUT UNIT -- LOGICAL UNIT 10 ASSUMED
//FT10F001  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*  OUTPUT UNIT -- LOGICAL UNIT 11 ASSUMED
//FT11F001  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(NEW,KEEP,DELETE),
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400),SPACE=(CYL,(5,5),RLSE),
//  DSNAME=&NAME2
//*  END OF PROCEDURE
```

The Excess-Data Elimination Program

The purpose of this program is to insert the missing-value indicator (MVI) at all points in a 1-minute, data deck, where the study area is absent. The MVI is an arbitrary value used to indicate that no value is available for this point. The input to this program is a 1-minute data deck, and the output is an identical data deck, except the MVI is inserted where appropriate.

There are three ways in which this program can be run. The first type of run, called a BULD, reads the boundary matrix for a 3-degree strip of the data base, and produces three logical matrices for subsequent runs. These matrices are stored on the disk file. In a BULD, the location of the study area also is plotted by 1-degree blocks on logical unit 20. For this reason, unit 20 needs to be specified as an 8-lines-per-inch print file, and the job card needs to be altered to allow 84 lines per page.

The second type of run is a FIX. This type of run is used to fix a boundary matrix that may have one or more errors in it. In this type of run, the location of the start of a fix is given, and the value of the boundary matrix is switched, starting at that point to the end of the row in which that point is located. This type of run will fix errors produced either by digitizing or problems in the Boundary-Matrix Generation Program. In a FIX, multiple fixes can be specified, and the fixes are cumulative in nature. Hence, a second fix in a row defines the end of the first fix, and the third fix in a row would define the beginning of a new fix.

The third type of run, called a RUN, takes the matrices prepared in a BULD, reads the input matrix, and inserts the MVI at appropriate places. The result of this type of run is an altered 1-minute data set that is ready for the Edit or Load Program.

The usual manner of using this program would be to first do a BULD and examine the output. If the plot of the boundary matrix looks correct, a RUN would be made. If the boundary has errors, then a FIX would be made. The boundary matrix is also plotted during a FIX, and a FIX can be repeated until the boundary matrix is acceptable.

This program will operate on a band of latitude that is as much as 3 degrees high and a band of longitude as much as 12 degrees wide. This restriction is necessary, because all boundary data are kept in memory, and handling a 3-degree by 12-degree area requires more than 500K bytes of storage.

Input

Parameter Card (read from logical unit 5):

Col 1-10	Input file logical-unit number (right justified)
Col 11-20	Output file logical-unit number (right justified)
Col 21-30	Control file logical-unit number (right justified)
Col 31-40	Plot file logical-unit number (right justified)
Col 41-50	MVI; Integer of five or less digits (right justified)
Col 51-56	Blank
Col 57-60	Type of Run (left justified): BULD, RUN, or FIX are the only acceptable values.

For a BULD, the boundary matrix is read on the input file and a control file is produced. For a RUN, the input file consists of data-base input cards (described in the Load Program documentation), and the output file consists of data-base input cards with a MVI inserted on the 401*-410* cards. For a FIX, the original boundary matrix is read on the input file, and the corrected boundary matrix is written to the output file. For a FIX, the control file contains the fix cards. The fix cards, which describe the starting (or ending) point of a fix, have the latitude and longitude of the fix in the same places as the 300* card, and the rest of the card is blank.

Sample Procedure

A procedure can be used to execute this program with the procedure name ELIM1. To execute the procedure, the following JCL would be used:

```
// . . . JOB . . .
//PROCLIB DD DSN=procedure.library,DISP=SHR
//   EXEC  ELIM1,NAME1=input,NAME2=output,NAME3=cntl
//SYSIN  DD *
        parameter card
//
```

where *input* is the data-set name of the input disk file; *output* is the data-set name of the output disk file; and *cntl* is the data-set name of the control disk file. The procedure ELIM1 could appear as follows:

```
//ELIM1  PROC  UNIT1='3330-1',VOL1=,NAME1=NULLFILE,UNIT2='3330-1',
//       VOL2=myvol,NAME2=NULLFILE,UNIT3='3330-1',VOL3=myvol,
//       NAME3=NULLFILE
//**********************************************************************
//*ELIM1 -- A PROCEDURE TO RUN THE EXCESS DATA ELIMINATION PROGRAM
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES):
//*
//*        UNIT1      DATA BASE INPUT CARDS DEVICE (3330-1)
//*        VOL1       DATA BASE INPUT CARDS VOLUME (myvol)
//*        NAME1      DATA BASE INPUT CARDS DSNAME (NULLFILE)
//*        UNIT2      DATA BASE OUTPUT CARDS DEVICE (3330-1)
//*        VOL2       DATA BASE OUTPUT CARDS VOLUME (myvol)
//*        NAME2      DATA BASE OUTPUT CARDS DSNAME (NULLFILE)
//*        UNIT3      CONTROL DATA INPUT FILE UNIT (3330-1)
//*        VOL3       CONTROL DATA INPUT FILE VOLUME (myvol)
//*        NAME3      CONTROL DATA INPUT FILE DSNAME (NULLFILE)
//*
//**********************************************************************
```

```
//*            THE REGION PARAMETER MAY NOT INCLUDE ENOUGH I/O BUFFERS
//ELIM1  EXEC  PGM=ELIM1,TIME=1,REGION=430K
//*   LIBRARY IDENTIFICATION
//STEPLIB  DD  UNIT=3330-1,VOL=myvol,DISP=SHR,
//         DSNAME=mylib
//*   CARD READER IS LOGICAL UNIT 5
//FT05F001  DD  DDNAME=SYSIN
//*   PRINTER IS LOGICAL UNIT 6
//FT06F001  DD  SYSOUT=A
//*   PUNCH IS LOGICAL UNIT 7
//FT07F001  DD  SYSOUT=B
//*   INPUT UNIT -- LOGICAL UNIT 10 ASSUMED
//FT10F001  DD  UNIT=&UNIT1,VOL=SER=&VOL1,DISP=SHR,DSNAME=&NAME1
//*   OUTPUT UNIT -- LOGICAL UNIT 11 ASSUMED
//FT11F001  DD  UNIT=&UNIT2,VOL=SER=&VOL2,DISP=(NEW,KEEP,DELETE),
//  DCB=(RECFM=FB,LRECL=80,BLKSIZE=6400),SPACE=(TRK,(5,5),RLSE),
//  DSNAME=&NAME2
//*   CONTROL UNIT -- LOGICAL UNIT 12 ASSUMED
//FT12F001  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DISP=SHR,DSNAME=&NAME3
//*   END OF PROCEDURE
```

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 program is to check the syntax of input records for these programs. The program also can perform other types of checks. These additional edits 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*, 350*, 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 1-degree blocks are grouped with the correct 3-degree block, and that the 10-minute blocks are grouped with the correct 1-degree block. The program determines if the statistical components are only given values on the records corresponding to the lowest level at which the data are stored. The program can optiontally 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 3-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 3-degree block), the program determines if that block of data has previously been stored in the data base. If the 3-degree block is already loaded into the data base, the program prints a message and aborts the program 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 valid list of parameter names. The person's name, on the 599* record, is checked against a list of valid 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 or not. 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 record is called the parameter card. Information on this card guides execution of the Edit Program. The parameter card supplies information that is needed to access the data base, including data-base name, data-base password, and the study unit. The parameter card 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 behind the parameter card, 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 card. The parameter card also contains the output-unit device number (which will be described in the section on output).

The parameter card 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 card 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	Records for Move Program
	6	Records for Edit Program
5	--	Unused
6	--	Unused
7	--	Unused
8	--	Unused
9	--	Unused
10	--	Unused

51-52	-	2X	Blank
53-60	AQNAME	2A4	Study unit for this data set [HIPLAINS for High Plains RASA (HPRASA)]
61-68	DBNAME	2A4	Data-base name (AQUIFER for HPRASA)
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	LODE	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 severely 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 (3-degree load record) is incorrect, it would not be possible to determine if the subsequent 200* records (1-degree load record) actually represent 1-degree blocks within the 3-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 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 Edit procedure 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. To run the Edit Program with both input and output disk files, the following JCL would be used:

```
    // . . . JOB . . .
    //PROCLIB  DD  DSN=procedure.library,DISP=SHR
    //STEPNAME  EXEC  EDIT,NAME1=input,NAME2=output
    //GOED.SYSIN  DD  *
         parameter card
    //
```

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. 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  EDIT
    //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=6400),SPACE=(TRK,(6,6),RLSE)
    //GOED.SYSIN  DD  *
      parameter card for Edit Program
    //STEP2  EXEC  RETREV,NAME1='output'
    //GORT.FT10F001  DD  DSN=&&EDOUT,DISP=(OLD,DELETE)
    //GORT.SYSIN  DD  *
       parameter card for Retrieval Program
    //
```

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

```
//EDIT PROC TIMEG=2,REG=290K,UNIT1='3330-1',VOL1=myvol,NAME1=NULLFILE,
//          UNIT2='3330-1',VOL2=myvol,NAME2=NULLFILE,SP1=1,SP2=10,
//          PROG=EDITHX
//*********************************************************************
//*  EDIT: A PROCEDURE TO RUN THE HPRASA DATA BASE EDIT PROGRAM
//*        WRITTEN BY CARMELO F. FERRIGNO NOVEMBER, 1980
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES)
//*
//*     TIMEG  TOTAL RUN TIME (2)
//*     REG    REGION SIZE (290K)
//*     UNIT1  EDIT PROGRAM INPUT CARDS DEVICE (3330-1)
//*     VOL1   EDIT PROGRAM INPUT CARDS VOLUME (myvol)
//*     NAME1  EDIT PROGRAM INPUT CARDS DSNAME (NULLFILE)
//*     UNIT2  EDIT PROGRAM OUTPUT CARDS DEVICE (3330-1)
//*     VOL2   EDIT PROGRAM OUTPUT CARDS VOLUME (myvol)
//*     NAME2  EDIT PROGRAM OUTPUT CARDS DSNAME (NULLFILE)
//*     PROG   NAME OF LOAD MODULE TO BE EXECUTED (EDITHX)
//*
//**********************************************************************
//GOED EXEC PGM=&PROG,TIME=&TIMEG,REGION=&REG
//STEPLIB  DD  DSN=mylib,DISP=SHR,VOL=SER=myvol,UNIT=3330-1
//         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=6400),
//             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 USERS' AND PARAMETER 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=133,BLKSIZE=1330)
//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))
//AQUIFER1  DD DSN=databasefile1,DISP=SHR
//AQUIFER2  DD DSN=databasefile2,DISP=SHR
//AQUIFER3  DD DSN=databasefile3,DISP=SHR
//AQUIFER4  DD DSN=databasefile4,DISP=SHR
//AQUIFER5  DD DSN=databasefile5,DISP=SHR
//AQUIFER6  DD DSN=databasefile6,DISP=SHR
```

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 1-minute 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 3-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, 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 procedure is to save the data base on magnetic tape; then, if necessary, the data base can be restored to its former condition, prior to the aborted attempt to load data.

Input

The first input record is the parameter card. The format of this card is found in the documentation for the Edit Program. The only fields on this card 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 on the parameter card 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 for the parameter. 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 at a 3-degree density has one value for each 3-degree block that is partially or completely within the study area. A 3-degree block is identified by the record type 100*. The 100* record contains the latitude and longitude of the southeast corner of the 3-degree block, as well as the value (average). 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 entire 3-degree block. The number of 100* records is equivalent to the number of 3-degree blocks partially or completely within the study area.

A parameter stored in the data base at a 1-degree density has one value for each 1-degree block within the study area. A 1-degree block is identified by the record type 200*. The 200* record contains the latitude and longitude of the southeast corner of the 1-degree block as well as the value. The record may also optionally contain the minimum, the maximum, the number of values, and the standard deviation. In the input data set, 1-degree blocks are grouped within 3-degree blocks. The 3-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 at a 10-minute density has one value for each 10-minute block within the study area. The 10-minute block is identified by the record type 350*. 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 10-minute block represents the base level for the parameter, all statistical components on the 350* card can be valued, but only the value component is required. The 10-minute blocks are grouped within 1-degree blocks, which in turn, are grouped within 3-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 at a 1-minute density has one value for each 1-minute block within the study area. The format of this data is slightly different from the preceding data. A 10-minute block is represented by a 300* record that contains the latitude and longitude of the southeast corner of the block. The 300* record is followed by ten records, types 401* through 410*, that contain values for the 100 1-minute blocks within the 10-minute block. These ten records may be considered a ten-by-ten matrix covering the 10-minute block. The 401* record contains ten values for the northern-most row of 1-minute blocks. Within a row, values are placed from west to east. If a 10-minute block is located along the boundary of the study area, some of the 1-minute cells may be outside the area. These cells are not given a valid value, but instead are given a value called the MVI. This value, which is found on the 003* record, indicates that a particular point has no valid value. Only 1-minute blocks that have valid values are used in computing statistics. As with 10-minute data, the 10-minute blocks are grouped under the corresponding 1-degree blocks, which in turn, are grouped with the proper 3-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	K	I9	Reserved Value A
50-60	C22	K	F11.3	Reserved Value B
61-70	C23	K	2A4,A2	Reserved Value C
71-72	-	-	2X	Blank
73-80	INSEQ	-	I8	Sequence Number, non-decreasing

002*: Second master record for the Load Program

01-04	TYPE	-	A4	002*
05-10	-	-	6X	Blank
11-18	C2	K	2A4	Study unit
19-48	C3	K	7A4,A2	Parameter Name
49	C5	NK	I1	Density (1=3-degree, 2=1-degree, 3=10-minute, 4=1-minute)
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, non-decreasing

003*: Third master record for the Load Program

Columns	Component or variable	Key or non-key	Format	Content
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, non-decreasing

050*: Remarks records

Columns	Component or variable	Key or non-key	Format	Content
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 card number 1 = First card of set 2 = Second card of set 3 = Third card of set 4 = Fourth card of set 5 = Fifth card of set
23-72	TEXT	NK	12A4,A2	Text of remark
73-80	INSEQ	-	I8	Sequence Number, non-decreasing

Note: The 050* Records must come in sets of five.

100*: Three-degree block record

Columns	Component or variable	Key or non-key	Format	Content
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, non-decreasing

200*: One-degree block 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, non-decreasing

350*: Ten-minute block record--10-minute data

01-04	TYPE	-	A4	350*
05-10	-	-	6X	Blank
11-16	C351	NK	I6	Latitude (DDMMSS)
17	-	-	1X	Blank
18-24	C352	NK	I7	Longitude (DDDMMSS)
25	-	-	1X	Blank
26-30	C361	NK	I5	Minimum
31-35	C362	NK	I5	Maximum
36-40	C363	NK	I5	Average (Value)
41-45	C364	NK	I5	Number of values
46-53	C365	NK	F8.2	Standard Deviation
54-72	-	-	19X	Blank
73-80	INSEQ	-	I8	Sequence Number, non-decreasing

300*: Ten-minute block record--1-minute data

Columns	Component or variable	Key or non-key	Format	Content
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	Average (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, non-decreasing

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

401*-410*: One-minute block 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, non-decreasing

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

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 3-degree blocks that were loaded into the data base and a graphic picture that indicates location of the 1-degree blocks within the study area.

Sample Procedure

To execute the Load Program, a procedure like the sample below could be used. A data set called DATAIN contains the input data. This data set consists of the parameter card, 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 card 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, 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  DBLOAD
//GO.DATAIN  DD  *
     parameter card and data records
//
```

The second card describes the magnetic tape used to save the data base prior to loading any new data. The procedure DBLOAD could appear as follows:

```
//DBLOAD PROC REG=780K,TIMEG=2,SP1=1,SP2=10
//* * *                  EXECUTE 'DBLOAD' PROGRAM                 * * *
//* PROCEDURE DEVELOPED BY RICHARD T. BLACKBURN, USGS, RASA, JULY 1980,
//* FOR PL/I EXECUTION OF THE PL/I OPTIMIZER COMPILED AND LINK EDITED
//* PROGRAM 'DBLOAD' - INVOKE THE PROCEDURE WITH
//* THE FOLLOWING:                    MY COMMENTS: (DO NOT CODE THESE)
//* //WHATEVER JOB (ACCT.,ETC.)       -YOUR JOB & ACCOUNT INFORMATION.
//* //PROCLIB  DD  DSN=procedure.library,DISP=SHR -IDENTIFIES PROC LIB
//* //STEP1   EXEC DBLOAD        -THE PROC WILL RUN PGM DBLOAD.
//* //GO.DATAIN DD *
//* ------YOUR 'EDIT' VERIFIED DATA GOES HERE, OR USE
//* //GO.DATAIN DD DSN=?????? TO IDENTIFY DATA SET CONTAINING DATA.
//* /*                                 -END OF DATA INPUT.
//* //
//* * *                                                           * * *
```

```
//GO     EXEC PGM=DBLOAD,REGION=&REG,TIME=&TIMEG,COND=(12,LE),
//       PARM='ISA(16K)'
//STEPLIB  DD DSN=mylib,VOL=SER=myvol,UNIT=3330-1,DISP=SHR
//          DD DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//          DD DSN=SYS1.S2K,DISP=SHR                SYSTEM 2000 LIBRARIES
//SYSLIB   DD DSN=SYS1.S2K,DISP=SHR
//          DD DSN=SYS1.PLIBASE,DISP=SHR
//SYSPRINT DD SYSOUT=A                                    PRINTED OUTPUT
//PLIDUMP  DD DUMMY
//ERRMSG   DD DSN=errorfile,DISP=SHR
//S2KMSG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//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))      LOCATE FILE, IF NEEDED
//LOCATE01 DD UNIT=SYSDK,SPACE=(CYL,(1,1))      LOCATE FILE, IF NEEDED
//LOCATE02 DD UNIT=SYSDK,SPACE=(CYL,(1,1))      LOCATE FILE, IF NEEDED
//***  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:
//AQUIFER1 DD DSN=databasefile1,DISP=SHR
//AQUIFER2 DD DSN=databasefile2,DISP=SHR
//AQUIFER3 DD DSN=databasefile3,DISP=SHR
//AQUIFER4 DD DSN=databasefile4,DISP=SHR
//AQUIFER5 DD DSN=databasefile5,DISP=SHR
//AQUIFER6 DD DSN=databasefile6,DISP=SHR
//***  THE FOLLOWING FILE IS FOR 'SAVE AQUIFER' COMMAND   ***
//TAPES2K  DD DSN=DBBACKUP,UNIT=(TAPE62,,DEFER),
//          VOL=(PRIVATE,RETAIN,SER=tapenum),DISP=(OLD,PASS)
//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 aquifer unit; (2) the parameter name; (3) the description 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, a 10-minute parameter has its primary data stored at level 3 of the data-base structure. In order to retrieve the 1-degree block statistics, the retrieval level should be set to 2. Retrieved data would then consist of 100* and 200* records, where the 200* records (1-degree records) contain the statistics computed from values for the 10-minute blocks that are within the 1-degree block.

The user can optionally request the inclusion of test proposals in the retrieved data. This is done through use of tne 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 test proposals, the test value will replace the currently accepted value in the output. If a 1-minute parameter is being retrieved, then for each 10-minute block (which represents 100 1-minute blocks), the currently accepted 1-minute values are replaced only if the 1-minute block has a corresponding test proposal. If more than one test proposal is located for a block, the latest test proposal is output. For 1-minute 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:

Card (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 (HIPLAINS for HPRASA)	
Col 61-68	DBNAME	2A4	Data-base name (AQUIFER for HPRASA)	
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	

Card (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 (HIPLAINS for HPRASA)
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	Sequence number

Card (3) - 006* - Read from same unit as Card (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	Sequence number

Card (4) - 999* - Read from same unit as Card (2)

Col 1-4	TYPE	A4	999*
Col 5-72	-	68X	Blank
Col 73-80	INSEQ	I8	Sequence number

There are several restrictions on the order of these records. Each 005* card must be followed by at least one 006* card. If this does not occur, the program will be aborted. An 006* card cannot exist without a corresponding 005* card. Retrieval requests can contain more than one 005* card, providing each 005* card is followed by one or more 006* cards. 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* card is required; this should be the last record.

Output

The output consists of:

(1) Messages - written to unit 6 (standard printer output file);
(2) Retrieved data - written to unit defined on parameter record (usually a disk file);
(3) Error messages - written to unit 21 (auxiliary printer output 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 mistake can be corrected by the 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 sixth through 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

To run the program, the following JCL could be used:

```
// . . . JOB . . .                                -Job card
//PROCLIB  DD  DSN=procedure.library,DISP=SHR     -Procedure Library
// EXEC RETREV,NAME1=output                       -Step Card
//GORT.SYSIN  DD  *
    parameter card
    retrieval request records
//
```

The procedure RETREV could appear as follows:

```
//RETREV PROC TIMEG=2,REG=740K,VOL1=myvol,NAME1=NULLFILE,
//       SP1=1,SP2=10,PROG=RETREV,UNIT1='3330-1'
//**********************************************************************
//*  RETREV: A PROCEDURE TO RUN THE HPRASA DATA BASE RETRIEVAL PROGRAM
//*          WRITTEN BY CARMELO E. FERRIGNO, MARCH, 1980
//*
//*
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESES)
//*
//*     TIMEG  TOTAL RUN TIME (2)
//*     REG    REGION SIZE (740K)
//*     UNIT1  RETRIEVAL PROGRAM OUTPUT DEVICE (3330-1)
//*     VOL1   RETRIEVAL PROGRAM OUTPUT VOLUME (myvol)
//*     NAME1  RETRIEVAL PROGRAM OUTPUT DSNAME (NULLFILE)
//*     PROG   NAME OF LOAD MODULE TO BE EXECUTED (RETREV)
//*
//**********************************************************************
//GORT EXEC PGM=&PROG,TIME=&TIMEG,REGION=&REG
//STEPLIB  DD  DSN=mylib,DISP=SHR,VOL=SER=myvol,UNIT=3330-1
//         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=6400),
//             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=133,BLKSIZE=1330)
//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))
//AQUIFER1 DD DSN=databasefile1,DISP=SHR
//AQUIFER2 DD DSN=databasefile2,DISP=SHR
//AQUIFER3 DD DSN=databasefile3,DISP=SHR
//AQUIFER4 DD DSN=databasefile4,DISP=SHR
//AQUIFER5 DD DSN=databasefile5,DISP=SHR
//AQUIFER6 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 block and parameter, test data are loaded in the data base as part of the data belonging to the parameter, but the test data are kept separate from the other data. This separate area is temporary in nature; the accepted data for a particular parameter reside in the permanent area of the data base. The Retrieval Program may optionally retrieve the test proposals.

An example of the use of a test proposal 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 block within the retrieved area that has a corresponding test proposal, the test proposal would replace the currently 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, the user may decide that some of the test proposals are better than currently accepted values. The user may then want the test proposals to replace corresponding accepted values. This is also done using the Instant-Update Program. When a test proposal is marked as an accepted proposal, it 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 proposal 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 it is a test proposal, an accepted value, or a rejected value. Only one test proposal should be added per 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. 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 it is part of the permanent data.

The program limits the number of test proposals per 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 block is the MVI, a proposal is not allowed. Also, the test proposal 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 block that already exists in the permanent area of the data base.

Input

Card 1: This is the 002* card. 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 Name	2A4	Study unit (HIPLAINS for HPRASA)
Col 19-48	Parameter	7A4,A2	Parameter to be updated
Col 49-72	-	24X	Blank
Col 73-80	Sequence	I8	Non-decreasing sequence number

Card 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 card 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* card 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	Non-decreasing number

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

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

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

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

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

Col 1-4	Record Type	A4	600*
Col 5-10	-	6X	Blank
Col 11-16	Lat	I6	Latitude of proposal (DDMMSS)
Col 17	-	1X	Blank
Col 18-24	Lon	I7	Longitude of proposal (DDDMMSS)
Col 25	-	1X	Blank
Col 26-30	Min	I5	Minimum
Col 31-35	Max	I5	Maximum
Col 36-40	Value	I5	Test proposal
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	Non-decreasing sequence number

If the action code is 3 and the data are 1-minute data, the 600* card contains only the latitude and longitude of the 10-minute block. It is identical in format to version 1 of the 600* card. The 600* card is then followed by a group of cards that contain proposed values for some or all of the 100 1-minute points contained within the 10-minute block whose latitude and longitude is defined on the 600* card. These records are type 701* through 710*. The 10-minute block can be thought of as a 10 X 10 matrix of 1-minute values. The 701* card represents the first (north) row of the 10 X 10 matrix; the 704* card corresponds to the fourth row of the matrix; and so forth. Not all ten rows are required to have test proposals, 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* card should be followed by 705*, 706*, 707*, and 708* records. Within these rows, only place values in the columns where a test proposal is to be inserted. If the points 2, 3, 4, and 5 within a row are to have test proposals, then positions 1, 6, 7, 8, 9, and 10 should be blank. The formats of records 701* through 710* are identical:

Card 4: 1-minute data - Action code 3

Col 1-4	Record Type	A4	701* through 710*
Col 5-10	-	6X	Blank
Col 11-60	Values	10I5	Test proposals
Col 61-72	-	12X	Blank
Col 73-80	Sequence	I8	Non-decreasing sequence number

The series 600*, 701*-710* are repeated as many times as necessary.
Last card: The last card in the deck must be a 999*

Col 1-4	Record Type	A4	999*
Col 5-72	-	68X	Blank
Col 73-80	Sequence	I8	Non-decreasing sequence number

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

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

```
002*      HIPLAINSANNUAL LAKE EVAPORATION                                    1
599*      LUCKEY                3                                            2
600*      330000 1030000        720                                          3
600*      330000 1040000        730                                          4
600*      340000 1040000        740                                          5
999*                                                                         6
```

Each of the 600* records represents a 1-degree block within the 1-degree parameter ANNUAL LAKE EVAPORATION. If the parameter was stored as 10-minute data, each 600* record would represent a 10-minute block.

Example 2: Adding test proposals for a parameter stored as 1-minute data.

```
002*      HIPLAINSWATER TABLE - 1960                                         1
599*      LUCKEY                3                                            2
600*      315000 1013000                                                     3
701*       2520 2530 2549 2550                                               4
702*       2510 2515 2520 2560                                               5
703*       2570 2580 2590                                                    6
600*      315000 1020000                                                     7
703*                 2770 2761 2752 2743                                     8
704*                 2766 2757 2748 2739 2730                                9
705*           2770  2761 2753 2745 2736                                    10
600*      320000 1013000                                                    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 10-minute data block; each value on the 701*-710* records is a test proposal for a 1-minute point within the 10-minute block. The position of the 1-minute block is determined by the record type and the position on 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*        HIPLAINSANNUAL LAKE EVAPORATION                        1
599*        LUCKEY              1 083181  124536                   2
600*        330000 1030000                                         3
599*        LUCKEY              2 091581  073126                   4
600*        330000 1040000                                         5
599*        LUCKEY              3                                  6
600*        330000 1050000        650                              7
999*                                                               8
```

The first 599* card indicates that a test proposal that was added on the date 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* card 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 information print files. The first information 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 information file is called the activity report. It contains specific information on each proposal that is rejected, accepted, or added. For parameters stored as 1-degree or 10-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 parameter value
   6. Currently accepted parameter 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. Previously accepted parameter value
   6. Newly accepted parameter 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 parameter value
   6. Currently accepted parameter value

For parameters stored as 1-minute 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 10 X 10 matrix of the 100 parameter values within the 10-minute block. The rejected values are highlighted with asterisks. The remaining values represent the currently accepted values for those 1-minute 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 10 X 10 matrix of the 100 parameter values within the 10-minute block. The newly accepted values are highlighted with asterisks. The remaining values represent the currently accepted 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 10 X 10 matrix of the 100 parameter values within the 10-minute block. The new test values are highlighted with asterisks. The remaining values represent the currently accepted values for those points.

The activity report also contains information on update requests that could not be accomplished. For example, if a test proposal has a value 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 a 1-minute parameter, the message contains the position in the 100 1-minute 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 1-minute test proposals is not allowable, the test proposal for the entire block will not be added.

The third information 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 EDITIU. This procedure is a two-step process. The first step executes the Edit Program. The input to the Edit Program is a parameter card 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 cards 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 quite 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 EDITIU procedure, the following JCL could be used:

```
// . . . JOB . . .                                   -Job card
//PROCLIB  DD  DSN=procedure.library,DISP=SHR        -Procedure library
// EXEC  EDITIU,MEMBER=COCF266A                      -Step card
//GOED.SYSIN  DD  *                                  -Card input
     Parameter card
     Update request records
//
```

On the EXEC statement (third card), 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 card input, the Instant-Update Program request records are preceded by the Edit Program parameter card. This card contains information necessary to the proper execution of the Edit Program. The card is as follows:

Col 1-30	Title	7A4,A2		Title of run
Col 31-32	-	2X		Blank
Col 33-34	Input file	I2		Should be set to 5
Col 35-36	-	2X		Blank
Col 37-38	Output file	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 (HIPLAINS for HPRASA)
Col 61-68	Data base	2A4		Data base name (AQUIFER for HPRASA)
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 EDITIU could appear as follows:

```
//EDITIU PROC TIME1=1,TIME2=1,REG1=760K,REG2=770K,PROG1=EDITHX,
//            PROG2=IUPDTE,MEMBER=,SP1=1,SP2=10
//**********************************************************************
//*  EDITIU: A PROCEDURE TO RUN THE HPRASA INSTANT UPDATE PROGRAM
//*          IN CONJUNCTION WITH THE AQUIFER 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.
//*
//*          WRITTEN BY CARMELO F. FERRIGNO  SEPTEMBER,1981
//*
//*  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 (760K)
//*    REG2       REGION SIZE INSTANT UPDATE STEP (770K)
//*    PROG1      LOAD MODULE NAME EDIT STEP (EDITHX)
//*    PROG2      LOAD MODULE NAME INSTANT UPDATE STEP (IUPDTE)
//*    MEMBER     NAME OF PDS MEMBER FOR DBA INFORMATION FILE
//*
//********************************************************************
//*
//*   STEP 1: EDIT
//*
//GOED EXEC PGM=&PROG1,TIME=&TIM1,REGION=&REG1
//STEPLIB  DD  DSN=mylib,UNIT=3330-1,VOL=SER=myvol,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 USERS' AND PARAMETER 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=133,BLKSIZE=1330)
//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))
//AQUIFER1  DD DSN=databasefile1,DISP=SHR
//AQUIFER2  DD DSN=databasefile2,DISP=SHR
//AQUIFER3  DD DSN=databasefile3,DISP=SHR
//AQUIFER4  DD DSN=databasefile4,DISP=SHR
//AQUIFER5  DD DSN=databasefile5,DISP=SHR
//AQUIFER6  DD DSN=databasefile6,DISP=SHR
//*
//*
//IU EXEC PGM=&PROG2,TIME=&TIM2,REGION=&REG2,PARM='ISA(23K)',
//        COND=(13,LT,GOED)
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,UNIT=3330-1,VOL=SER=myvol,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),UNIT=3330-1,
//         VOL=SER=myvol,DISP=(OLD,KEEP)
//*
//*   DEFINE SYSTEM 2000 FILES
//*
//S2KMSG   DD  SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//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))
//AQUIFER1 DD DSN=databasefile1,DISP=SHR
//AQUIFER2 DD DSN=databasefile2,DISP=SHR
//AQUIFER3 DD DSN=databasefile3,DISP=SHR
//AQUIFER4 DD DSN=databasefile4,DISP=SHR
//AQUIFER5 DD DSN=databasefile5,DISP=SHR
//AQUIFER6 DD DSN=databasefile6,DISP=SHR
```

The Move Program

The Instant-Update Program gives users of the DMS an opportunity to propose changes to data 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 the accepted value; the Instant-Update Program changes the flag to indicate that the proposal contains the accepted value 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 structure 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 Instant-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. 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 card image contains a password having the proper update authority, the aquifer unit, and a code indicating the mode of operation. If the code is 1, all the data-base parameters are processed and this card is the only input card. If the code is 2, only selected parameters are processed and the parameter names are input. The format of card 1 is as follows:

Col 1-4	Data-base password, left-justified
Col 5-12	Study unit, left-justified (HIPLAINS for HPRASA)
Col 13	Operation code

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

Col 1-30	Parameter name, left-justified

If the code is 1, the parameter names are obtained from a file called PARMS, 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 proposals 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 processing. If this message is printed, the computer staff responsible for the program should be contacted.

Sample Procedure

A procedure called DBMOVE is used to execute the Move Program. To execute the program, the following JCL could be used:

```
// . . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//STEPNAME  EXEC  DBMOVE
//MOVE.DATAIN  DD  *
   input records
//
```

The procedure DBMOVE could appear as follows:

```
//DBMOVE  PROC  TIM=2,REG=840K,PROG=DBMOVE,SP1=1,SP2=10
//*************************************************************************
//*  DBMOVE: A PROCEDURE TO RUN THE HPRASA DATA BASE MOVE PROGRAM
//*          WRITTEN BY CARMELO F. FERRIGNO  OCTOBER,1981
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*   TIM     TOTAL RUN TIME (2)
//*   REG     REGION SIZE (840K)
//*   PROG    LOAD MODULE NAME (DBMOVE)
//*
//*************************************************************************
//MOVE  EXEC  PGM=&PROG,TIME=&TIM,REGION=&REG,PARM='ISA(20K)'
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
```

```
//          DD  DSN=mylib,UNIT=3330-1,VOL=SER=myvol,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=133,BLKSIZE=1330)
//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))
//AQUIFER1  DD DSN=databasefile1,DISP=SHR
//AQUIFER2  DD DSN=databasefile2,DISP=SHR
//AQUIFER3  DD DSN=databasefile3,DISP=SHR
//AQUIFER4  DD DSN=databasefile4,DISP=SHR
//AQUIFER5  DD DSN=databasefile5,DISP=SHR
//AQUIFER6  DD DSN=databasefile6,DISP=SHR
```

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 lowest level of the data 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 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 1-minute data. Using the values at the base level, statistics are computed at each higher level. For example, to calculate statistics at the 1-degree level for a 10-minute parameter, all values for the 10-minute blocks within a 1-degree block are used. From these values, the minimum, maximum, average, number of values, and standard deviation are computed for the 1-degree block. These five statistics are then placed in the data base in the record representing the 1-degree block. Similarly, values for all of the 10-minute blocks within a 3-degree block are used to calculate the same five statistics at the 3-degree level. This process continues and a set of five statistics that reflect all the 10-minute blocks for a parameter are stored in the root level.

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

Input

The input is one card containing the following:

Col 1-4	Data-base password with the proper update authority
Col 5-12	Study unit (HIPLAINS for HPRASA)
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 name 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 can be used to execute the Statistics Program. To execute the procedure, the following JCL would be used:

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

The procedure DBSTAT could appear as follows:

```
//DBSTAT  PROC  TIM=2,REG=670K,PROG=DBSTAT,SP1=1,SP2=10
//**********************************************************************
//*  DBSTAT: A PROCEDURE TO RUN THE HPRASA DATA BASE STATISTICS
//*          PROGRAM
//*          WRITTEN BY CARMELO F. FERRIGNO  SEPTEMBER,1981
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*   TIM     TOTAL RUN TIME (2)
//*   REG     REGION SIZE (670K)
//*   PROG    LOAD MODULE NAME (DBSTAT)
//*
//**********************************************************************
//STAT  EXEC  PGM=&PROG,TIME=&TIM,REGION=&REG,PARM='ISA(5K)'
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB.DISP=SHR
//         DD  DSN=mylib,UNIT=3330-1,VOL=SER=myvol,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=133,BLKSIZE=1330)
//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))
//AQUIFER1 DD DSN=databasefile1,DISP=SHR
//AQUIFER2 DD DSN=databasefile2,DISP=SHR
//AQUIFER3 DD DSN=databasefile3,DISP=SHR
//AQUIFER4 DD DSN=databasefile4,DISP=SHR
//AQUIFER5 DD DSN=databasefile5,DISP=SHR
//AQUIFER6 DD DSN=databasefile6,DISP=SHR
```

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, Z) triplets. The second phase uses the (X, Y, Z) 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, Z) 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 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 for the parameter 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 which of the interpolation techniques has been requested. 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 1-minute 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 accomplish 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 1-minute 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 1-minute data. There may be a sufficient number of 1-minute values within the model to use another interpolation technique. Second, if all the retrieved 1-minute 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 1-minute values within the model. If this estimate is greater than 1600 points, the run is aborted, and the user is asked to re-run the job using a different interpolation technique.

When using average or weighted-average interpolation, the user has three additional options for 1-minute data. First, only 1-minute values can be used to determine the model matrix. Second, only 10-minute values can be used to determine the model matrix. Third, a mixture of 1-minute and 10-minute values can be used. For the first and third option, 1-minute data are retrieved from the data base; for the second option, 10-minute values 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 1-minute and 10-minute values.

Model-Core Determination

Model-core determination is a process that is used with 1-minute data. The process divides the model into two regions. The first region is called the model core and is the area where 1-minute 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 10-minute values will be used to compute the model matrix.

The model core is used because of the possibility that, using 1-minute 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 10-minute values 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 10-minute block at the latitude and longitude of the principal node. This area, the standard comparison area, contains exactly one 10-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 blocks are calculated. The ratio of the area of the block to the standard comparison area yields an estimate of the number of 10-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 10-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 10-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 10-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 1-degree data, use trend-surface analysis. 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 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 1-minute data. However, if the program estimates that there are more than 1600 1-minute values within the model, the run will abort and a message will be printed suggesting the user re-submit the job 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, Z) 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 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 block which contains the point is calculated. The value of the point divided by the square of the distance and the reciprocal of the square of the distance are then calculated; sums of each of these results are kept for each block. After all the points have been properly located, these quantities are divided and the result is assigned as the value for that block. The calculation for each block is as follows:

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

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 abort.

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 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 parameter 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. One-Degree Data: With 1-degree data, the model grid may be regular or irregular. Because the amount of data-base input is 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 do not fall into the model area. If trend-surface analysis is requested, all the data are used for determining the model matrix.

2. Ten-Minute Data: With 10-minute data, the model grid may be regular or irregular. Because the amount of data-base input is 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 do not fall within the model area. If trend-surface interpolation is requested, all the data are used.

3. One-Minute Data: With 1-minute 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 10-minute values are to be used to determine the model matrix. The user must have retrieved 10-minute values from the data base. If this option is chosen and the retrieved data contain the 1-minute records, the run will abort. 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 1-minute 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 1-minute 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 1-minute values and 10-minute values 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 1-minute and 10-minute values, 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 card 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 1-minute data and the program estimates 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 re-run after the card 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 occurred. 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

Card Input: Read from data set DATAIN

Card (1) - Execution Options

Col 1	-	0	Regular Grid Pattern
		1	Irregular Grid Pattern
Col 2	-	blank	If the parameter is stored as 10-minute or 1-degree data
		0	10-minute values used
		1	1-minute values used
		2	Mixture of 1-minute and 10-minute values
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

Card (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)

Card (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 card depends on the value of the first execution option. If regular grid option set:

Card (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:

Card (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 card is used if a mixture of 1-minute values and 10-minute values are used to compute the model matrix. If the program determines the model core, the user must input the number of 10-minute values needed in a model block to use the 10-minute values to compute a value for a block.

Card (5) - Comparison number of ten-minute values

Col 1-2	Number of 10-minute values (I2) -- usually set to 5

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

Card (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

Card Input: Read from data set DATAIN

Card (1) - Same as Card (3) of step 1
Card (2) - Same as Card (4) of step 1

If trend surface analysis is requested a third card is added containing the order of the polynomial.

Card (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 10-minute data read from DBINPT and 1-minute data read from DBINPT2

Output:

Messages - written to SYSPRINT
Model Matrix - written to MODDATA

Sample Procedure

To run the program the following JCL could be used:

```
// . . . JOB . . .                                    -Job Card
//PROCLIB DD DSN=procedure.library,DISP=SHR           -Procedure Library
//  EXEC DTIP,NAME1=input                             -Step Card
//DTP.DATAIN DD *
     input for first step
//INTERP.DATAIN DD *
     input for second step
//
```

The procedure DTIP could appear as follows:

```
//DTIP  PROC  TIME1=2,TIME2=1,REG1=430K,REG2=190K,PROG1=DTP,
//            PROG2=INTERP,UNIT1='3330-1',VOL1=myvol,NAME1=NULLFILE,
//            UNIT2=SYSDK,VOL2=,NAME2=,UNIT3=SYSDK,VOL3=,NAME3=
//**********************************************************************
//*  DTIP: A PROCEDURE TO RUN THE HPRASA DATA TRANSFORMATION 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.
//*        WRITTEN BY CARMELO F. FERRIGNO  APRIL,1981
//*
//*  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 (430K)
//*  REG2    REGION SIZE FOR INTERP STEP (190K)
//*  PROG1   LOAD MODULE FOR DTP STEP (DTP)
//*  PROG2   LOAD MODULE FOR INTERP STEP (INTERP)
//*  UNIT1   RETRIEVED DATA BASE DATA DEVICE (3330-1)
//*  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 (NONE)
//*  VOL2    INTERP STEP FIRST INPUT FILE VOLUME (NONE)
//*  NAME2   DTP STEP FIRST OUTPUT FILE DSNAME (NONE)
//*  NAME2   INTERP STEP FIRST INPUT FILE DSNAME (NONE)
//*  UNIT3   DTP STEP SECOND OUTPUT FILE DEVICE (SYSDK)
//*  UNIT3   INTERP STEP SECOND INPUT FILE DEVICE (SYSDK)
//*  VOL3    DTP STEP SECOND OUTPUT FILE VOLUME (NONE)
//*  VOL3    INTERP STEP SECOND INPUT FILE VOLUME (NONE)
//*  NAME3   DTP STEP SECOND OUTPUT FILE DSNAME (NONE)
//*  NAME3   INTERP STEP SECOND INPUT FILE DSNAME (NONE)
//*
//***********************************************************************
//DTP  EXEC  PGM=&PROG1,TIME=&TIME1,REGION=&REG1,PARM='ISA(192K)'
//*
//*  STEP 1:  DATA TRANSFORMATION
//*
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,UNIT=3330-1,VOL=SER=myvol,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=12,BLKSIZE=6444),
//            SPACE=(TRK,(5,5))
//DBOTPT2  DD  UNIT=&UNIT3,VOL=SER=&VOL3,DSNAME=&NAME3,
//             DISP=(NEW,PASS,DELETE),
//             DCB=(RECFM=FB,LRECL=12,BLKSIZE=6444),
//             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,UNIT=3330-1,VOL=SER=myvol,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 mathematical functions: (1) Linear combinations; (2) multiplication; (3) division; (4) logarithm; and (5) antilogarithm of data sets that are in the data base structure and format. The functions take the following forms:

LINEAR Function: A*X1 + B*X2 + C
MULTIPLICATION Function: A*X1*X2 + C
DIVISION Function: A*(X1/X2) + C
LOGARITHM Function: A*LOG(X1) + C
ANTILOGARITHM Function: A*ANTILOG(X1) + C

where X1 and X2 represent data base format data, and A, B, and C are constants.

With the five available 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;
4. Compute the antilogarithm of the results of step 3.

An alternative would be to express the above equation as:

$$LOG(Y) = N*LOG(X1) - N*LOG(X2) \ .$$

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;
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.

Input

The mathematical function to be executed and all other parameters will be described on the input record. There are a maximum of five fields on this record. One field describes the function to be performed. This is the FNCTION field and can take one of five values:

LINEAR	(A*X1 + B*X2 + C)
MULTIPLY	(A*X1*X2 + C)
DIVIDE	(A*(X1/X2) + C)
LOG	(A*LOG(X1) + C)
ANTILOG	(A*ANTILOG(X1) + C)

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 in the input records. For the functions MULTIPLY, DIVIDE, LOG, and ANTILOG, the constants A and C must be defined. For the function LINEAR, A, B, and C must be defined. Failure to define one or more of the required constants will cause the run to abort. The last 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.

These five fields are input in a free-field format. They can be placed on one or more cards. 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. Examples are as follows:

```
FNCTION='LINEAR' A=1 B=1 C=0 PARM='WATER TABLE-1960';
FNCTION='LOG',PARM='SPECIFIC YIELD',C=0,A=-10;
PARM='CROP REQUIREMENTS' FNCTION='ANTILOG' A=1 C=0;
```

The other major input to the Data-Manipulation Program is the data file 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*, 350*, 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 MVI from the 001*, 002*, and 003* records. By using the value of the data density, the program determines which of the records contains the data 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 sets can contain more or less data than the other. The program only performs the data manipulations for the data points that are common to both sets.

Output

The output is in two forms. The first is the standard print output that contains information such as the selected function, the values of A, B, and C, and the names of the parameter or parameters being processed. The new scale factor and the new MVI associated with the output data are also shown.

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

```
// . . . JOB . . .                                         -Job Card
//PROCLIB  DD  DSN=procedure.library,DISP=SHR              -Procedure Library
//   EXEC   DBMATH,NAME1=input1,NAME2=input2,NAME3=output
//LC.CMDIN  DD  *                                          -Function Input
     one or more cards
//
```

The procedure DBMATH could appear as follows:

```
//DBMATH  PROC  TIM=2,REG=455K,PROG=DBMATH,UNIT1='3330-1',VOL1=myvol,
//        NAME1=NULLFILE,UNIT2='3330-1',VOL2=myvol,NAME2=NULLFILE,
//        UNIT3='3330-1',VOL3=myvol,NAME3=NULLFILE
//*********************************************************************
//*  DBMATH. A PROCEDURE TO RUN THE HPRASA DATA MANIPULATION PROGRAM
//*          WRITTEN BY CARMELO F. FERRIGNO  JUNE,1981
//*
//*  USER MAY SUPPLY THE FOLLOWING PARAMETERS (DEFAULTS IN PARENTHESIS)
//*
//*  TIM     TOTAL RUN TIME (2)
//*  REG     REGION SIZE (455K)
//*  PROG    LOAD MODULE NAME (DBMATH)
//*  UNIT1   FIRST DATA BASE INPUT FILE DEVICE (3330-1)
//*  VOL1    FIRST DATA BASE INPUT FILE VOLUME (myvol)
//*  NAME1   FIRST DATA BASE INPUT FILE DSNAME (NULLFILE)
//*  UNIT2   SECOND DATA BASE INPUT FILE DEVICE (3330-1)
//*  VOL2    SECOND DATA BASE INPUT FILE VOLUME (myvol)
//*  NAME2   SECOND DATA BASE INPUT FILE DSNAME (NULLFILE)
//*  UNIT3   OUTPUT FILE DEVICE (3330-1)
//*  VOL3    OUTPUT FILE VOLUME (myvol)
//*  NAME3   OUTPUT FILE DSNAME (NULLFILE)
//*
//*********************************************************************
//*
//LC  EXEC  PGM=&PROG,TIME=&TIM,REGION=&REG,PARM='ISA(260K)'
```

```
//STEPLIB  DD  DSN=SYS1.PLIX.TRANSLIB,DISP=SHR
//         DD  DSN=mylib,UNIT=3330-1,VOL=SER=myvol,DISP=SHR
//*
//*  DEFINE STANDARD I/O DATA SETS
//*
//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=6400),
//             SPACE=(TRK,(5,5))
//*
//*  DEFINE TEMPORARY FILES
//*
//TEMPIO   DD   DSN=&&TEMP,UNIT=SYSDK,
//              DCB=(RECFM=FB,LRECL=29,BLKSIZE=6438),SPACE=(TRK,(5,5))
//TEMPIO2  DD   DSN=&&TEMP2,UNIT=SYSDK,
//              DCB=(RECFM=FB,LRECL=52,BLKSIZE=6396),SPACE=(TRK,(5,5))
```

The Graphics Program

The Graphics Program produces contour maps and three-dimensional perspective drawings (3-D plots) of data. The program will process data of densities 2 through 4. The data is 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 contour 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 a 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 data that is 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 re-producing 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 data values increase and doubling the number of data values more than doubles the processing time.

Input

Card Input: Read from data set DATAIN. The first set of cards are always required.

Card (1) - Execution Options

Col 1	blank	No map projection selected (using previously prepared 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 card depends on the value placed in column 3 of Card 1. If generating data and plot(s):

Card (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:

Card (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 printout of a previous execution of the Graphics Program.

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

Card (3) - Scale of map

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

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

Card (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)

Card (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 cards is required only for 3-D plots.

Card (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

Card (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 messages 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 data value showing the data 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

To run the program, several variations of JCL could be used. If generating data and plotting contour maps and (or) 3-D plots, the following JCL could be used:

```
//. . . JOB . . .
//PROCLIB  DD  DSN=procedure.library,DISP=SHR
//  EXEC  DBGRAPH,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  DBGRAPH,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  DBGRAPH,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  DBGRAPH,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 is saved, and *savefile2* is the data set name of the file where the data needed to produce 3-D plots is saved.

The procedure DBGRAPH could appear as follows:

```
//DBGRAPH  PROC  PROG1=DBGRAPH,PROG2=gpcpgrid,PROG3=MDFILE,
//         PROG4=three906,TIME1=1,TIME2=1,TIME3=1,TIME4=1,
//         REG1=275K,REG2=450K,REG3=110K,REG4=250K,
//         NAME1=NULLFILE,UNIT1='3330-1',VOL1=myvol,
//         NAME2=NULLFILE,UNIT2='3330-1',VOL2=myvol,
//         DSP2='(NEW,KEEP,DELETE)',NAME3=NULLFILE,UNIT3='3330-1',
//         VOL3=myvol,DSP3='(NEW,KEEP,DELETE)',NAME4=NULLFILE,
//         UNIT4='3330-1',VOL4=myvol,DSP4='(NEW,KEEP,DELETE)'
//*********************************************************************
//*
//*   DBGRAPH: A PROCEDURE TO RUN THE HPRASA 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 (DBGRAPH)
//*        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 (275K)
//*        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 (3330-1)
//*        VOL1      DATA BASE DATA INPUT FILE VOLUME (myvol)
//*        NAME2     CONTROL POINT DATA  FILE DSNAME (NULLFILE)
//*        UNIT2     CONTROL POINT DATA  FILE UNIT (3330-1)
//*        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 (3330-1)
//*        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 (3330-1)
//*        VOL4      GPCP PNCH FILE VOLUME (myvol)
//*        DSP4      GPCP PNCH FILE DISPOSITION ((NEW,KEEP,DELETE))
//*
//*********************************************************************
//*
//GRAPH  EXEC  PGM=&PROG1,TIME=&TIME1,REGION=&REG1,PARM='ISA(11K)'
//*
//*         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=6400),SPACE=(TRK,(5,2))
//*
//*  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=plotter.library,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,(5,2))
//*  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=6400),SPACE=(TRK,(5,5))
//*
//*  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=plotter.library,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.
3. The computer system has the Fortran IV and PL/I 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 the application programs.

A section is provided for each program. Each section begins with an introduction and then lists changes required for each subroutine in the program. A listing is provided only for the routines 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 for the program, the Edit Program has become a very large and somewhat unmanageable program. This program will require more implementation changes than any other program in the system. In its present state, the program performs the necessary tasks; hence, no major effort has been made to revise the program.

The Edit Program checks the input to the Load, Retrieval, and Instant-Update Programs. The first change to the Edit Program should be to break it into three separate programs; then, each of the three editing programs would check the input to only one other program, which would make the editing programs more manageable, 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.

The program also contains some subroutines that were not implemented, because of changes in the programs for which these subroutines were originally written. The subroutine called CHK004 was to be used to check the input to the Move Program. Because of design changes in the Move Program, this subroutine is no longer needed and could be deleted. The subroutines CHK007, CHK008, and CHK009 were intended to check input to a general updating program, which was not written because this type of program was not needed. CHK301 and CHK302 were to be used to edit data that were to be stored in the data base in 30-minute and 15-minute blocks. No data at these levels of detail are needed for the High Plains study. These subroutines would be needed if data were stored at these levels of detail. The subroutine BOUNDS was to be used to check whether or not data to be loaded fell within the boundary of the study area. This was never fully implemented, because boundary checking is now performed in the data-generation step. In general, these subroutines could be eliminated, and the Edit Program would still perform its assigned task. If this is done, references to these subroutines must be deleted from the calling routines.

Although information is provided to implement the program in its present state, it is suggested that the major design changes be made. The program does work, but it is by no means an efficient program in its present condition. The following subroutines within the Edit Program need to be changed.

BDAT

This is a block data subroutine used to set up common areas and initialize variables. The dimensions of several of the arrays could be changed. The BND and LEVEL1 arrays are used for boundary and duplication checking. They cover the entire United States 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 these arrays 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 BND and LEVEL1 arrays. 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 valid users.

INIT

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

CHK100

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. A check is made to assure that the latitude, on a 100* record, is divisible by three. A similar check is made for the longitude. These checks are valid for the High Plains aquifer, because the chosen 3-degree blocks fulfill these criteria. This does not imply that the 3-degree blocks chosen for another study must fulfill this criteria. These checks can be eliminated if necessary.

NAMECK

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

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

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 being studied.

LTCHK2

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

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.

STK

This is the only subroutine within the Edit Program that accesses the data base. Hence, it is the only routine that contains System 2000 PLEX commands. The data-base name used in some of the PLEX statements must be changed. This routine uses the arrays BND and LEVEL1 for boundary and duplication checking. It may be desirable to change the dimension of these arrays. Constants used to compute positions in these arrays will also need to be changed.

This subroutine contains some code pertaining to boundary checking. It is only the initial code required to fully implement boundary checking. Its purpose is to retrieve some of the boundary data from the data base. The full implementation of boundary checking would require the development of the subroutine called BOUNDS.

DUPL

This subroutine also uses the BND and LEVEL1 arrays. This routine is mainly used for duplication checking of data prior to loading into the data base. It also contains some code for boundary checking. If boundary checking is to be implemented through the BOUNDS subroutine, this code should be extracted from this routine.

Load Program

The Load Program consists of a main procedure containing internal subroutines and three external subroutines. The program loads parameters of densities 1 through 4. 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 3-degree blocks of data at one time. The main program and its internal subroutines will be discussed as one unit under the heading of DBLOAD; the external subroutines will be discussed separately.

DBLOAD

The main portion of the program should work essentially unchanged. There are two incomplete internal subroutines within the main program. These routines, called LOAD_30_MIN and LOAD_15_MIN, can be used to load data at 30-minute and 15-minute levels of detail. If data of these densities are required for another study, these routines can be expanded. The data-base name used in the System 2000 PLEX commands should be changed.

An array, called GRAPH, is declared in the main program. It is used to produce graphics output for the Load Program 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. It has a position for each 1-degree block within and on the boundary of the aquifer. As a 1-degree block is loaded in the data base, an asterisk is placed in the GRAPH array position corresponding to the 1-degree block. Dimensions of this array would have to be changed for another study.

INITGR

This routine 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 routine would have to be completely changed for a different study.

REPORT

This routine 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. It is dimensioned exactly like GRAPH. It is initialized with the letters 'O', 'P', and 'C'. The 'O' means that the 1-degree block corresponding to that position is outside of the study area. The 'P' means that the block is partially within the study area, and the 'C' means that the 1-degree block is completely 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.

Statistics Program

This program computes statistics for parameters stored at densities 2 through 4. 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 PLEX commands must be changed. No other modifications are required.

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 data of densities 1 through 4. 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 PLEX commands must be changed. The routine uses system subroutines to compute date and time of the run. These routines 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.

Instant-Update Program

The Instant-Update Program consists of a main program and two external subroutines. The program processes proposal data for data densities 2 through 4. 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 subprograms 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 permanently 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 subprograms easily can be modified to incorporate any changes to these rules.

INSUPDT

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 PLEX commands must be changed.

UPDATE1

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

UPDATE2

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

Move Program

The Move Program consists of a main procedure and six external subprograms. It manipulates the proposal records for parameters stored at densities 2 through 4. Because of its modular design, it easily can be expanded if more densities are required. In the three subprograms 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 PLEX commands must be changed.

Data-Manipulation Program

This program performs mathematical manipulations on data for densities 2 through 4. There are five functions available for manipulating the data. Because of its modular design, the program can be modified to manipulate data stored at other densities and to provide other functions. The program consists of one main program, called DBMATH, and nine subprograms.

DBMATH

The variables LATMIN3, LATMAX3, LONMIN3, LONMAX3, LATMIN1, LATMAX1, LONMIN1, LONMAX1, DD3, and DDD3 are used in conjunction with the subprograms that perform the multiplication, division, and linear functions. These variables are initialized in this program to values that reflect latitude and longitude extremes of the High Plains aquifer. Variables that end in the number '3' refer to the minimum and maximum latitudes and longitudes of the 3-degree blocks. Variables that end in the number '1' refer to the minimum and maximum latitudes and longitudes for the 1-degree blocks. These initial values would have to be changed for another study.

LINEAR2

The arrays EXIST1, EXIST2, and INTSECT are dimensioned to have one position for each 3-degree block within the High Plains aquifer. The arrays VALUES1 and VALUES2 are dimensioned to contain one position for each 1-degree block within the aquifer. These array dimensions would have to be changed for another study.

LINEAR3

The arrays EXIST1, EXIST2, and INTSECT3 are dimensioned to have one position for each 3-degree block. The arrays EXIST11, EXIST12, and INTSECT1 are dimensioned to contain one position for each 1-degree block. The arrays VALUES1 and VALUES2 are dimensioned to have one position for each 10-minute block. All of these dimensions would need to be changed for another study.

LINEAR4

The array EXIST3 is dimensioned to have one position for each 3-degree block within the High Plains aquifer. These dimensions would need to be changed for another study.

Data-Transformation Program

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

DTP

The DTP program consists of a main procedure, called DTP, and 18 subprograms. 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 data of densities 2 through 4. The program can be expanded to accommodate other data densities. Two subprograms within the DTP program require changes.

PREP4

The array LONDIS is used to estimate the number of 1-minute 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 1-minute 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 subprogram. It is the same array as used by the PREP4 subprogram. The same comments apply to LONDIS as discussed for PREP4.

INTERP

The INTERP program consists of a main routine and 5 subprograms. 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 subprograms may require changes.

TREND

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

☆ U.S. Government Printing Office 1983 - 677-355/89 Reg. 8