DEVELOPMENT OF AN INTELLIGENT DECISION SUPPORT SYSTEM FOR RESERVOIR MANAGEMENT

BY

JOVAN GRAHOVAC

A Thesis

Submitted to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Civil Engineering

University of Manitoba

Winnipeg, Manitoba

August, 1990

ISBN 0-315-71815-3

DEVELOPMENT OF AN INTELLIGENT DECISION SUPPORT SYSTEM FOR RESERVOIR MANAGEMENT

BY

JOVAN GRAHOVAC

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

Expert systems are receiving a widespread attention across various disciplines of science, engineering, and business. This thesis describes a part of a larger research effort conducted at the University of Manitoba and Manitoba Hydro that focuses on joint application of expert systems, computer graphics, and systems analysis to practical problems within the field of water resources. These three elements, along with engineering expertise, form the basis of the intelligent decision support approach. This approach was used in the development of a computer tool designed to support reservoir and energy scheduling. The thesis describes the experiences gathered during the work on the project, and it gives recommendations based on the lessons learned from it as well as suggestions for future development.

ACKNOWLEDGEMENTS

Many people at Manitoba Hydro and the University of Manitoba have been involved in the development of the concepts and ideas used in this research. I wish to express my appreciation to Mr. A.D. Cormie for his valuable assistance and time spent in connection with this project. Involvement of Mr. Paul Barritt-Flatt is also greatly appreciated. A special thanks to my advisor Dr. S.P. Simonovic for his guidance during this study.

I would like to acknowledge the financial support provided by Manitoba Hydro through its Research and Development Committee.

A special thanks to my fellow students at the University of Manitoba for their understanding and encouragement, and especially to miss Kim Barlishen who helped me at the beginning of my involvement in this project and whose work on its previous phase provided a solid basis to build upon.

TABLE OF CONTENTS

Appendices page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1.
1. INTRODUCTION

1.1 Problem

Artificial intelligence (AI) is the area of study that explores ways of making computers reason, make judgement, and learn. During its thirty years of research and development, AI has mainly been focused on two goals. One is the development of computational, machine-based, models of intelligent behavior including both its cognitive and perceptual aspects. The other goal is more engineering and practice oriented and it consists of the development of computer programs that can solve problems normally thought to require human intelligence [Duda and Shortliffe, 1983]. In spite of the fact that neither of these goals have been completely achieved so far, the field has accumulated theoretical knowledge and developed techniques mature enough to be applied to practical problems in robotics, medicine, image analysis, and various branches of engineering.

A class of AI problem-solving computer programs intended to serve as consultants and support in decision making and, in doing so, can reach performance comparable to that of a human expert are often called "expert systems". Developers of these programs try to capture and encode the knowledge human specialists use while solving problems in their everyday work. Development of expert systems came with a more pragmatic approach in AI that has resulted in emergence of knowledge engineering, a new profession that examines methods of acquiring knowledge from experts and representing it correctly. Instead of trying to emulate the most elusive aspects of human reasoning such as the use of analogy or mental images, this approach focuses on a relatively narrow problem and tries to provide a computer with enough explicit knowledge

needed to solve it. Because of their strong emphasis on knowledge, these computer applications are also called "knowledge-based expert systems".

A great potential for expert systems development is seen in various fields of engineering. Knowledge-based systems that encode design manuals and procedural codes thus putting this inactive and static knowledge into active forms are often constructed in order to assist engineers in performing routine, time consuming tasks. In water resources engineering (WRE) however, expert systems applications need to do more than this in order to be successful. The main characteristics of the systems this field of engineering deals with are uncertainty and their ever growing complexity. Through their long term experience, experts that do analysis and planning, or manage operation of water resources systems, have gathered much expertise and heuristic knowledge, and have learned how and where to apply intuition and engineering judgement in their work. In addition to this, a number of these experts have seen systems analysis and its sophisticated optimization techniques make their way into water resources engineering during the last two decades. Application of these techniques was an answer to the growing complexity of water resources systems. It has resulted in a substantial body of, often poorly documented and mainly individual, expertise. Obviously, a successful application of expert system technology in water resources engineering would, in addition to encoding knowledge already available in books and manuals, capture some of this private expertise and make it available to other individuals within an organization.

One of the main characteristics of almost any task in water resources engineering is having to deal with massive spatially and time varying data. Interactive computer graphics appears to be a good tool for making data management easier for managers, planners and policy makers [Loucks et al., 1985]. Since in transferring engineering knowledge and explaining different

concepts we often use graphs and pictures, the need to combine computer graphics and expert systems in order to assist water resources engineers in performing their tasks is obvious.

1.2 Purpose

This research describes an actual development of an expert system in water resources engineering. The selected problem includes the use of linear programming in planning and operation of a complex water resources system that consists of a number of reservoirs, rivers, hydro generating stations, and control structures. The research goals are to carry out knowledge acquisition and capture the diverse engineering and systems analysis expertise necessary to perform the task, to formulate appropriate rules, and to enter the rules into a computer code. Another aspect of this research is to integrate the available expert system technology with interactive computer graphics, to explore how these two tools interact, and to observe benefits and problems that arise from this interaction. This approach is based on the concept of intelligent decision support [Simonovic and Savic, 1989] that links engineering expertise, the systems approach, expert systems, and interactive computer graphics. The research focuses on how the experiences gained so far in knowledge engineering and the available technology and tools can be used in practice, rather than on achieving computer performance that may be compared with that of human experts.

1.3 Scope

This thesis examines application of the Intelligent Decision Support (IDS) concept to a case study that involves a joint project between the University of Manitoba and Manitoba Hydro. Manitoba Hydro operates a system composed of a number of reservoirs, lakes, natural and controlled flows, and thirteen hydro and

three thermal generating stations. A linear programming optimization model called EMMA is used to optimize planned short-term and long-term operations of the system. The use of EMMA is expertise-intensive and frequent. Since it is necessary in the decision-making process in planning and operation of the system it is selected as the case study in this research.

A review of the four basic elements and ideas behind the IDS approach in Section 2 is followed by the description of the previous work on the project in Section 3. The problem selected as a case study and the acquired engineering knowledge that is used at Manitoba Hydro in solving it are presented in detail in Section 4. Description of the developed IDS system is given in Section 5, and the experiences gained during this research are summarized in Section 6 along with recommendations for possible future development.

CHAPTER 2.
INTELLIGENT DECISION SUPPORT

Standard knowledge engineering (KE) relies on communication between the expert and the knowledge engineer in creating an expert system application. The approach that is used in this research is different from classical knowledge engineering procedures [Simonovic and Savic, 1989]. It is called the engineering expert system approach (EES) and it replaces the knowledge engineer by a water resources engineer who has additional knowledge about expert systems techniques. Unique expertise in the engineering field can thus be combined with knowledge about expert systems technology in an efficient manner. Judging from the expert systems applications reported in the literature and at different conferences it appears that approaches similar to EES are widely accepted within different subdisciplines of civil engineering. In contrast to traditional expert system research where the focus is on general procedures and frameworks, with the EES approach the research concentrates on the application and development of technology as it directly relates to the specific engineering field and the particular problem environment.

The EES approach is the fundamental characteristic of the IDS concept, graphically illustrated in Figure 1. In order to assist in water resources engineering planning, decisions, and operations, this concept combines the following four basic elements : (i) engineering expertise, (ii) systems approach, (iii) computer graphics, and (iv) expert systems. The concept envisions the expert system as a part of a larger system more suited to the user's needs. It extends the hybrid model-based decision support approach [Fedra et al., 1986]. An approach similar to the EES was taken by Palmer et al. [1987, 1988] who use expert systems to integrate databases, mathematical programming, and computer graphics.

The presented concept envisions decision and policy makers and engineers as major users of the decision support software. In this environment, the computer is seen as a tool for establishing a link between the expert and the decision maker, in other words, between science and policy. The computer's role is therefore expanded from a tool for computational analysis to a vehicle for communication, experimentation, training and learning. The major strength of this concept is that the products are application and problem oriented. In this way, new technology can be successfully combined with more classical techniques of engineering analysis, data processing, and systems analysis.

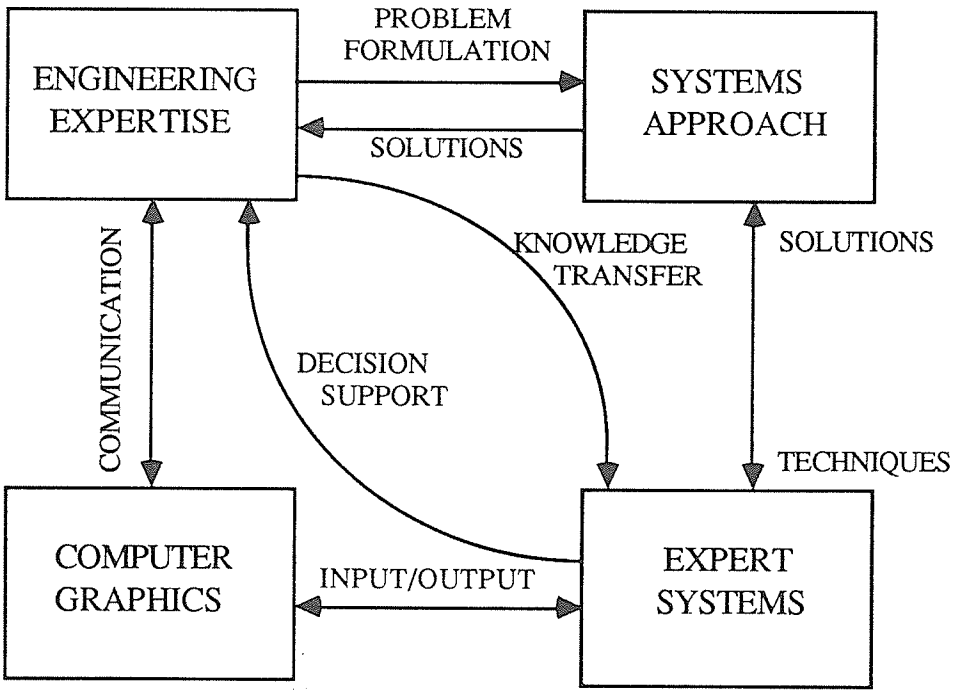


Figure 1. Graphical illustration of Intelligent Decision Support concept in water resources (after Simonovic and Savic, 1989)

2.1 Expert Systems

2.1.1 Expert System Elements

The British Computer Society defines an expert system as "the embodiment within a computer of a knowledge based component from an expert skill in such a form that the system can offer intelligent advice or take an intelligent decision about a processing function." Expert systems may function as an assistant or a partner to an expert, or to fill the void between an average worker or a novice and the expert [Jenkins and Jowitt, 1987]. A complete expert system should consist of the following six elements:

1) user interface,
2) explanation subsystem,
3) knowledge base,
4) working memory,
5) inference engine, and
6) knowledge acquisition subsystem.

The **user interface** provides for communication between the user and the expert system. It prompts the user for input which it translates to the form used by the expert system and it presents generated results to the user. Development of a good and friendly user interface is a significant step towards the acceptance of an expert system by the final user, which means that it can be critical for success of an entire project.

The **explanation subsystem** tries to explain the reasoning behind any action the expert system takes. This can be a trace of the execution of the expert system or the ability to give answers if the user wants to know why the expert system is asking a particular question or performing an action.

The **knowledge base** contains knowledge about the application domain of the expert system. The simplest way of encoding this knowledge is through a set of rules that define conditions and the corresponding actions to be taken if these conditions are met. In cases where a large number of rules makes their structuring or grouping helpful, semantic nets and frames can be used. These two approaches have been described by Waterman [1986]. Most expert systems reported in engineering literature have successfully used simple sets of rules, although their further development may indicate the need for a more sophisticated knowledge representation.

The **working memory** contains information about the current problem to be solved. It contains relevant information provided by the user or derived through the expert system's reasoning about that particular user-defined problem.

The **inference engine** controls the reasoning process of the expert system. It has to perform some kind of search through the knowledge base in order to find knowledge necessary to derive conclusions or to ask the user intelligent questions on the way towards the final solution. The two most often used search techniques are backwards chaining and forward chaining. In backward chaining, the system starts with a solution and, using appropriate rules from the knowledge base, tries to find the data and conditions that will lead to that solution. In forward chaining, the system starts from known data and facts and, using applicable rules for deriving conclusions, progresses towards the final solution.

The **knowledge acquisition** subsystem modifies the contents of the knowledge base. In the ideal situation, this process should take place through machine learning - automatic generation of knowledge based on generalizations drawn from the knowledge base and the problems solved by the expert system. At the current level of AI development however, this is usually accomplished through

manual update of the knowledge base by the system developer, or direct intervention by the expert.

2.1.2 Expert System Development

Development of an expert system is usually accomplished through cooperation of: experts whose knowledge is to be used by the system, knowledge engineers whose task is to perform knowledge acquisition and representation, computer programmers responsible for encoding the formulated knowledge into a computer code, and end users who will interact with a computer.

Tools for expert systems development include programming languages and shells. Programming languages, such as Lisp or PROLOG, offer much flexibility, but do not provide any assistance in knowledge representation [Barlishen, 1989]. Shells usually support some kind of knowledge representation and search techniques, and often provide friendly user interfaces. In addition to this, most shells offer an explanation facility, which can range from a simple trace of execution of the expert system to developer-defined or generated explanations [Barlishen, 1989]. Some widely available shells are friendly enough to encourage knowledge engineers or experts to directly enter knowledge into computers without help from programmers.

The development of an expert system usually consists of the following phases [Rolston, 1988]: selection of a suitable problem, construction of a prototype expert system, formalization of knowledge compatible with the chosen development tool, implementation, evaluation of the expert system's performance, and long-term evaluation with necessary adjustments and corrections in the expert system in order to increase its usefulness. Due to the nature of expert systems, many aspects of the application domain remain unknown until an actual implementation. Unlike in classical programming, the exact specifications of what

can be done and how, are impossible to make in advance [Bobrow et al., 1986]. As a consequence, in most cases, several iterative cycles are needed for development of an expert system.

2.2 Engineering Expertise

It has been said that "engineering consists of acquiring and manipulating knowledge so it can be used productively", and expert systems try to do the same thing [Hogley and Korncoff, 1986]. Some of that knowledge, especially the part that is closely related to basic scientific principles or whose origin can be traced back to well known facts, is well documented in books and design manuals. Engineering expertise means having skills in using this, often large amount of knowledge and number of rules that have to be obeyed, in an efficient and accurate manner. Expert systems that encode design manuals and procedural codes, thus putting this inactive and static knowledge into active forms, can assist an inexperienced or average engineer to approach the level of an expert. They can help perform routine time consuming tasks faster and in a more consistent way.

Another type of expertise comes from a long term experience in working on a particular problem. It means knowing when and how to apply intuition and engineering judgement gained through work on a large number of cases. This kind of engineering expertise is important in dealing with problems where the trace-back to known facts or underlying physical principles is difficult to establish, when available data are inaccurate or incomplete, or when a severe time or budgetary constraint is imposed. Formulating and capturing some of this very scarce expertise means making it permanently available to other engineers within an organization.

In water resources engineering, knowing how to apply intuition and engineering judgment plays a very important role for a number of reasons. The

goal of water resources management is to ensure that water is available, in sufficient quality and quantity, at the right location and time, and to protect humans and their activities from the harmful effects of water [Kindler, 1986]. Uncertainty and complexities that are present in most water resources systems, stemming from their nature and size, diversity of water use and resulting conflicts of interests, and the impact these systems have on the environment and society make it very hard to accurately and completely describe these systems. Advancements in technology and massive use of computers have resulted in the fast acquisition of large volumes of data. Proper and timely interpretation and analysis of these data also requires a lot of experience. All this results in the need for good approximation and judgement in analysis, planning, or management of water resources systems, and therefore for potential application of expert systems.

The multidisciplinary nature of the problem, the uncertainty, and the complexity inherent in water resources systems result in a number of engineers specializing and becoming experts in narrow subdomains of the problem. An important future role for expert systems is seen in capturing the diverse expertise of all these experts thus facilitating communication between them and providing benefits to all other potential users of their knowledge.

2.3 Systems Approach

The systems approach can be defined in water resources as the use of rigorous mathematical methods to help determine preferred planning or operational decisions for complex systems. This approach combines knowledge of the available analytical tools, understanding of when each of them is appropriate, and skills in applying them in everyday practice. It is mathematical and intuitive, as are planning and operations processes.

Systems analysis has been making its way to water resources engineering in parallel with the massive introduction of computers into the field. Two major factors played an important role in this process: (i) the reduction in time required to make operating decisions; and (ii) the ability to incorporate into the decision making process all aspects of a very complex physical system. Coming up with the right answer in a timely manner results in significant benefits in the case of large water resources systems. It has been observed that the acceptance of systems analysis has been a rather slow process [Barritt-Flatt and Cormie, 1989] and that it occurred as the confidence in its optimization models grew. It can be expected that the acceptance of expert systems will take place in a similar manner.

Water resources engineers who use systems analysis methods in their work have developed a new kind of expertise which includes both thorough understanding of these methods and skills in applying them to various practical problems in the field. They have learned how to represent the physical systems they are dealing with to the models they are using. They have developed expertise in making correct assumptions, simplifications, and approximations. Expert systems can facilitate the use of analytical tools for those not familiar with them, and they can probably provide a significant help and save time to experts by doing as much as possible of intelligent data pre- and post- processing.

2.4 Computer Graphics

Most of the systems in the field of water resources consist of large numbers of different elements and their databases are usually hard to maintain and update. This is one of the reasons why they are difficult to analyze, model and present. According to Loucks et al. [1985] the primary purpose of developing computer models and using computer technology is to derive and manage data. The input and output files of these models are in the form of long, cumbersome tables and

sequences of numbers. Interactive computer graphics can enhance the use of these models and other software packages, as well as the utility of databases. It appears to be a good tool for making data management easier for managers, planners, and policy makers. Within a decision support context, this is particularly important in order to allow the user to have more time to concentrate on actual decision making.

The success of any modelling attempt is related to the involvement of nonmodelers in model use [Loucks et al.,1985]. The best way to involve nonmodelers is through a "friendly" interface, which usually refers to a pictorial display of information and the ability of the user to modify that display. Interactive computer graphics, in a general sense, refers to a display of pictorial information generated by a computer and to the man-machine interaction in modifying that display [Simonovic and Grahovac, 1990].

Widely available hardware and software designed to facilitate development of interactive computer graphics applications provide for combined use of alphanumeric, symbolic and graphical elements in color schemes, maps, graphs, etc. This can be used to display the current state of a system or output from a mathematical model, and to assist data input, editing and display. As a result of the wide availability of graphical input and output devices and the efficiency and appeal in transferring information, interactive computer graphics is increasingly becoming the standard means of interaction between the user and the computer. It seems logical to adopt this standard wherever appropriate for communication between the user and the expert system.

CHAPTER 3.
REVIEW OF PREVIOUS WORK

3.1 Decision Support Systems

The Term Decision Support Systems (DSS) was introduced approximately two decades ago, and there has been a little agreement till today between different authors about the main characteristics and the architecture of these systems. DSS are computer-based systems that support decision making. The borders that separate them from other computer-based systems are not yet well defined. Most of the available literature on DSS comes from the fields of business and management. Fiew authors have written about the subject in the water resources engineering context. These are the reasons why the development of DSS in business and management is briefly reviewed in this section of the thesis. Some of the most interesting works on the subject in engineering are also presented.

3.1.1 Decision Support Systems in Business and Management

In his book on DSS, R.J. Thierauf [1988] gives an overview of the evolution of information systems in business. Prior to the introduction of computers, these systems were primarily concerned with recording historic events, and they relied on manual methods of entry and retrieval. These systems made no attempt to integrate records that might serve several functions at the same time.

When computers were introduced in business, they were used as large accounting machines that could perform usual jobs quickly and efficiently. The information systems developed in this phase did not give managers more control over operations. The use of a single data entry for multiple purposes transcending the traditional boundaries within a business environment came with the development of Integrated Data Processing Systems.

The next step in the evolution towards DSS was the development of Integrated Management Information Systems. These systems provide decision oriented information that can be used in planning, control, and evaluation of activities in an organization. A class of these systems that can process relevant data and feed it back to the appropriate source in sufficient time to change or control the operating environment are called Time Management Information Systems. Numerous applications of this kind are in use in hotel accounting and reservation systems, airline reservation systems, and patient hospital records. Management Information Systems (MIS) have had their main impact on structured tasks where operating procedures, decision levels, and flows of information were predefined. They greatly reduced the cost and turnaround time, and their implementation led to the development of DSS.

DSS are designed to support decision making for problems with sufficient structure that justifies the use of computers but where the user's judgement is essential. Structure in a problem is in this context defined as the possibility of quantifying objectives and decision variables. Emphasis in DSS is on providing a supportive tool that does not automate the decision process, impose solutions, or replace the decision maker. According to Thierauf, the most important characteristics of DSS are:

1) The focus on problem finding and solving. The concept of problem finding is based on the use of the data base and quantitative models to discover trends and possible problems.

2) Interactive processing mode and query capabilities, especially the capability to answer "what if" questions.

3) A comprehensive systems approach.

DSS should be adaptive over time and provide support regardless of problem structure. A great emphasis should be on user-machine interface and the

use of graphic displays to distill quickly the essence of massive data. According to Thierauf [1988], future developments in DSS will be towards broader "what if" questions, the use of more complex quantitative models, and less structured problems.

Davis [1984] addressed the difference between DSS and the Knowledge Based Systems (KBS). According to this author, DSS assist in decision making by performing complex computations and by helping to manage large amounts of data. For each particular problem, these two tasks follow a predefined structure that has to be accommodated by the DSS. This is the main difference between DSS and KBS that are used without structural restraints and that are built recursively from the zero level to a comprehensive base of information.

3.1.2 Decision Support Systems in Engineering

DSS in business and management evolved exclusively from computer applications designed to manage data. In engineering however, DSS were additionally influenced by the previous use of numerous computational models for simulation and optimization. Guariso and Werthner [1989] define Environmental DSS as computer-based systems that help decision makers to utilize data and analytical models in the solution of unstructured problems in environmental engineering and management. The authors propose an architecture for these systems that consists of the following elements:

1) a friendly user interface,

2) the system manager control unit that controls all other parts of the system,

3) the data base that has to deal with spatial data and time series,

4) the model base with a set of quantitative models,

5) the knowledge base containing declarative knowledge necessary for handling the models and data.

According to Guariso and Werthner [1989], the main difference between DSS and Expert Systems (ES) is that unlike a DSS, an ES tries to take decision itself and therefore replace the human expertise and the expert. This is illustrative of the controversy and the confusion that still exist in the literature not only about DSS but ES as well.

DSS have been scarcely addressed in the water resources engineering literature. One interesting project, named HERMES, was initiated at Manitoba Hydro in 1984 and some of its parts are already in use [Herzog et al.,1985]. The project envisions the development of a DSS that could be used as a laboratory for conducting experiments by constructing, running, observing, and analyzing energy management models related to a system of reservoirs and hydro and thermal generating stations. HERMES will use diverse and changing models for load, energy market, and flow forecasting along withe a linear programming optimization model and a relational data base. An extensive use of computer graphics and spread sheet programs is envisioned within the Documentation Subsystem for preparation of management reports.

Palmer and Holmes [1988] describe a DSS used to aid in drought decisions. The system provides guidance for initiating voluntary and mandatory restrictions of water use. It incorporates an expert system component that contains human decision making expertise represented in a series of rules and facts. The DSS consists of the following elements:

1) an expert system that incorporates necessary expertise and integrates all other elements into a single system.

2) a linear programming model that is used for optimization,

3) computer graphics interface used for the display of information, and

4) data base management tools for the storage and manipulation of data.

This architecture, instead of confronting ES and DSS and adding to the confusion about these two terms, brings an innovative concept of incorporating ES within DSS. This concept is very similar to the IDS described in Section 2 of this thesis, and it focuses on providing the most efficient decision support tool that can be made using available technology and knowledge.

3.2 Interactive Computer Graphics

Water resources engineers often have to deal with massive spatial and time-varying data. There is a general agreement between the authors that addressed the problem during the last decade about the utility of computer graphics in easing this task [Fedra 1985, Fedra and Loucks 1985, Fedra et al., 1986, Herzog et al, 1985, Loucks et al., 1985, Palmer and Holmes, 1988, Taylor and French, 1985]. Interactive computer graphics appears to be a good tool for making data management easier for managers, planners and policy makers [Loucks et al., 1985]. In the field of water resources computer graphics should: (i) facilitate data input and editing; (ii) provide an effective interface between models and modelers; and (iii) improve the comprehension of spatial and time-dependent information [Taylor and French, 1985] .

When the objective is the communication of complex patterns or relationships, graphical display of information is certainly by far superior to any alphanumerical format [Fedra, 1985]. Since in transferring engineering knowledge and explaining different concepts we often use graphs and pictures, the need to combine computer graphics and expert systems within a DSS in order to assist water resources engineers in performing their tasks is obvious.

To know what one wants, one needs to know what he or she can get. [Fedra, 1985]. Therefore, the use of menus enhances the user's understanding of the

computer models he or she is using by making it clear what exactly can be done at any point of the user-computer interaction. The use of menus also speeds up the work with the computer by letting the user select from a predefined set of possible options instead of making him or her type on the keyboard the name of the option. It also allows users without the knowledge of any classical programming languages to use computers [Fedra and Loucks, 1985].

Since all the factors mentioned above may play a role in the acceptance of a DSS by potential users, the advantages of adopting interactive computer graphics and a menu-driven user interface within a DSS are obvious. The style of the visual part of the interface has to be considered from the beginning of the project and will require a considerable design effort [Fedra and Loucks, 1985].

The reviewed literature provided no specific recommendations about graphical presentation and user interfaces for computer applications in water resources engineering. It seems that the developers of these interfaces follow the common sense in designing graphical displays, and that this usually gives good results. Some of the developers have background in water resources engineering, which provides them with additional insights into the needs of the final user [Taylor and French, 1985]. The development of interactive computer graphics applications by non-programmers is made possible by the availability of numerous user-friendly software tools for the task.

The selection of software and hardware for these interactive computer graphics interfaces does not seem to be an issue. It is usually predetermined by the equipment and computational models already in use within an environment where a DSS is being developed. Fortunately, most of the equipment that is currently in use supports some kind of graphical presentation and compatible libraries of graphical routines are usually available.

3.3 Expert Systems

This thesis describes the second phase of the project initiated at the University of Manitoba and Manitoba Hydro in 1988. The first phase started by an extensive review of available literature about expert systems and their applications in engineering and other fields [Barlishen, 1989]. The research that followed focused on expertise in water resources, ways of eliciting this knowledge, and suitability of application of expert systems in the field. A brief summary of the results of the first phase of the project follows.

3.3.1 Knowledge Acquisition

Eliciting knowledge may take the largest portion of the development time for an expert systems project. Present methods of acquiring expertise through human communications are slow and inefficient. These techniques involve a combination of observing the expert solve problems, questioning him or her directly, and relying on the expert to perform some degree of formalization of his or her thinking process. There are no strict guidelines for obtaining a successful transfer of knowledge in an engineering environment. The success depends on the ability of the expert to articulate and explain the necessary information, and the ability of the inexperienced person to learn and understand.

The first task of an expert system developer is to learn about the domain. Since the expert's time available for the project is usually very limited, the best way to begin is by examining the available static knowledge from books, manuals, and reports before attempting a direct discussion. Experience can also be gained by running any relevant computer models. The second task is the development of a small prototype system that will be carefully expanded to deal with more complexity and expertise as the expert system grows. This incremental approach [Barlishen et al., 1990] requires selection of an appropriate scope for the initial

prototype. Finally, through knowledge acquisition sessions, the knowledge engineer attempts to describe the manner and resources with which the expert works in order to obtain an acceptable solution.

Both phases of the project described in this thesis focussed on the same case study. The linear programming optimization model called EMMA is used at Manitoba Hydro to optimize planned short-term and long-term operations of the system composed of a number of reservoirs, lakes, natural and controlled flows, and hydro and thermal generating stations. The first step in the research was to identify different kinds of expertise and knowledge that the expert that uses EMMA in Manitoba Hydro needs in his work. These include: (i) knowledge of the relevant technical aspects of the physical system; (ii) knowledge about how to consistently represent the physical and time structure of the problem in the analytical model he uses, (iii) knowledge about the algorithm used by the model, and (iv) knowledge necessary for interpretation of the model output.

The complexity of the domain and time constraints prohibited expert system developers from gaining quickly the full understanding of the problem domain. However, two appropriate problem areas were identified as potential applications for prototype expert systems, "intelligent" pre- and post-processing of data for the model. The prototype pre-processor is an expert system designed to perform "intelligent" editing, comparing important input parameters to historical data. The user is advised about potential problems with the values set for inflows, reservoir levels and energy production coefficients. On the output side, the post-processor expert system deals with feasible and infeasible solutions. For feasible solutions, it helps the user assess the practicality of the operating plan by looking at historical reservoir releases and levels. For infeasible solutions, the prototype expert system post-processor tries to identify all possible causes.

At first, both the expert and the system developers were a little discouraged by the lack of significant progress in the knowledge acquisition. All the difficulties experienced were attributed to the lack of problem understanding by the members of the research team. It was only by developing the prototype expert systems that a more suitable and achievable project focus was identified. Joint effort by the expert and another Manitoba Hydro engineer and the members of the team indicated that the initial expert system development scope appeared to have been too narrow. Thus, the initial effect of knowledge acquisition was a learning experience for everyone involved.

In order to perform an effective knowledge acquisition procedure in an engineering environment, a set of recommendations has been prepared and presented by Barlishen et al. [1990].

The development team:

1) Persistent and patient people who will ask questions, understand the problem, and continually demonstrate what they have learned should be selected for members of the development team. Background engineering experience in the domain field is very valuable.

2) If the team is relatively inexperienced, start with a small, simple, preferably structured, problem to gain experience.

3) Considerable effort should be invested in determining a model of the expert's problem solving process. Try to separate the problem into modules, so at any time only a portion of the expert's expertise will be under consideration.

4) Strong emphasis should be placed on incremental development. Start with common problem scenarios and build in complexity and specific cases.

5) Expert systems are advice providers, therefore, help and explanation facilities are very important.

6) Long development times should be expected. Constantly evaluate progress and do not be afraid to start again.

The expert system developer:

1) A co-operative, enthusiastic expert, who is aware of the large time commitment required should be selected. Educate the expert about expert systems, what to expect and when.

2) Learn about the domain. If qualitative information is required, make sure that the expert's view of the terms is clear and make a user aware of this interpretation. Establish not only the knowledge an expert possesses, but also how and when it is used. The knowledge implicit in rules should be made explicit to the user through help and explanation facilities.

3) Knowledge acquisition is an iterative process rather than a linear one. Knowledge is elicited piece by piece. Rules are often the most natural and understandable form of knowledge presentation for engineering problems .

4) A regular meeting time should be established with the expert, even if some meetings turn out to be more of coffee-break discussions. Tape record the sessions. Try to fill in the gaps in your interpretation of the expert's reasoning through continued questioning or criticism of prototypes. Procedures used to obtain the expert's knowledge depend on the specific problem and expert involved. Acquisition process should rely on a combination of observing the expert solve problems, and discussions and questioning to analyze what was done. The best way to establish rules is by working through several problems with the expert and examining the differences and similarities between them. Considerable experience can be gained by "playing around" with the tools or techniques used by the expert. It seems that the only way to uncover true expertise is to interpret what was learned in a session by establishing rules or a small prototype system. An expert may find

it easier to criticize than to explicitly say what is needed. As sessions proceed, an idea will be developed as to which knowledge acquisition approaches work best with the expert.

3.3.2 Expert System Development Tools

In parallel with the work on knowledge acquisition, different tools for building expert systems were examined. The research focused on three tools: RuleMaster 2 Unix version by Radian Corporation, PCPLUS by Texas Instruments Inc., and MPROLOG by Logicware Inc.. Four basic criteria were chosen for the evaluation of the tools: (i) extensibility, (ii) clarity, (iii) flexibility, and (iv) efficiency. Extensibility was defined as the capability of the tool to interact with other software. Clarity was the measure of how easy it was to understand and use the tool. Flexibility was concerned with the ability of the tool to adequately represent knowledge, and to reason about the knowledge base. Efficiency was defined as the response speed of the tool.

The extensibility of the tools varied considerably due to the use of different hardware and programming languages in their development. RuleMaster 2 produced expert systems written in C language, which allowed for integration with the analytical model and direct access to the data structures used at Manitoba Hydro. PCPLUS, based on PC Scheme language (a subset of LISP language), could not be integrated in a similar way due to the lack of a version that could be run on the Apollo workstations used in Manitoba Hydro.

The clarity of the three tools was an important issue because none of them came with a "real" and comprehensive example illustrating the tool's use. In their original documentation, both RuleMaster and PCPLUS provided only very simple examples, that did not even begin to exploit all their capabilities. MPROLOG's documentation provided only program fragments and did not go much beyond the

examples without any practical application. Since its documentation was the most complete, PCPLUS appeared at first to be the easiest to understand of all three tools. As the members of the team gained more experience, it became much more convenient and efficient to use RuleMaster 2 and its editor.

Flexibility of the tools was not, for the most part, a major concern. Both RuleMaster and PCPLUS produced essentially what can be referred to as rule based systems. The knowledge representation scheme of PCPLUS was enhanced by the tool's ability to make it frame based. This allowed organizing related rules into frames, which added more structure to the knowledge base and enabled more efficient search during a consultation. MPROLOG, being based on Prolog language, was very flexible with respect to knowledge representation. The reasoning schemes, or search mechanisms, provided by the three tools varied to some extent. RuleMaster produces expert systems that search their knowledge base using both, forward and backward chaining. In the Manitoba Hydro case study, forward chaining was the most natural choice since the two prototypes were essentially diagnostic in nature. PCPLUS uses a backward chaining search strategy although it could also search in the forward direction. MPROLOG has the built in backward chaining search technique.

The explanation facility of any development tool is very important due to the fundamental explanatory role of expert systems. The explanation facility of PCPLUS was the most complete. The one provided by RuleMaster was ambiguous at times although it could generally produce good explanations during a consultation. MPROLOG required that the developer provide the explanation facility.

The efficiency of the tools was a concern only with PCPLUS. An earlier version of a post-processor expert system prototype was implemented on an IBM

AT class machine, and the response time of a consultation was considerably longer than for other tools.

The tool ultimately selected for further implementation of the expert system was RuleMaster 2. The choice was made based on this tool's ability to produce highly integratable, portable, and efficient C language functions, and its relatively convenient development facilities.

CHAPTER 4.
MANITOBA HYDRO CASE STUDY

4.1 Physical Characteristics of the Manitoba Hydro System

Manitoba Hydro is a Provincial Crown Corporation with the responsibility of operating the integrated power system of Manitoba and Winnipeg Hydro. The integrated generation system is composed of 13 hydro-power plants with a total capacity of 3548 MW and 3 thermal plants with a total capacity of 404 MW. Manitoba's supply is augmented by a seasonal diversity agreement with Northern States Power Company (NSP) which provides 300 MW of winter capacity until 1993. The 1330 MW Limestone Generating Station is under construction, with first power expected later this year [Manitoba Hydro, 1989].

More than 90% of the total system capacity is hydro, as shown in Table 1. Because of such a strong dependence on the hydro-electric generation, a relatively large variation in the annual hydro energy potential is the main characteristic of the system. The annual production varies between 26000 GWH for good water conditions and 16000 GWH in drought conditions, while the average is 22000 GWH. Because the major objective of the utility is to meet the load under drought conditions, surplus energy is available in most years. Nine transmission lines to the neighboring utilities permit the interchange of surplus energy and the improvement of the system reliability.

The hydraulic system receives water from the Nelson River basin, shared by the provinces of Alberta, Saskatchewan, Manitoba and Ontario in Canada and the states of North Dakota and Minnesota in the United States, and the Churchill River Drainage basin, shared by the provinces of Alberta, Saskatchewan and Manitoba. The flow from the Churchill River at Southern Indian Lake has been diverted through the Rat and Burntwood Rivers to the Nelson River hydro-power stations.

The hydro-power plants of Manitoba can be grouped into three subsystems: (a) the Winnipeg River system which receives water from the Lake of the Woods and Lac Seul; (b) the Grand Rapids plant on the Saskatchewan River; and (c) the Nelson River system which receives water from both previous systems through Lake Winnipeg and an additional flow from Southern Indian Lake. The lakes, although shallow, are capable of storing huge amounts of water due to their large areas.

Table 1. 1992 Manitoba Hydro Operating Capacity (MW)

Winnipeg River (6 generating stations)	560
Grand Rapids	440
Jenpeg	10
Kelsey	216
Laurie River (2 generating stations)	10
Kettle	1200
Long Spruce	970
Limestone	1330
TOTAL HYDRO	4828
Brandon and Selkirk Thermal	369
NSP Diversity	300
SYSTEM CAPACITY	5497

Source: Manitoba Hydro, 1989

It is characteristic for the Manitoba system that high river flows occur in spring and summer while high demands for power occur during the winter. The lakes are normally capable of storing all flows upstream of generating stations except during the high spring runoff. Due to these specific circumstances, accurate short- and medium-term flow forecasts for the system may be obtained from upstream flows, water levels in the lakes, and precipitation measurements.

The generating plants on the Nelson River and the Winnipeg River cannot store large amounts of water in their forebays. The releases from the plants are determined by the operation of the distant upstream lakes and by natural flows. Therefore, they are operated as run-of-river plants without high head variations. The other aspect of storing water far from the power plants is that significant time has to pass before the released water can be used for generation of energy. Some of the reservoirs in the system are operated exclusively for power production, while others are multipurpose. All the lakes and reservoirs in the system are subject to regulatory limits.

4.2 Reservoir and Energy Scheduling Engineer

The operations planning of the Manitoba Hydro system is performed in the System Operations Department by the Reservoir and Energy Scheduling (RES) Engineer. The purpose of operations planning is the preparation of an operating schedule of reservoir releases, hydro and thermal generation, and imports and exports of energy [Cormie and Barritt-Flatt, 1987]. In preparing the plan, the objectives that must be considered by the RES Engineer, in order of priority, are to:

1) ensure that winter energy supply exceeds the forecasted Manitoba load even under the most severe winter weather conditions;

2) ensure that sufficient summer energy is supplied;

3) ensure that reserve energy is available within the hydraulic system to satisfy any firm energy commitments in subsequent years, under the worst drought conditions;

4) maintain winter operating reliability to avoid a system blackout during winter months;

5) maintain summer operating reliability to avoid a system blackout during summer months;

6) meet social responsibilities to other lake or reservoir users, e.g. environmental requirements of fish and other wildlife, concerns of other businesses and recreational users regarding lake levels or fluctuations in levels, and agreements with outside control boards;

7) operate as economically and efficiently as possible.

The RES engineer also has to take into account various constraints imposed on the hydraulic and electrical systems. Physical constraints include water and fuel supply, generating station capacities, and sizes and rating curves for reservoirs. Regulatory constraints include licences, agreements, and environmental orders restricting actions that can be taken at certain sites. Practical constraints deal with how close to the hydraulic and electric limits the system can be operated.

The operations plan is derived on the basis of separate studies of flow forecasting, power demand forecasting, export and import energy market forecasting and system maintenance. The role of the RES Engineer in the System Operations Department is to synthesize the above information and to determine the operation decisions keeping in mind the set goals. The derived operations plan is reviewed regularly and updated with new forecasts of precipitation, river flows, domestic power demands, and export market prices as they become available. Beside the hydrological and energy market situation, the operations plan has to take into account the availability of the generating equipment. Maintenance required by the generating units in the system needs to be scheduled to minimize the impact on revenue while maintaining system reliability levels. The prepared operations plan is used as a guide to the real time hourly operation of the system, where the specific actions are implemented.

In order to efficiently make decisions, the RES Engineer at Manitoba Hydro has used a number of different tools. In the early period before 1970 extensive use of simulation was characteristic for reservoir management at Manitoba Hydro. Long term simulation programs were written for each of the major river systems with the operating rules developed and refined through trial and error until a satisfactory result was obtained. Familiarity with the simulation results gave the RES Engineer an understanding of what could be expected from the river system and what constituted normal operations. Several different computer programs were required with many hand calculations. Good judgement and experience was required to minimize the effort. The typical time to prepare a complete schedule for budgeting purposes was one month [Cormie and Barritt-Flatt, 1987].

The development of the Nelson River since the 1970's has changed the nature of the Manitoba Hydro system and more than doubled its generation capability. The hydraulic system has become much more complex with the regulation of Lake Winnipeg and the diversion of the Churchill River into the Nelson River. During this period, the public has become demanding and more aware of the system operations, which has resulted in the need to incorporate interests other than hydro energy generation in the planning and operations of the utility. The growing demands and increasing complexities of the system have placed an increasing burden on the RES Engineer.

The traditional techniques of analysis to aid in decision making were not capable of analyzing the system as a whole with all of its interdependences and growing complexities in the timely manner required for operations. In order to improve and speed up the analysis required by the RES Engineer, the development of the Energy Management and Maintenance Analysis (EMMA) optimization model was initiated at Manitoba Hydro in 1981.

4.3 Energy Management and Maintenance Analysis Program

The role of EMMA in the operations evolved as the confidence in the model's results grew and as the computer program became more sophisticated. The comprehensive character of the tool provided a great deal of flexibility in its application to different operations problems. The scale, complexity, and impact of the system whose operation is optimized by EMMA raises a number of political, environmental, and social issues. Accounting for all of these issues in the model requires much knowledge, experience, and intuitive reasoning that cannot be learned from books. Implementation of the EMMA model requires a thorough understanding of the details related to its solution algorithm, preparation of massive input, and interpretation of output. To date, only one person at Manitoba Hydro has the required experience to properly formulate and analyze a planning problem using EMMA model. Since the use of EMMA is so expertise-intensive and since the interest in formulating and capturing this expertise was very strong, the project described in this thesis concentrated on the development of a decision support system around EMMA. The potential benefits were seen in improving and facilitating the expert's work, as well as making transfer of his knowledge to inexperienced colleagues faster and easier. A detailed description of EMMA, necessary for understanding of the conducted knowledge acquisition process and the developed IDS tool, follows.

EMMA is a deterministic optimization model that uses linear programming in order to optimize planned short-term or long-term operations of the system. EMMA calculates schedules of reservoir releases, power generation, maintenance, and energy imports and exports for a specified stream flow and load scenario and within a specified time horizon conveniently divided into time steps. The algorithm used by the program ensures that loads are met while maximizing revenues and minimizing costs. Since EMMA is a deterministic model, the expert performs

appropriate adjustments in input data and repeatedly runs the program in order to account for the stochastic nature of the problem. EMMA's intrinsic complexity stems from the nature of modelling operation of the interconnected power utility, a task that includes dealing simultaneously with three distinct subsystems: the hydraulic network, the electrical network, and the maintenance subsystem.

The hydraulic network is composed of reservoirs, lakes, and rivers. Reservoirs generally have two possible outlets, a penstock that routes the flow through a power plant, and a spillway. Lakes may have natural outlets or they can be controlled by control structures. The storage capacity of lakes and reservoirs can be discretized using horizontal segments in order to represent more accurately stage-storage curves and stage-discharge relationships. The uppermost segment can further be divided into smaller horizontal parts called figments for a more accurate calculation of the lake level and the volume of water in storage.

There are two types of rivers in the hydraulic system, inflows into the system's lakes and reservoirs and flows between them. Inflows are described by the flow scenario. Flows between lakes and reservoirs can be controlled by one of the following three elements: a natural outlet, a control structure, and a hydro generating station.

The following hydraulic constraints are taken into account by EMMA:

1) **sum of figments:** Sum of figment storages is equal to the storage in the last segment;

2) **flow continuity:** For any lake or reservoir, the change in storage is equal to the difference between the inflows (all natural inflows, and penstock, spillway, gated control, and natural outlet discharges from any upstream lake or a reservoir) and the outflows (one or more penstock, gated control, spillway, and natural outlet discharges);

3) **run-of-river:** Operation of a generating station with another generating station upstream and no reservoir between them is coordinated in order to account for the fact that water cannot be stored between these two stations;

4) **same stage:** If necessary, the initial elevation of a lake can be set to be equal to the elevation at the end of the planning period;

5) **generating station flow bound:** The outflow from a spillway or a powerhouse is limited by a function of the upstream lake level; and

6) **control structure flow bound:** The outflow from a control structure is limited by a function of the upstream lake level.

The electrical network consists of hydro and thermal generating stations, transmission lines for the exchange of energy with the adjacent utilities, the domestic load given by the load scenario, and energy import and export markets. The transmission lines are referred to as tielines by the RES Engineer at Manitoba Hydro and the same term is used in this thesis.

The domestic load is represented by a load duration curve for each time step. The load duration curve represents intensity of the load during a time step reorganized in descending instead of chronological order. In other words, for a certain load, the curve specifies the time (or the fraction of the total time step) during which the load will be exceeded. Load duration curves are approximated by a number of strips and the instantaneous peak. The concept of the instantaneous peak load ensures that there is enough generating capacity in the system to meet the consumption when it reaches its highest value during the time step. The strips ensure that the system produces the required overall amount of energy during the time step. There is at least one on-peak and one off-peak strip to account for varying energy prices according to these classifications.

An example of a load duration curve representation using one off peak and two on peak strips is shown in Figure 2.

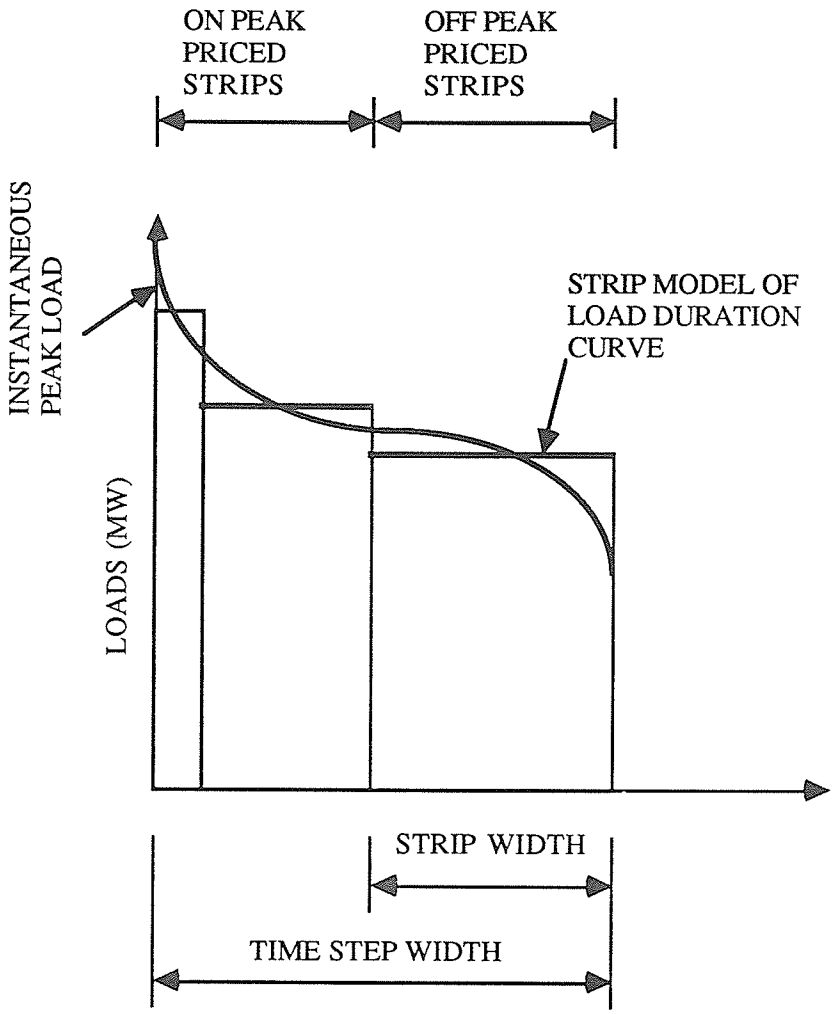


Figure 2. Load duration curve approximation using 2 on-peak and 1 off-peak strips (after Barritt-Flatt and Cormie, 1988)

The tielines to other utilities are used for firm or interruptible energy imports and exports. A firm purchase or sale of energy has to be supplied through the contract time, and it is therefore treated as a constraint by EMMA. Interruptible energy exports and imports are a part of the optimization process and are only limited by the available tieline capacity.

The constraints that can be formulated for the electrical system are:

1) **energy supply and demand:** For every load duration curve strip, the system load plus firm exports are supplied from hydro generating stations, thermal generating stations, and imports;

2) **hydro production:** The powerhouse discharge over a time step is converted into energy for each strip of the corresponding load duration curve;

3) **hydro shaping:** The capacity utilized at a hydro generating station decreases moving from the on peak to the off peak part of the load duration curve, or that it remains the same if the station is base loaded;

4) **generation source shaping:** The capacity utilized at a thermal generation source decreases moving from on peak to off peak part of the load duration curve, or that it remains the same if the station is base loaded;

5) **peaking:** The maximum change in storage or drawdown in a reservoir caused by the peaking downstream hydro station during on peak strips in a time step should not exceed the maximum permitted value;

6) **tieline total load:** The sum of import and export time does not exceed total time for each tieline;

7) **import fraction limit:** The total interruptible (not firm) import energy is limited to a percentage of the required system energy for each time step;

8) **tieline minimum energy:** A minimum amount of energy has to be delivered for firm energy contracts;

9) **grouping constraints:** The total energy production at a group of selected energy sources or other energy variables (hydro and thermal stations, imports, and exports) over strips, time steps, or the entire time horizon can be made larger than a minimum or smaller than a maximum value expressed as a function of the system load;

10) **waste heat:** The maximum capacity of a thermal plant is limited when the thermal pollution of the cooling water exceeds a specified maximum temperature or maximum temperature change.

The maintenance system is modelled in order to allow for an annual maintenance schedule that will take into account maintenance requirements of all stations as well as system operation requirements. It also enables the RES engineer to constantly review the predetermined maintenance schedule. EMMA can model preferences of station staff for maintenance periods. The following constraints are taken into account:

1) **crew scheduling:** For each maintenance crew, holidays have to be accounted for;

2) **required maintenance:** Sufficient maintenance has to be provided for each generating station;

3) **maintenance space:** The peak load has to be met after maintenance and the possibility of forced outages are accounted for;

4) **available capacity:** The available capacity for each generating station is not exceeded by the sum of energy generated and scheduled maintenance;

5) **crew availability:** Maintenance done by a crew within a time step does not exceed their availability in hours.

The electrical and hydraulic systems interact where a generating station is located on a river. Maintenance comes into play when a generating station or a tieline goes out of service and cannot be used for a certain period of time.

The objective function maximized by EMMA adds revenues from sales of energy and the value of water left in storage at the end of the time horizon, and it subtracts costs attributed to energy production and imports, water releases, and

maintenance. These revenues and costs can be used to reflect a number of issues [Cormie et al, 1988]. Positive cost coefficients for water in storage may reflect its expected future value for energy generation, while negative values may account for possible flood damages that may occur if upper reservoir segments are full. Cost coefficients attributed to energy may reflect fuel and maintenance costs for the plants. Export and import energy cost coefficients can be used to describe the structure of the interruptible market and the firm contract energy prices. The cost coefficients associated with scheduled maintenance may be used to reflect non-economic scheduling preferences, and coefficients attributed to releases can express benefits and costs associated with flow assignments.

EMMA handles the nonlinearities present in the problem using piece-wise linearization and iterative approximation. Piece-wise linearization is used, for example, in approximating the stage-storage curves for the reservoirs. Iterative approximation is needed to model the nonlinearities of the nonseparable hydro production function. This is a multivariate function of the release and storage, which are both decision variables in the optimization process. In order to linearize the problem, a constant value is assumed for the production coefficients, defined as produced energy per a unit of release. Using the assumption of constant production coefficients an LP problem is formulated and the solution provides the values of releases during the planning period. The values obtained for releases are then used to determine new production coefficients for the next iteration. The process terminates when there is no significant difference between the assumed and calculated values of the production coefficients.

EMMA requires massive input preparation, and can produce a number of reports. There are 17 different kinds of data that have to be included in an input file:

1) job title that serves as a reminder of what the EMMA run is about,

2) report selection that determines which of the 30 possible reports are to be made during the program's execution,

3) linear programming parameters that control the precision and the execution of the LP algorithm,

4) network and constraint size data that define the total number of elements in the hydraulic, electrical, and maintenance subsystems,

5) time structure data that define the planning horizon and the time steps to be used in the problem,

6) load duration curves that give the load scenario for each time step,

7) lake data that describe each reservoir and lake in the system,

8) data that describe each hydro generating station,

9) data that describe each control structure at a lake outlet,

10) data that define the characteristics of each natural outlet in the system,

11) local inflow data that define the scenario of inflows into the lakes and reservoirs,

12) data that describe every generating station other than hydro in the system,

13) tieline data that describe all the interconnections with other utilities and the respective energy market characteristics,

14) generation strip groups data that, if necessary, group energy sources by a strip of the load duration curve, and impose a minimum or a maximum performance on these sources,

15) generation time groups data that, if necessary, group energy sources in all strips of the load duration curve by a time step, and impose a minimum or a maximum performance on these sources,

16) generation period groups data that, if necessary, group energy sources in all strips of the load duration curve over all time steps by a period, and impose a minimum or a maximum performance on these sources, and

17) data that describe each maintenance crew in the system.

4.4 Expert system content

The preceding phase of the project, conducted by Barlishen [1989], that focused on knowledge acquisition in water resources engineering and used EMMA as a case study, greatly contributed to the phase of the project described in this thesis. It established a knowledge acquisition procedure that consisted of regular, tape recorded, weekly interviews with the expert from Manitoba Hydro. It also provided the expert with experience in formulating step-by-step procedures and organizing the knowledge he uses in his work to forms that can be transferred into computer codes.

At the beginning of the project, the knowledge engineers explored the EMMA algorithm and numerous constraints and decision variables it incorporates. They were discouraged because this did not enable them to emulate the expert's work and to obtain useful results by running the program.

The problems were overcome when the expert, helped by another colleague from Manitoba Hydro, recognized that EMMA can be used to perform one of the following three tasks:

1) to provide a basis for preparation or update of the annual budget,

2) to prepare a weekly operating schedule for energy generation, imports and exports, and

3) to evaluate benefits from installing additional capacities, or to examine some other change in the long term operation of the system.

In this thesis, these three tasks are referred to as motivations or contexts for running EMMA. Once the knowledge engineers realized that the main part of the expertise is not in running EMMA, but in performing one of the three tasks and using EMMA as a tool, they could proceed with the knowledge acquisition successfully.

Further interviews with the expert revealed that different sequences of EMMA runs have to be performed in order to obtain useful results within each of the three contexts. All of these runs have to be made with appropriate adjustments in input data and with attention to particular parts of EMMA output.

Major breakthroughs were made in a short period of time when the knowledge engineers concentrated on underlying system analysis and probabilistic principles and tried to understand what the expert was trying to accomplish with different EMMA runs. Once EMMA was understood as just another tool that the expert uses in performing useful tasks, it became easier to understand the different data adjustments the expert makes, and how they interact with the LP algorithm in order to handle the complexity and the uncertainty of the system whose operation is optimized.

One important conclusion about the development of expert systems may be drawn from this experience in knowledge acquisition: the project is more likely to succeed if it concentrates on a particular task performed by the expert instead of trying to capture a body of knowledge related to a tool or a problem area. Current methods of knowledge acquisition, which rely mainly on communication between experts and knowledge engineers, are better suited for revealing steps of a problem solving procedure along with the accompanying necessary criteria, than for covering a broad area of knowledge. The latter will usually make it extremely hard for the expert to organize

and formulate relevant knowledge, and almost impossible for the knowledge engineer to achieve satisfactory understanding of the explored domain. The first, and by no means easy, step in the development of an expert system is to define exactly what it is supposed to do.

Only one expert in Manitoba Hydro has the necessary expertise for the correct use of EMMA. The first year of the project was spent in examining the ways of eliciting this knowledge. The first prototype expert system appears to have been focused on a too narrow area of intelligent data pre-processing for EMMA [Barlishen, 1989]. Choosing one of the three "real" expert's tasks explained on pages 40 and 41 would have been more successful and less frustrating for the knowledge engineers. This thesis focuses on the second phase of the project. This phase started with the knowledge acquisition that covered the three motivations for running EMMA. The acquired knowledge is presented in the following section of this chapter.

4.4.1 Firmness Test

In performing any kind of analysis of the system's operation, the RES engineer has to take into account the possibility that contingencies, such as major outages or severe droughts, can cause problems in supplying energy to customers. In order to meet the objectives outlined in Section 4.2, the RES engineer has to ensure that the system has energy reserves available at all times to cope with these contingencies. In the use of EMMA, sufficient energy reserves can be maintained in the system by planning for enough energy in storage at the end of the planning horizon. The RES engineer has to represent the system in the model in the way that will ensure that the optimal policy formulated by EMMA will not reduce these reserves.

The amount of reserve energy held in reservoir storage at the end of the time horizon is determined by performing the procedure known as the firmness test. This procedure has to be performed before running EMMA because its results are important for correct preparation of input and interpretation of results. The firmness test consists of the following three parts:

Firmness Test - Part 1:

The first step in performing the test is to evaluate potential scenarios the system has to be prepared for. The scenarios examined in most cases are major outages or severe droughts.

Outages to generation or transmission facilities that affect the energy supply should be considered when setting energy reserves. An outage should be a low probability event.

A drought will probably be the contingency with the largest affect on the energy supply to a hydro utility. The larger the hydraulic component of the system the larger the energy reserves necessary for drought. Reserves are usually evaluated for droughts outside the planning horizon. For outages, if they occur in the planning period, drought reserves may have to be used. Drought reserves are usually set to withstand the worst drought on record, thus making sure that the system will have enough reserves to meet energy demand even if the worst sequence of low flows on record should occur. These reserves also protect the system from outages and other contingencies because the probability of simultaneous severe droughts, long-lasting outages, and spells of extremely cold weather with high consumption of energy is especially small. The worst drought on record is identified by the RES engineer at Manitoba Hydro by performing the following three step procedure:

1) Calculate the total monthly energy production in the system for all available flow records using only inflows into the system and not water drawn from storage. Add thermal generation and committed imports to the calculated series of hydro generation .

2) Compare the energy supply from Step 1 with the corresponding monthly demands for energy and identify droughts as periods when there is deficiency of energy.

3) The worst drought on record corresponds to the period with the largest cumulative energy deficiency calculated as the sum of deficiencies from the beginning until the end of the drought. The corresponding sequence of of monthly energy production from Step 1 will be used in the following parts of the firmness test.

Firmness Test - Part 2:

The second step in performing the firmness test is to calculate the amount of necessary energy reserves for the end of the planning period. This is accomplished using the following four step procedure:

1) Monthly energy demands have to be estimated and plotted for the time span extending over the planning period in EMMA plus a couple of durations of the worst drought on record beyond the planning time horizon.

2) A scenario with the occurrence of the worst drought on record should be assumed for each of the years covered by the time span from the Step 1 above. For each of these scenarios, monthly energy generation, available from the Step 3 in the Firmness Test - Part 1, should be plotted against the corresponding monthly demands from Step 1 above.

3) For each scenario from Step 2 above, energy deficiencies for each month should be calculated by subtracting the estimated demands from the Step 1 above from the available energy from the Step 2 above.

4) Going backwards in time until the end of the planning time horizon to be used in EMMA and summing up energy deficiencies, it is possible to calculate reserves that are necessary in the system for each of the scenarios from the Step 2 above. The largest of these values corresponds to the worst scenario the system should be protected from. This reserve is the result of the firmness test and it has to be left in the system at the end of the time horizon in each EMMA run.

Firmness Test - Part 3:

The amount of energy evaluated in Firmness test - Part 2 is only available to the system at the end of the planning horizon in EMMA runs. If the system relies mainly on hydro generation, this usually means planning this energy in storage in its lakes and reservoirs. The problem of allocating the necessary energy reserves between the reservoirs in the system can be solved in the following order:

1) Multipurpose reservoirs are targeted not to drop below their normal levels corresponding to the end of the planning time horizon.

2) Energy in storage between the normal level and the minimum level is calculated for each of these multipurpose reservoirs. Thus allocated reserves are summed.

3) The remaining energy reserves, if there are any, are allocated iteratively to all other reservoirs of the system, filling them gradually in the way that minimizes the probability of spillage. The probabilities of spillage can be calculated based on the historic records for each reservoir. When all of the necessary reserves cannot be allocated to the reservoirs of the system additional

generation capacities, new reservoirs, or water diversions are necessary in order to firm up the energy supply.

In preparing EMMA input, the reserves allocated in the firmness test are given the highest value exceeding any other source of energy or export value in the system. This forces the LP algorithm to keep these reserves in the system until the end of time horizon, and to use all possible alternatives before it starts using them to meet the energy demands. If energy reserves are required to meet the firm energy demands reservoirs will be drawn below target levels at the end of the planning horizon. This is observed by examining the levels of these reservoirs in the program's output. EMMA takes care of the firmness test reserves through their values in the objective function instead of treating them as constraints. This way, the program provides clear solutions in the cases when the firmness test reserves are reduced at the end of the time horizon. Treating these reserves as constraints would result in infeasibilities , which are hard to trace due to the large overall number of constraints in EMMA.

4.4.2 Preparation of Annual Budget

The following 5 steps are necessary in order to obtain results that can be used in preparation or updates of the annual budget:

1) The firmness test should be performed in order to evaluate reserves that are necessary in order to ensure reliable supply of energy to the customers in case of a severe drought or outage in the system. This test is performed just once, and outside EMMA, for the specified time horizon and its results are used thereafter for all subsequent runs of the program.

2) The scenario called the severe case should be examined first. This means that a low water supply (say 95% probability of exceedence) and an early

freeze-up date are assumed. This run formulates a release and energy production plan that satisfies the system demand in cases where operating conditions are severe. Keeping firmness test reserves ensures that the system will be capable of coping with contingencies if they should occur in addition to these severe conditions.

3) The scenario called the **expected case** should be examined next. This means that the water supply with 50% probability of exceedence and the expected freeze-up date are assumed. This run formulates a release and energy production plan that satisfies the demand in cases where the likeliest flow and load scenario occurs. Hydro generation will be larger in this scenario than in the severe case because more water is available in the system. This will also result in smaller expenses for non-hydro energy supplies. However, an annual budget cannot be prepared based on the results of this run because the system should be able to perform if the severe scenario occurs.

4) The severe case should be run repeatedly, gradually increasing the lower bounds on hydro generation for the first three or four time steps. This is accomplished by adjusting the part of the input that describes the hydro generating stations of the system. This process should be stopped when these increased energy outputs from hydro stations reach the corresponding values from the expected case described in the Step 3 above. If EMMA starts reducing the system's reserves determined in the firmness test before this happens, the severe case runs in this step should be stopped. The idea behind this step is to increase hydro generation and releases in the immediate future for the severe scenario, and compensate for that with expensive thermal generation in later time steps. This way, the system will perform economically in the immediate future, with a small probability, equal to that of the occurrence of the severe scenario, of paying penalty for this in later time steps. In most cases however, inflows into the system will be larger than predicted

in the severe case, and the system will avoid penalty. When hydro generation for the immediate future in the expected case and a severe case re-run coincides, the budget can be prepared based on the expected case run.

5) The final step is called the convergence test. Levels in the system's reservoirs and forebays operated according to operational rules should now be re-calculated manually, based on the releases from the latest EMMA run. EMMA should be run with these levels in order to make sure that the operating policy converges. The levels are constrained by making appropriate adjustments in the part of input that describes lakes of the system.

The time horizon in all these EMMA runs lasts till to the end of the fiscal year because that period is relevant for the preparation of the budget. The RES engineer uses 12 time steps, each 1 month long. In updates of the annual budget, if they are required, the time horizon remains the same and the number of time steps decreases because the beginning of the time horizon is always the current moment in time. Initial lake levels are constrained to be equal to the actual observed levels in the system. Final levels are left free to allow the LP algorithm to optimize them.

The resulting energy production, reservoir levels, and other physical data from the final run of the program can be used in preparation of the annual budget. The objective function in EMMA gives only relative dollar values that can be compared only between different EMMA runs within one context. Therefore it cannot be used in preparation and updates of the annual budget.

4.4.3 Operational Plan

The use of EMMA for the purpose of preparing an operating plan for releases, generation of energy, imports, exports and maintenance depends on whether the system is in a drought or not. The system is in a drought if reserves in

the reservoirs are below usual and inflows are not big enough to enable sufficient energy production, which makes it necessary to use water in storage.

The five step procedure used for preparation and updates of the annual budget can be used in this context if the system is in a drought. The RES engineer uses the following simplified 3 step procedure when the system is not in a drought.

1) Firmness test should be performed as described in Step 1 in Section 4.4.2. Since the operating plan is prepared every week, and the firmness test reserves are stable, this step need not be performed every time. Once evaluated, the firmness test reserves can be used for all subsequent EMMA runs during the year.

2) The expected case, as described in Step 3 in Section 4.4.2, should be run. Since the conditions in the system are favorable and since this cannot change rapidly, the RES engineer knows that the available reserves are plentiful and that the severe case run is not necessary. Keeping firmness test reserves ensures that the system will be capable of coping with contingencies.

3) Finally, the convergence test, as described in Step 5 in Section 4.4.2, should be performed based on the expected case EMMA run from step 2.

The time horizon in all these EMMA runs lasts to the end of the fiscal year. The RES engineer uses 3 or 4 initial time steps, each one week long, for immediate future because they are the most important for planning of the operation of the system and because the flow and load data are more certain. For the following 2 or 3 time steps, bi-weekly periods are used, and monthly time steps are used thereafter for the entire period until the end of the time horizon. Initial lake levels are constrained to be equal to the actual observed levels in the system. Final levels are left free to allow the LP algorithm to optimize them.

4.4.4 Long Term Study

EMMA can be used to evaluate benefits of introducing additional hydro or thermal generating capacities, a water diversion, or a new reservoir in the system. It can also be used to examine some other change in the long term operation of the system. The following 5 steps are necessary in order to obtain results that can be used in long term studies of the system's performance.

1) The firmness test should be performed in order to evaluate reserves that are necessary in order to ensure reliable supply of energy to the customers in case of a severe drought or outage in the system. This test is performed just once, and outside EMMA, for the envisioned future time horizon, and its results are used thereafter for all subsequent EMMA runs within the long term study.

2) The system without the examined change is first entered to EMMA and different sequences of flows and loads (5%, 25%, 50%, 75%, 95% probability of exceedence.) or other scenarios of interest are examined.

3) The convergence test, as described in Step 5 in Section 4.4.2, should be performed based on the EMMA runs from Step 2 above.

4) The system with the examined change, such as a new thermal generating station or a new reservoir, is examined next. This means performing EMMA runs for the same scenarios as in Step 2 above.

5) The convergence test, as described in Step 5 in Section 4.4.2, should be performed based on the EMMA runs from step 4.

All these EMMA runs are made for the time horizon of one year divided into 12 monthly steps. Both initial and final levels in the lakes and reservoirs in the system are left free in order to let the LP algorithm choose the optimum values. However, the initial and final levels at each lake are constrained to be equal to each other in order to make sure that the energy supply is determined from inflows and

available storage and not mined during the study period. This procedure gives a realistic comparison of the system's performance with and without the examined change, because it takes its optimal response in both cases. The resulting energy production, reservoir levels, and other physical data obtained for the utility with and without the examined change should be used for further evaluation of benefits and not values of EMMA objective function itself. The objective function in EMMA gives only relative dollar values that can be compared only between different EMMA runs within one context.

4.4.5 Preparation of Input

Extensive interviews with the expert covered all items in an input data set for EMMA. They revealed that a sophisticated knowledge base and a number of other models and special data preparation programs are used in order to prepare input for EMMA. The knowledge engineers tried to extract knowledge about basic concepts that are used in preparing these data and about the meaning and the role of each data item in the program's algorithm. All these explanations were encoded later into the developed computer application in order to assist a novice engineer in gaining knowledge necessary for a correct use of EMMA. Printouts of the screens that appear during a consultation with the expert system part of the developed IDS system can be found in Appendix A. At the current level of sophistication of the IDS system, these explanations look more like an extended version of EMMA manual. Their envisioned role is to serve as explanatory screens and provide the user with a broad understanding of the data and concepts used in EMMA.

The knowledge engineers in fact tried to supplement the EMMA manual [Cormie et al., 1985] with what they felt were necessary explanations about procedures for preparing data, hints about what values to use whenever possible, and the role of different items in EMMA input. Their background in civil

engineering enabled them to focus the knowledge acquisition on the areas where an inexperienced engineer would find problems in using the program. The acquired knowledge is fragmentary in nature and cannot be presented independently, without repeating the entire EMMA manual. The compiled information necessary for the use of EMMA that includes the knowledge elicited through knowledge acquisition can be found in Appendix A where the screen output of the developed expert system is given.

CHAPTER 5.
DEVELOPMENT OF DECISION SUPPORT SYSTEM

It has been pointed out in this thesis that, to date, only one person at Manitoba Hydro has the required expertise and experience to properly formulate and analyze the planning problem using available tools including EMMA. The basic motivation for this project was to evaluate potential application of expert systems and integration of computer graphics in order to assist the RES engineer. One of the main objectives was to capture some of the RES Engineer's expertise, experience, and intuitive reasoning that cannot be learned from books, and to encode it into a knowledge-based IDS system.

Potential benefits were seen in making the transfer of the RES Engineer's knowledge to inexperienced colleagues faster and easier. Consequently, the RES Engineer's duties could be shared and his valuable time could be used more effectively. In addition to this, successful development of an IDS would make the tools used exclusively by the RES Engineer available to other employees at Manitoba Hydro.

The IDS project was divided into two components: expert systems and computer graphics. The major emphasis of the IDS was on expert systems. The computer graphics component of the IDS system was developed to support the expert system and to illustrate the possible integration of these two tools. The benefits were seen primarily in: (i) facilitating data input and editing and (ii) improving the comprehension of spatial and time-dependent information.

The work on this intelligent decision support system was carried out on Apollo Domain Personal Workstations operating under a version of the Unix operating system.

5.1 Expert System Component

5.1.1 Knowledge Acquisition and Formulation

In order for an expert system to achieve any sort of acceptable performance level, the expert system developers have to correctly perform the procedure of acquiring and interpreting knowledge. The knowledge engineers started by a careful examination of available manuals and articles about EMMA. The knowledge acquisition procedure afterwards consisted of regular weekly interviews with the expert that were all tape recorded.

Early in the development of the expert system, it was realized that it was necessary to limit the knowledge acquisition and work on the expert system to a manageable part of the problem. The developers of the expert system concentrated on the use of EMMA in formulating release and generation policy that is used as the basis for preparation of the annual budget and its subsequent updates. This decision was made according to the principle of incremental development of expert systems [Barlishen, 1989], which means beginning with a prototype and slowly and carefully expanding the knowledge base to deal with more complexity and involve more expertise.

Formulation of the basic five step procedure described in Section 4.4.2 enabled the construction of a small prototype of the expert system. This sped up further development substantially because it provided a structure that was gradually filled with more expertise and advice, and it also enriched the communication between the expert and the system developers. By looking at the rules and knowledge that were already formulated and entered into the prototype, the expert could correct all the errors and misconceptions on the part of the expert system developers. This was an iterative process that required much patience and persistence on the part of both the expert and the knowledge engineers. For

example, more than four iterations were needed before the firmness test steps were formulated and the corresponding rules in the knowledge base refined.

The early introduction of the expert system prototype also enabled an early beginning of the verification phase of the expert system. The prototype system gradually grew and gave more detailed advice and explanations as new modules were added to it. The experience gained during the work on the budgetary branch of the expert system and the prepared modular structure of the prototype enabled completion of the other two branches of the expert system in a very short time. It also sped up acquisition of knowledge related to data preparation and general procedures that are used within all three contexts.

5.1.2 Structure of the Expert System

The pattern of EMMA use dictated the structure of the developed expert system, shown in Figure 3. The introductory part gives general information about EMMA and, if requested, basic concepts of linear programming. Further explanation about EMMA input and how the program uses LP in solving energy generation problem follows. These introductory screens can be found in Appendix A. With the help of computer graphics, the user can then compose and visualize the system he or she wants to analyze. This part of the expert system enables the user to put together basic elements used by EMMA, such as lakes, natural and controlled outlets, hydro and thermal generating stations in a fashion that is acceptable to the program.

The explanation of the firmness test, described in Section 4.4.1, which has to be performed in order to ensure sufficient reliability of power supply follows. This part of the expert system familiarizes the user with all the steps that have to be performed in order to evaluate reserves in the system necessary for this purpose. The expert system also provides guidelines on how to allocate these reserves to the

reservoirs of the system, and how to force the LP algorithm to leave these reserves untouched in formulating the optimal generation and release policy.

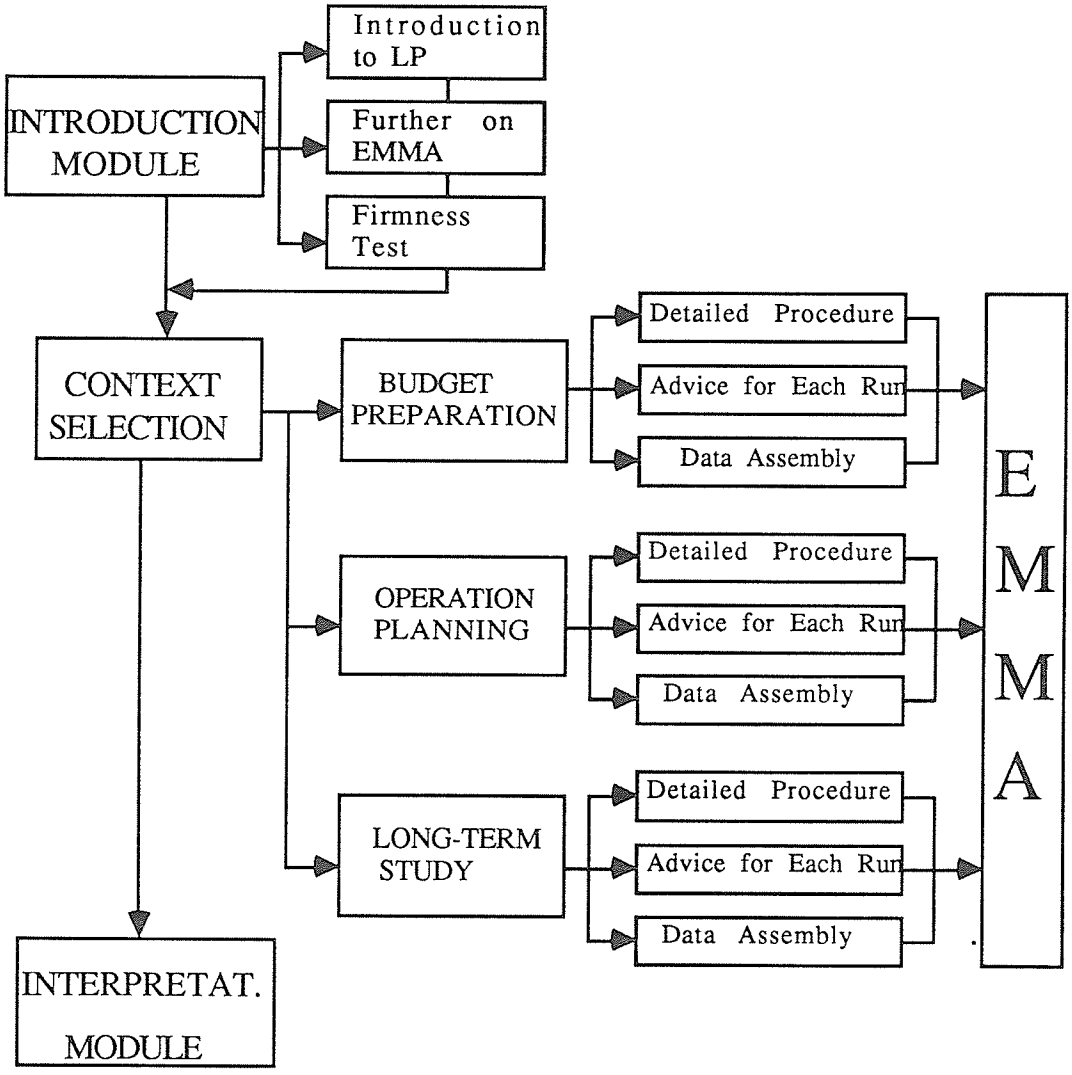


Figure 3. Structure of the developed expert system

The next part of the expert system familiarizes the user with the three basic motivations for the use of EMMA and what useful tasks can be performed by running the program. The user can then select a motivation, and the expert system guides him or her through the entire sequence of EMMA runs that have to be performed in order to obtain the desired results. The focus was on explaining the basic ideas and reasons for performing each step.

For each run, after explaining the reasons for performing it, the expert system follows the structure of input data for EMMA. It guides the user through each of the necessary 17 data modules explaining in detail what data to use and why. The user has the option of skipping data modules that do not change between consecutive EMMA runs within a context. More experienced users can select the option of going independently through necessary data modules.

After going through explanations of input data, the user can choose to stop the execution of the expert system and run EMMA. The explanation of how to analyze the output and prepare for the next EMMA run, if it is necessary, follows.

The nature of the expertise required to run EMMA resulted in the shape of the expert system consultations shown in Figure 4. For each context, the user is guided through the necessary EMMA runs that lead to the final run whose results can be used for the intended purpose. For each of these runs, the purpose and the underlying concept are first described, the seventeen groups of data necessary for running EMMA are explained, and any available advice on what values to use is given.

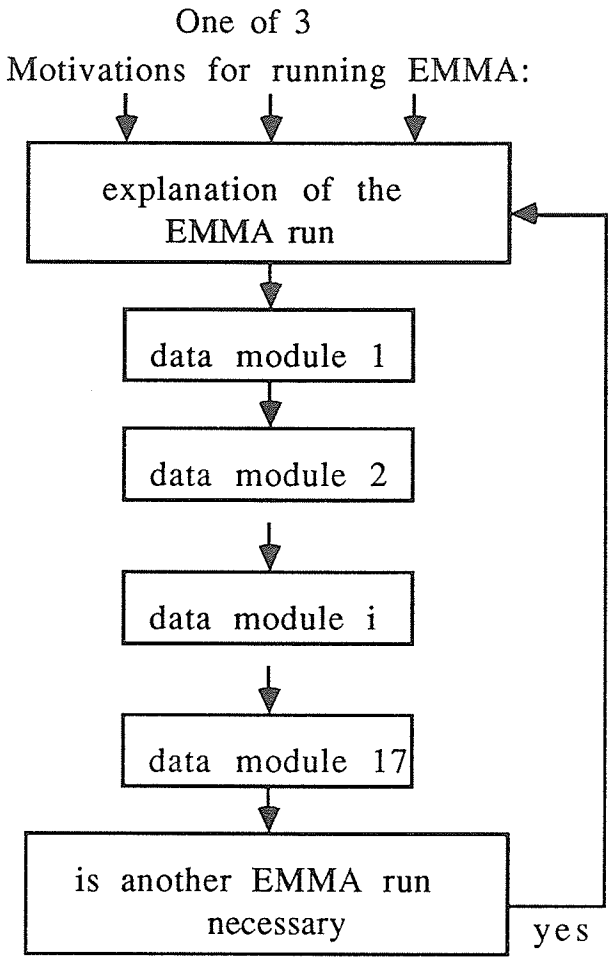


Figure 4. Flow chart of an expert system consultation

The explanations provided by the expert system may be classified in four groups:

1) General manual-like explanations of items necessary in an input data set. All of these explanations list the number and the name of the data module they explain and all data necessary for that data module. For example, the beginning of

the explanation of the data module 4 that describes the size of the hydraulic and electrical networks begins like this:

> "DATA MODULE 4 - NETWORK AND CONSTRAINT SIZE:
> This module defines the size of the physical problem and the number of grouping constraints. If the size of the problem becomes a problem to EMMA, report #2 (parameter data) gives the size limits that can be accepted by the LP algorithm. The data are required in the following order (explanations are given in brackets):
> 1. Number of time steps - T (This item can be obtained from MODULE 5. It is convenient to complete MODULE 5 before completing MODULE 4. This item is very important for data entry in other data modules, and will be referred to as T, later in this consultation)
> 2. ..."

Most of these explanations were found in the EMMA manual. A few were obtained through interviews with the expert and careful analysis of one input data set for the program.

2) Advice for a particular EMMA run and motivation. The following example gives advice on the third entry necessary in the data module 5 that defines the time structure of the problem. It is related to a severe case EMMA run within the budgetary or operational plan context:

> "DATA MODULE 5 - TIME STRUCTURE:
> ADVICE:
> Item 3 - Assume a severe winter with an early freeze-up date, the ICE-COVER time steps start earlier, and last longer during EMMA time horizon!
> ..."

3) Hints about specific values that can be used in input. These are the values that are used by the expert and that usually give good results. For example, the expert system recommends the values that can be used in the data module 3 that controls the operation of the linear programming algorithm:

"DATA MODULE 3 - LINEAR PROGRAMMING PARAMETERS :

These data control the operation and solution of the linear programming algorithm. The following data are required (recommended values are given in brackets):

1. Absolute tolerance for LP zero tests (1E-9)..."

4) Explanations of outside programs and procedures used by the expert for preparation of certain parts of input:

"CONSTRUCTION OF THE LOAD DURATION CURVE - ADVICE:

It is possible, based on 52 occurrences in the past year, to find the relationship between temperature and demand for energy, for each hour of the week.

Using that relationship, it is possible to calculate energy consumptions that correspond to different historic sequences of temperatures.

This enables calculation the mean and standard deviation for energy consumption, as well as energy demands with different probabilities for each EMMA time step.

These energy demands can be adjusted to account for the growth of consumption between the last year and the year EMMA is run for."

All of these explanations were compiled and entered into the computer code. The screen output of the expert system is given in Appendix A.

The input that the expert system needs from the user consists of: (i) selection of the motivation for running EMMA, (ii) choice to suppress the appearance of data modules that do not change between subsequent EMMA runs for the selected motivation, and (iii) choice to select independently data modules to be explained.

The main part of the expert system consists of a chain of eighteen modules. The first module describes the particular EMMA run, and the other seventeen give explanations and advice on data assembly and use. Each of these modules sends to the screen sequences of textual information that depend only on two parameters: (i) the motivation for the use of EMMA selected by the user, and (ii) the current EMMA run for that motivation, which is recorded by the expert system during a

consultation. This simple structure reflects in the developed knowledge base that consists of a couple of rules for each data module. Obviously, the nature of the acquired knowledge did not require the use of semantic nets or frames and very sophisticated techniques for search through the knowledge base.

5.1.3 Use of the Selected Tool

The expert system component of the IDS system was developed using RuleMaster 2 as a rule-based expert system development tool. Towards the end of the expert system development some malfunctions in the tool considerably slowed down the work. RuleMaster 3, a purified version of RuleMaster 2, was released too late to be applied and tested in this project.

The basic unit of an expert system developed using RuleMaster 2 is an induction module. An induction module can call other subordinate modules and pass and receive values from them. Each module consists of one or more induction states. There are three distinct parts in each induction state, and they are: actions block, conditions block, and examples block.

The **actions block** lists all possible things that the expert system may do within the induction state. These include calls to other modules, calculations of numerical or logical values, or advice for the user.

The **conditions block** lists all relevant conditions that will influence the choice of actions to be taken within the induction state. A condition can be any expression that reduces to a single unique value, such as: the user's answer to a question, variable calculation or a value, or a call to a subordinate module that returns a single value.

The **examples block** states the actions to be taken and the next induction state within the induction module to be executed for all possible combinations of conditions.

Backwards chaining is accomplished by calling lower level modules through the conditions part of an induction state, while forwards chaining takes place when a lower level module is invoked from the actions block. RuleMaster 2 deduces rules and performs helpful intelligent logical checks for entered induction states and modules. It also generates C code, which can be compiled in order to speed up expert system consultations. A sample code of an induction module written in RuleMaster's Radial language is given in Appendix B.

The above explained structure of RuleMaster 2 was found to be well suited for this expert system because it does not have to perform major inferencing or backwards chaining in order to give an advice to the user and because it consists of a number of different modules.

The main characteristic of the developed expert system is that it has to transfer a lot of expertise and information to the user. The major role of a good development tool in this situation was facilitating the input and organization of all this knowledge and providing an efficient and appealing transfer of information to the user by the use of colors, graphs and a good screen control.

The selected tool, RuleMaster 2, successfully fulfilled only the first of these two tasks. RuleMaster's modular structure greatly facilitated organization of knowledge and gradual appending of data modules to the initial version of the expert system prototype. The tool also provided a friendly editor for fast generation of code, as well as intelligent checks of its logic.

The major shortcoming of the tool was its poor graphical and screen control capability. Breaking long sequences of text with graphs, and refreshing the appearance of the expert system with colors would have made the transfer of information and knowledge to the user much more efficient and less tiring.

5.2 Interactive Computer Graphics Component

In transferring engineering knowledge and communicating information about a system like an interconnected power utility that consists of diverse elements, it is often necessary to draw graphs and schemes. The dull appearance and poor screen control of RuleMaster 2 forced the developers of the expert system to explore the possibility of using an external graphics library in order to enhance the communication between the user and the expert system. The structure of RuleMaster 2 and its capability to generate C code functions were found to be well suited for adding external graphics C routines.

In the Manitoba Hydro case study, the computer graphics part of the IDS system was developed for enhancing the use of EMMA model in planning the operations of Manitoba Hydro system and to assist in the input data preparation for the model. Both of these tasks can be very discouraging and frustrating for an inexperienced user, especially in cases where numerical approximations, linearization, and segmentation of data are required in formulating the problem for EMMA.

Introduction of graphics provided additional benefits in quick checks of input data such as load duration curves, rating curves, inflows, etc. This, combined with interactive, mouse-driven features that enable picking lakes, hydro and thermal power plants, and other elements from the scheme that represents the entire system, substantially eases the burden of data management for the user. It also speeds up the work both with the EMMA model and the expert system, thus enabling the user to concentrate on more important issues and the system's performance. The graphics supplement of the expert system consists of two distinct parts.

The first part enables interactive editing of the system scheme which consists of lakes, inflows into the system, controlled lake outlets, hydro generating stations,

natural lake outlets, and thermal generating stations. The flow chart of this part of the developed graphics package is shown in Figure 5.

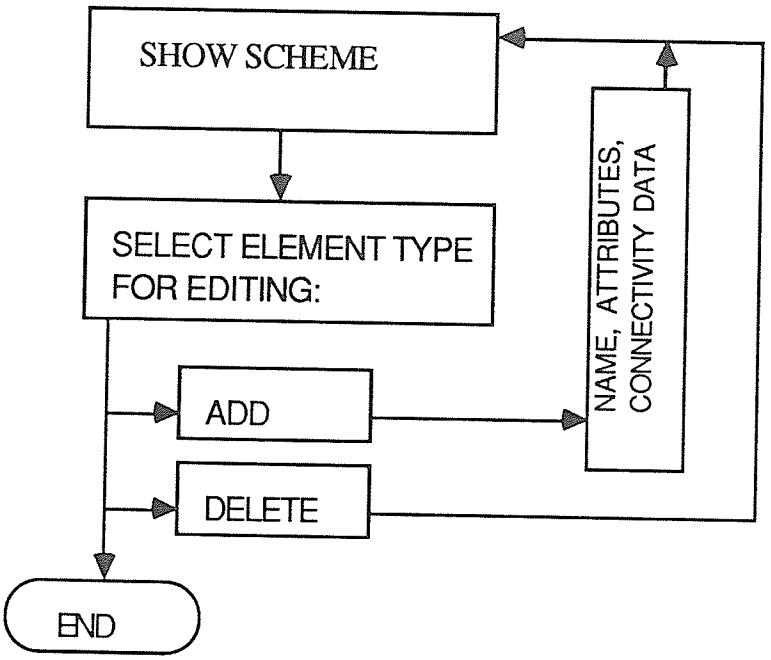


Figure 5. Editing elements in the scheme of the system

Lakes are represented in the scheme by circles whose colors mark the purpose of the lake. Inflows are represented by blue triangles, and natural outlets by blue rectangles between the lakes they connect. Hydro and thermal generating stations, and control structures are represented by green, blue, and white rectangles respectively. The size of each element can be adjusted interactively to reflect its relative importance or size. The program incorporates the logic of interconnecting these elements that is used by EMMA. In this way, by building the scheme of the utility, the user can easily conceptualize the approach used by EMMA in modelling the system. Later, in the use of both EMMA and the expert system, the scheme can be used for overview of the system performance and quick

access to parts of EMMA input and output, as well as to illustrate the advice provided by the expert system.

An example of a system scheme is shown in Figure 6. The main menu of the part of the developed graphics package that enables interactive editing of the scheme can be seen in the left part of the figure. If the user selects to add an element to the scheme, the menu shown in Figure 7 appears on the screen. The user can then select to add one of the possible six kinds of elements. When the user selects the option to add a lake, as shown in Figure 7, he or she is prompted to enter the lake name as shown in Figure 8. A lake can be unregulated or regulated. Lakes can be regulated for hydro power generation or for some other purpose. The selection of one of these three options is shown in Figure 9. The selection of the position of the new lake is shown in Figure 10. Finally, the user can adjust the size of the new lake as shown in Figures 11 and 12.

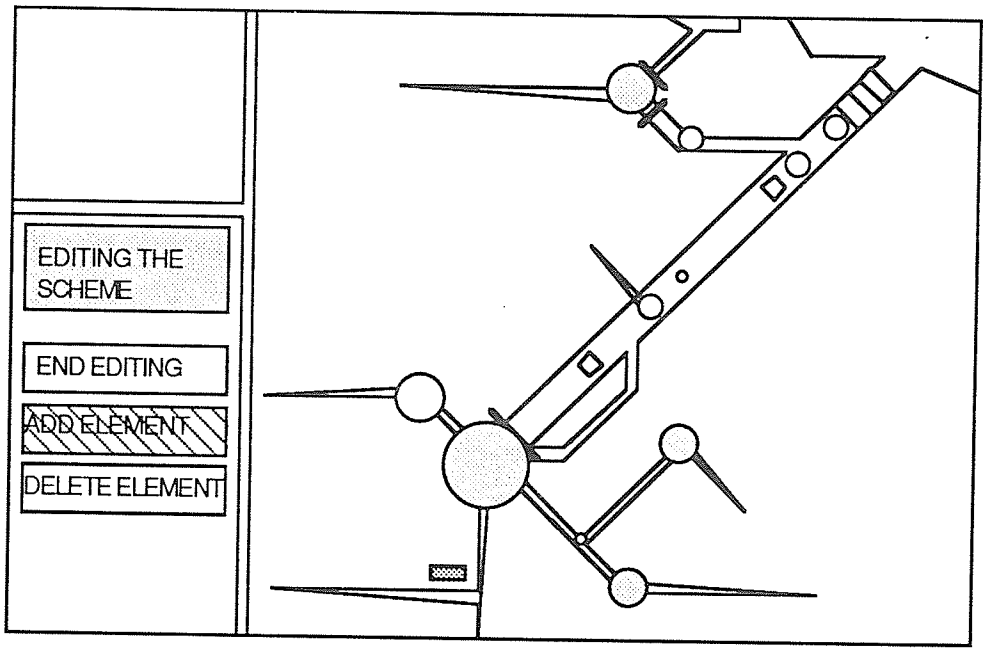


Figure 6: Adding an element to the scheme of the system

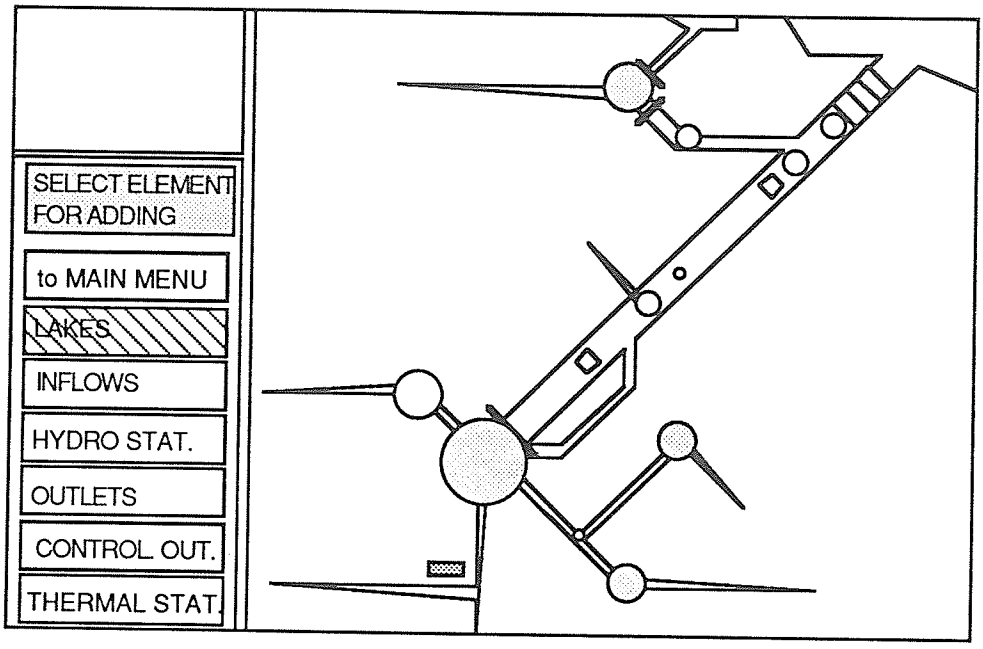


Figure 7: Selecting to add a lake to the scheme of the system

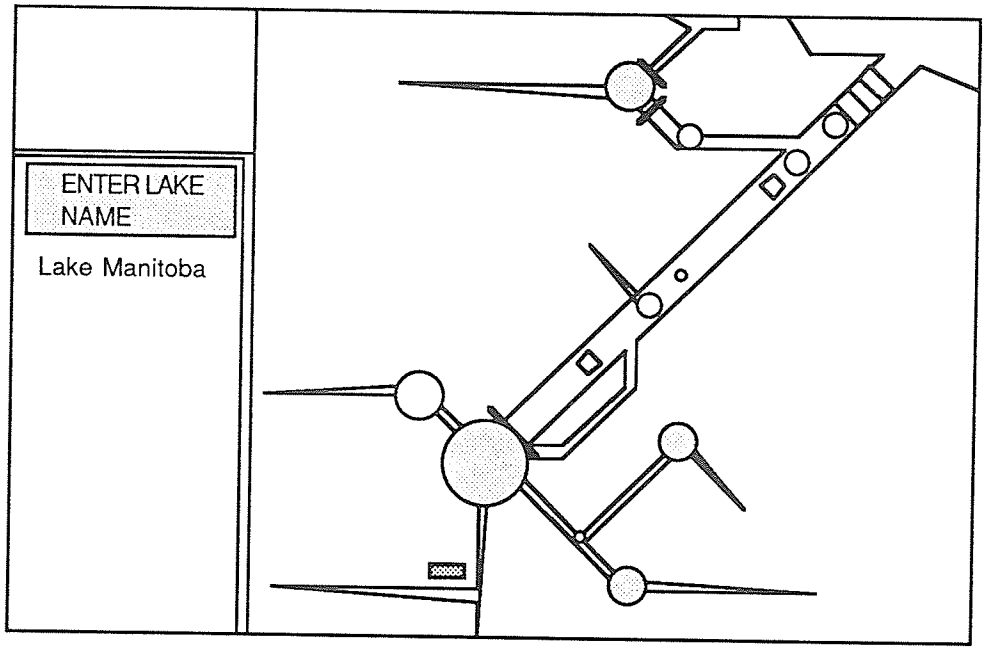


Figure 8: Entering the name of the new lake in the system.

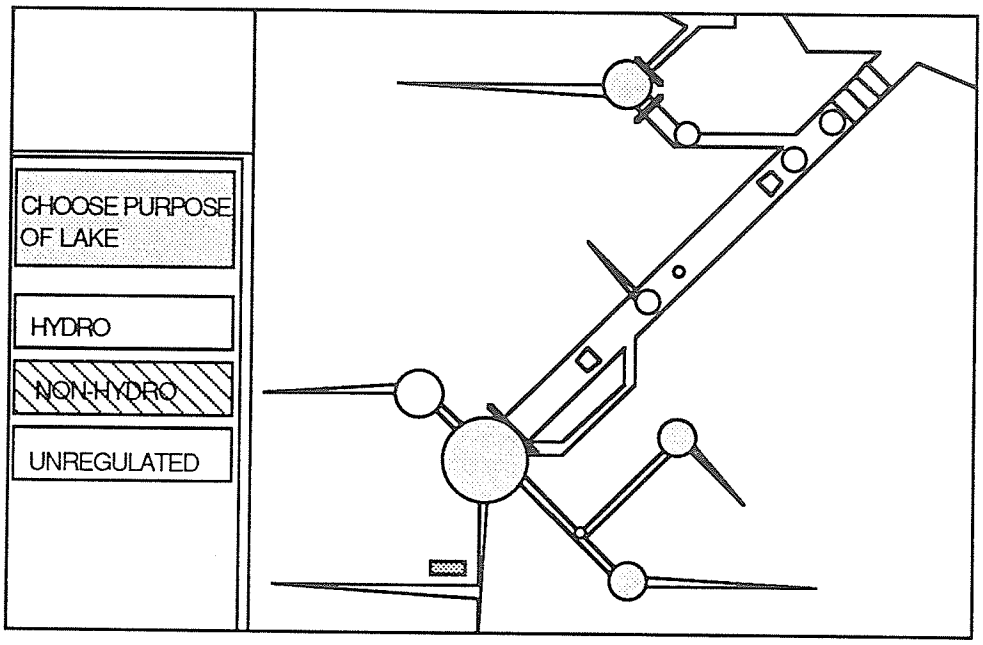


Figure 9: Specifying the purpose of the new lake in the system

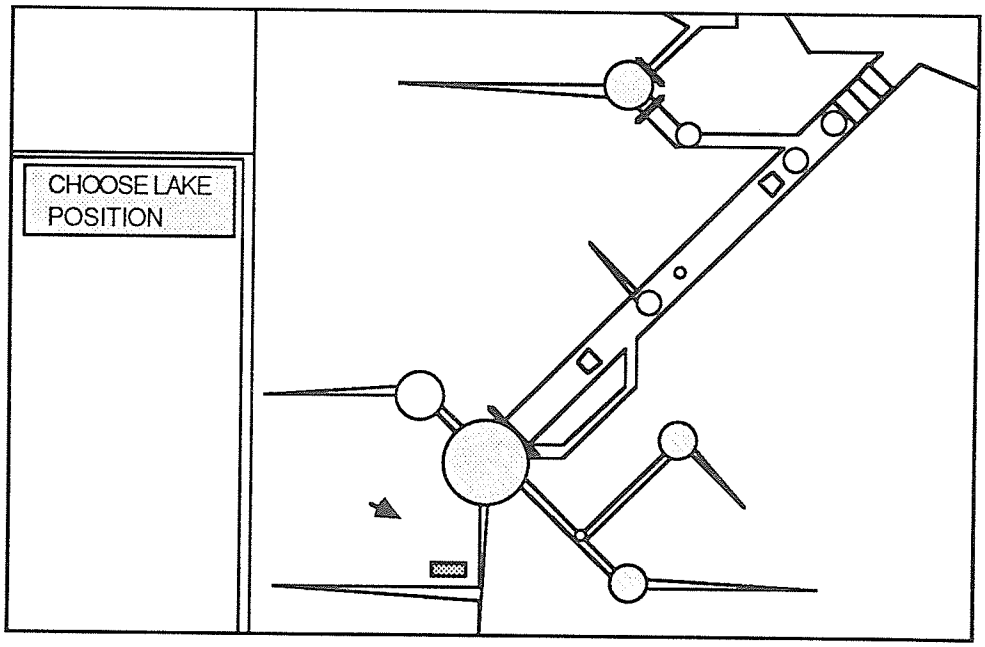


Figure 10: Selecting the position for the new lake in the system

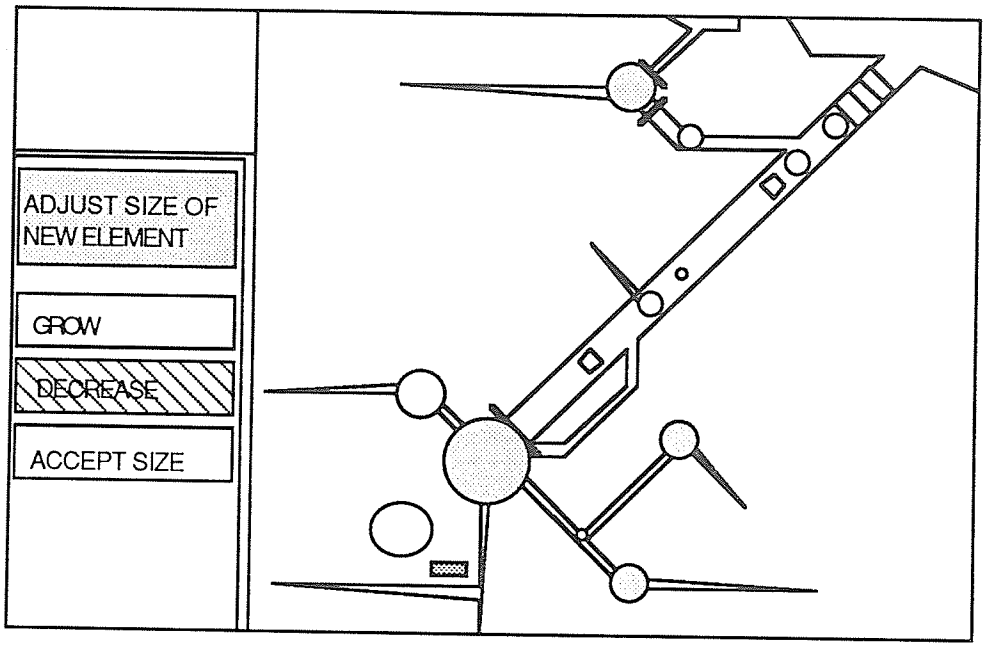


Figure 11: Adjusting the size of the new lake in the system

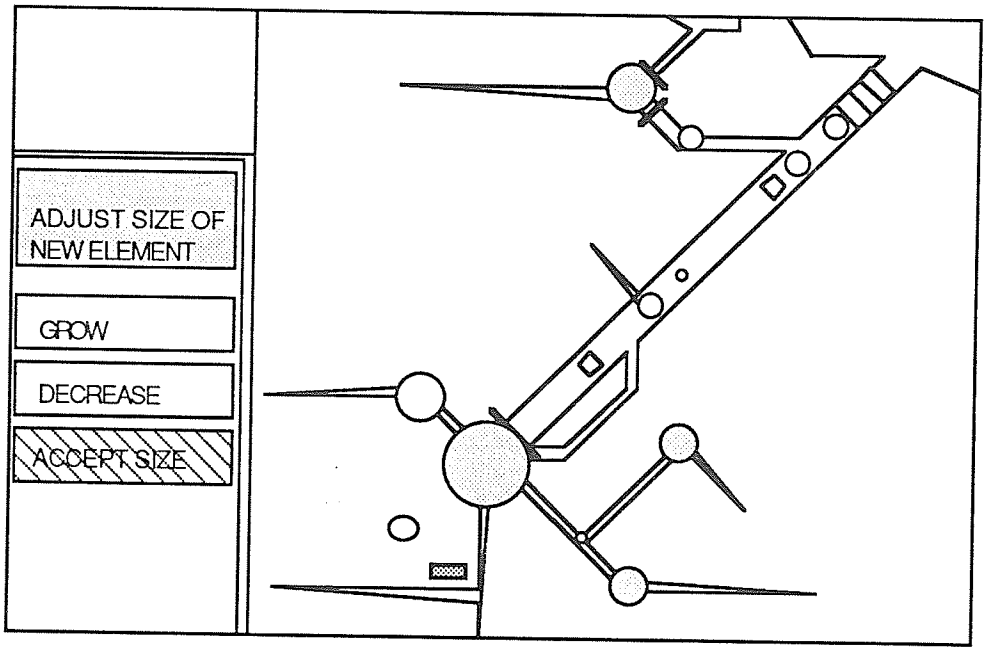


Figure 12: Accepting the size of the new lake in the system

Hydro and thermal stations, natural and controlled outlets, and inflows can be added to the scheme in a similar manner. All these elements can be deleted by selecting the appropriate option from the menu shown in Figure 6 and by picking the element to be deleted by the use of a mouse.

The second part of the graphics supplement to the expert system provides an interactive check of data related to the elements picked from the main scheme of the utility. The flow chart is shown in Figure 13.

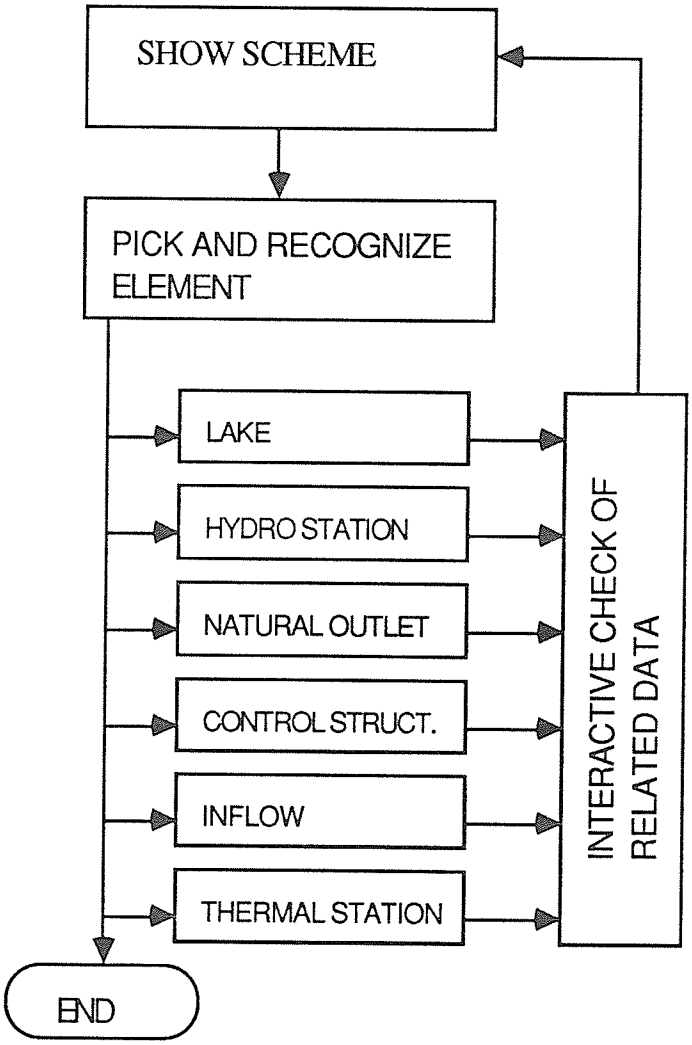


Figure 13. Showing data related to the elements of the system

This part is envisioned to eventually incorporate much engineering expertise related to proper data assembly and use acquired through knowledge acquisition. The images that show different graphs with appropriate comments appear on the screen as the expert system guides the user through data explanatory modules. Currently, the system displays the following graphs:

1) for lakes and reservoirs: stage-storage curves, historic levels, and value of water in storage;

2) for inflows: historic flows, and flow scenarios prepared for EMMA;

3) for control structures: historic releases;

4) for natural outlets: stage-discharge relationships, and historic flows; and

5) for hydro stations: efficiency curves, and open water and ice cover tailrace curves.

The routines developed to show data in graphical form can be easily applied to display other data if the need arises.

A system scheme and the example of picking a lake is shown in Figure 14. The selection of the rating curve that corresponds to the picked lake is shown in Figure 15. Larger graphs, such as historic levels or discharges, that cannot fit into the screen can be scrolled left and right. An example showing historic levels for the picked lake is given in Figure 16.

2D Graphics Metafile Resource (2D GMR) library of C routines that comes with the Apollo Domain 4000 workstations was used in developing this interactive supplement of the expert system. This software package enables storage of graphic data in the form of disk metafiles, good screen control, and quick display of desired parts of the metafile. Graphic data in a metafile are grouped in segments and this feature was used to group the elements of the utility in the LAKES, INFLOWS, HYDRO_STATIONS, CONTROLLED_OUTLETS, NATURAL_OUTLETS, and THERMAL_STATIONS segments.

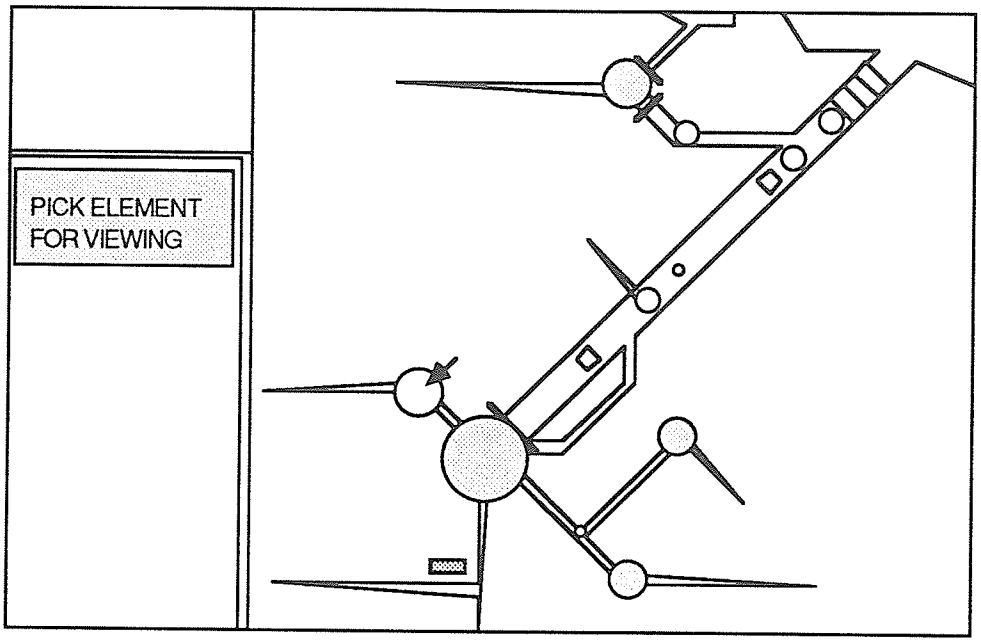


Figure 14: Picking an element for interactive check of related data

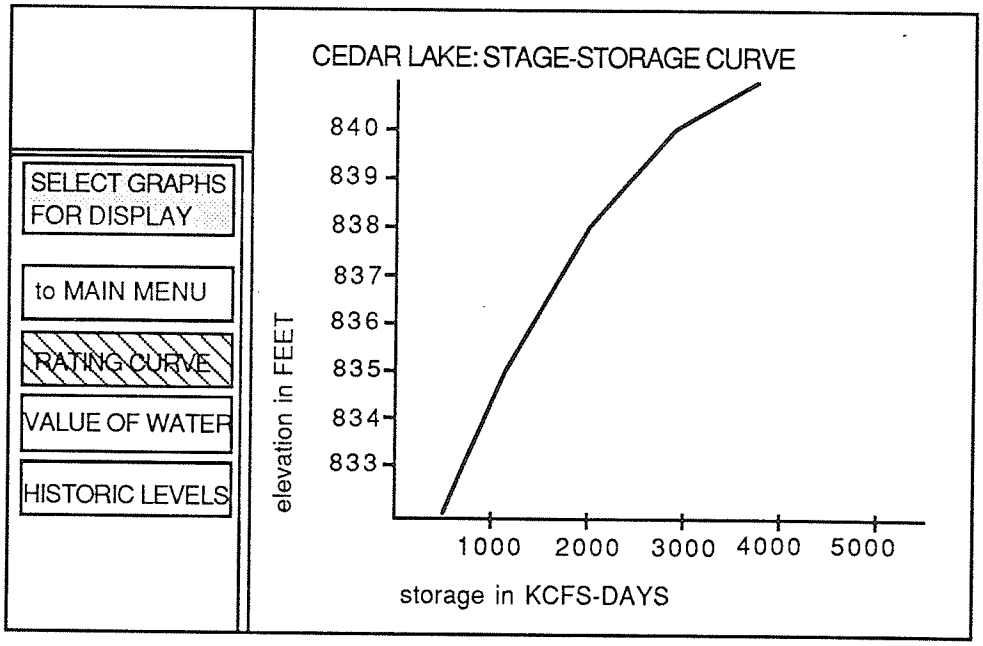


Figure 15: Rating curve for the picked lake

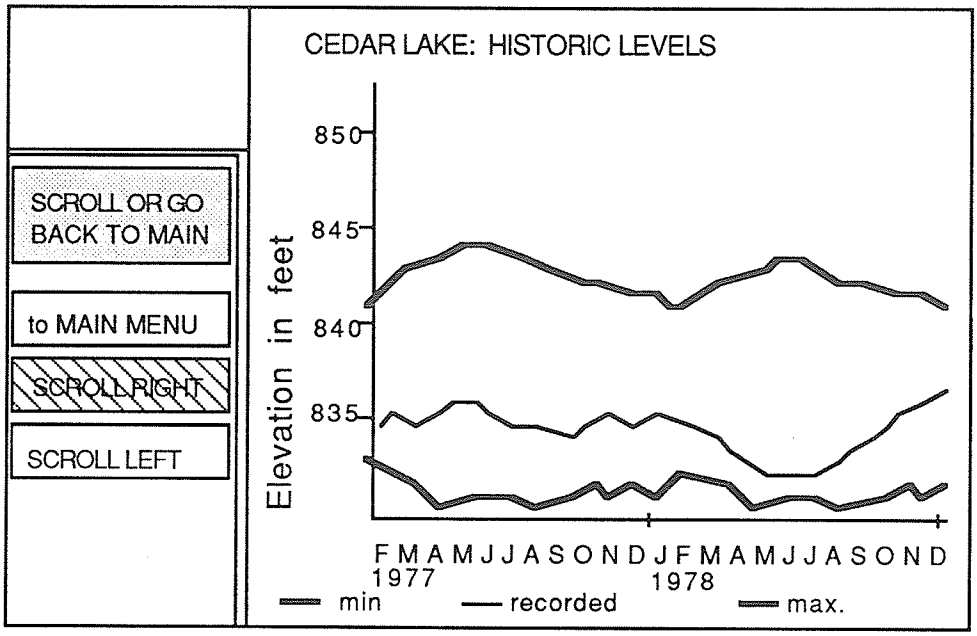


Figure 16: Historic levels for the picked lake

The graphics library also enables storing textual attributes, such as names or comments for entered elements. This enabled storage of connectivity data, for example, the name of the downstream and the upstream lake for each natural outlet, hydro generating station, or control structure. The search and pick routines provided by the library enable identifying an element picked by the user from the scheme using a mouse, as well as searching for elements by their names from metafile segments. These features can be further exploited in generating appropriate and more specific advice by the expert system component of the IDS system. This also points to the possibility of expanding both the expert system and computer graphics towards a sophisticated and efficient database management.

Powerful search capabilities of 2D GMR package combined with flexibility of the C language for which they are written also indicate the possibility of providing help in actual data assembly for EMMA runs by the expert system. The

entire C code of the graphics application developed during this research can be found in Appendix C.

5.3 Integration of the Decision Support System

The final appearance of the developed IDS system indicates that, in a sense, the poor graphics capability of RuleMaster 2 in comparison with other expert system development tools was an asset in this particular case. Early in the development of the application, it became obvious that it was necessary to use an external, completely graphics oriented library. This enabled a level of sophistication and flexibility that could not have been achieved with other tools.

In the final integration of the IDS system, it became apparent that the RuleMaster 2 shell as well as the tool's screen control interfered with the developed graphics component. Some malfunctions were noticed in the shell's part that is supposed to provide interaction of the expert system with external programs as well as in handling arrays and computation of numerical values. After a significant slowdown in the development of the IDS system that was caused by these problems, the developers decided to abandon the use of RuleMaster 2, and to proceed with the use of C language and the C code routines generated by the tool until then.

The overall experience with RuleMaster 2 shows that the tool enabled rapid prototyping, which proved beneficial in the knowledge acquisition phase and enabled an early start of verification of the expert system. On the other hand, when the project reached the level of sophistication when interaction with other software and manipulation with disk files were necessary, the use of RuleMaster 2 became a limiting factor.

At this point, it seems that only programming languages, such as C, can provide enough flexibility and respond to all project-specific tasks that may be required in the development of an IDS system. Using programming languages in

this manner requires substantial programming skills, which means that at least one of IDS system developers should have background in computer science.

CHAPTER 6.
CONCLUSIONS

This thesis has focused on the application of the IDS concept to the case study of reservoir and energy scheduling at Manitoba Hydro. The concept introduced by Simonovic and Savic [1989] and ideas based on the previous phase of the project and obtained from the available literature on expert system were used in the development of the IDS system.

Four basic tasks had to be dealt with during this research: (i) knowledge acquisition, (ii) development of an expert system, (iii) development of an interactive computer graphics supplement to the expert system, and (iv) integration of an IDS system. Although the developed computer application is very far from entering everyday use in an actual working environment, it clearly points to possible benefits from the use of such an IDS system and directions for further development. It also demonstrates the benefits of expert system technology when combined with computer graphics, engineering expertise, and the systems approach in forming a strong basis for the decision making process.

This research focused more on exploring what could be done using available technology than on simulating the expert's reasoning. New insights into the problem can be expected from the next phase in which the developed IDS system is being tested and verified with the assistance of an inexperienced user at Manitoba Hydro.

6.1 Lessons Learned

Development of an expert system is, to some extent, a journey into the unknown. Neither the experts nor the knowledge engineers can have a completely clear vision of the final shape of the expert system at the very beginning of the

project. It is through the time consuming, sometimes frustrating and discouraging, process of knowledge acquisition and gradual growth of the prototype expert system that both sizes slowly realize where real problems lie and where and what kind of expertise comes into play. Ending up with something slightly different from what was originally envisioned means that some misconceptions either about expert systems or about the problem itself have been rectified.

This particular research provided additional evidence for some observations made by numerous authors who examine expert systems. It showed that knowledge acquisition, and expert systems in general, should focus on actual useful jobs and tasks performed by experts. Formulation of rules applicable to a particular problem is well suited for the currently available knowledge acquisition techniques. It also leads to knowledge representation that is acceptable to widely available computer hardware and software. On the other hand, trying to capture and formulate general knowledge and expertise will likely result in a failure. In the development of an expert system, the importance of continual interaction between the expert, knowledge engineer, and the final user cannot be overstressed. Their joint effort and communication, interest, motivation, and persistence are the most important factor in the development of an effective expert system.

Most reported successful applications of expert system technology are diagnostic in nature, and appeared in fields such as medicine, geology, and chemistry. These applications usually have to take into account a large number of factors, search through a knowledge base and provide an answer or advice. The available expert system development tools ease the task of organizing and encoding a large number of facts and rules in a knowledge base. They also assist in constructing an efficient search mechanism. This indicates that these tools may be appropriate for similar diagnostic applications. On the other hand, the tools are likely to perform worse than classical programming languages in applications

where the need to interact with databases and other software is stressed over the need to organize a knowledge base and a search strategy. In the selected case study, the developers had to abandon an expert system development tool and use the C programming language. This indicates that the use of classical programming languages from the beginning of an expert system development may be appropriate for the WRE problems that require the use of quantitative models and massive data.

The need to make an expert system interact with other tools, or even become incorporated within an IDS system will likely prevent the expert system from following rigidly the initially envisioned structure that separates the knowledge base and the inference engine. This will put the expert system even further from the final goal of easy updates of the knowledge base and eventually automated machine learning. The maintenance and updates of such an expert system will be complicated and will require permanent involvement of people with necessary computer science and knowledge engineering skills. On the other hand, integration of an expert system with quantitative models, data bases, and other tools within an IDS system will increase the utility of the developed computer application. The presented IDS concept is an example of such a pragmatic approach in water resources engineering.

6.2 Future Development

Although it can provide basic explanations and guidelines for the use of EMMA, the developed IDS system will have to further integrate other tools and expertise related to their use by the RES engineer in order to become really useful. Some of the directions for further development are the flow and load forecasting models used by the RES engineer at Manitoba Hydro. The two expert systems developed during the previous phase of the project, that deal with intelligent data

pre- and post- processing could also become useful parts of the developed IDS system.

The most interesting direction for further development is integration with the very sophisticated knowledge base that is already in use at Manitoba Hydro. In fact, management of data is of such importance for numerous tasks in water resources engineering that it could become the fifth basic element in the IDS concept. This is particularly applicable to data-intensive tasks similar to the use of EMMA.

The expertise that is in the focus of this research deals with the use of LP and other procedures in optimization of operations of an interconnected hydro-thermal power utility. Some of this expertise was formulated for the first time during this research. The formulated rules may be a starting point for improvements of the procedures and methods that are currently in use. Some of the formulated rules and procedures are case-specific. Their generalization is an interesting task for possible further research.

6.3 Benefits from Expert System Development

Systematization and codification of knowledge will often provide the expert with new insights into his or her own expertise and serve the expert as a useful help later in his or her work. Formulating scarce expertise and knowledge , and exposing it to a wider group of users or other experts, may be a starting point for an improvement of problem solving methods, or for the trace back to the basic scientific principles and better understanding of heuristic procedures. Hayes-Roth [1984] observed that the development of expert systems is an answer to knowledge creation that has outpaced its use and dissemination. An interesting phenomenon in the future may be a further speed up in knowledge creation caused by the development of expert systems.

For an expert, an IDS system that successfully integrates different tools can significantly ease the burden of dealing with massive data and going through monotonous tasks. To a novice engineer, in addition to this, it can provide guidance in gaining necessary expertise. The organization where this process takes place will benefit from the captured knowledge and its availability to more of its members in their everyday practice.

REFERENCES

1. Alim, S., "A PROLOG-based Expert System for Encoding Seismic Design Provisions", Civil Engineering Systems, 4(1), 1987, p.39-44.
2. Allen, R.H., M.G. Boarnet, C.J. Culbert, and R.T. Saveley, "Using Hybrid Expert System Approaches for Engineering Applications", Engineering with Computers, 2(2), 1987, p.95-110.
3. Baffaut, C. and J.W. Delleur, "Expert Systems for Calibrating SWMM", Journal of Water Resources Planning and Management, 115(3), 1989, p.278-298.
4. Barlishen, K., "Knowledge Acquisition in Water Resources Engineering", M.Sc. thesis, University of Manitoba, 1989.
5. Barlishen, K.D., S.P. Simonovic, and D.H. Burn, "Knowledge acquisition within the expert system approach for water resources engineering applications", ASCE Journal of Water Resources Planning and Management, submitted for publication, 1990.
6. Barritt-Flatt, P.E., and A.D. Cormie, "A Comprehensive Optimization Model for Hydro-Electric Reservoir Operations", Computerized Decision Support Systems for Water Managers, edt. by Labadie, J.W., L.E. Brazil, I. Corbu, and L.E. Johnson, ASCE, p. 463-477, 1989.
7. Blockley, D.I., "Uncertainty Analysis in Expert Systems", Civil Engineering Systems, 4(1), p.3-6, 1987.
8. Bobrow, D.G., S. Mittal and M.J. Stefik, "Expert Systems: Perils and Promise", Communications of the ACM, 29(9), 1986, p.880-894.
9. Bramer, M.A., "Expert Systems: The Vision and the Reality", in Research and Development in Expert Systems, M.A. Bramer (ed.), Cambridge University Press, New York, 1985, p.1-12.

10. Burgoyne, C.J. and S.H.R. Sham, "Application of Expert Systems to Prestressed Concrete Bridge Design", Civil Engineering Systems, 4(1), 1987, p.14-19.

11. Cohn, L.F., R.A. Harris, and W. Bowlby, "Knowledge Acquisition for Domain Experts", Journal of Computing in Civil Engineering, 2(2), 1988, p.107-120.

12. Coombs, M. and J. Alty, "Expert Systems: An Alternative Paradigm", in Developments in Expert Systems, M.J. Coombs (ed.), Academic Press, Toronto, 1984, p.135-157.

13. Cormie, A.D. and P.E. Barritt-Flatt, "Hermes: A decision support system for reservoir operations at Manitoba Hydro, paper presented at the Operations Planning Section, Canadian Electrical Association, Vancouver, March, 1987.

14. Cormie, A.D., P.E. Barritt-Flatt, R.W. Herzog, L. Kessler, and R. Welwood, EMMA: Energy Management and Maintenance Analysis, Manitoba Hydro, 1985.

15. Cupello, J.M. and D.J. Mishelevich, "Managing Prototype Knowledge/Expert System Projects", Communications of the ACM, 31(5), 1988, p.534-550.

16. Davis, M.W., "Anatomy of Decision Support", Datamation, June 15, 1984, p. 201-208.

16. Dreyfus, H., "Why Computers May Never Think Like People", Technology Review, 89(1), 1986, p.42-61.

17. Duda, R.O. and E.H. Shortliffe, "Expert Systems Research", Science, 220(4594), 1983, p.261-268.

18. Dym, C.L., "Expert Systems: New Approaches to Computer-aided Engineering", Engineering with Computers, 1(1), 1985, p.9-25.

19. Fayegh, D. and S.O. Russel, "An Expert System for Flood Estimation", in Expert Systems in Civil Engineering, ASCE Publication, New York, 1986, p.174-181.

20. Fedra, K., E. Weigkricht, and L. Wilkenbauer, "A Hybrid Approach to Information and Decision Support Systems: Hazardous Substances and Industrial Risk Management", IFAC, Economics and artificial intelligence, 1986, p. 169-175.

21. Fedra, K., "A Modular Interactive Simulation System for Eutrophication and Regional Development", Water Resources Research, 21(2), 1985, p. 143-152.

22. Fedra K. and D.P.Loucks, "Interactive Computer Technology for Planning and Policy Modeling", Water Resources Research, 21(2), 1985, p. 114-122.

23. Fenves, S.J., M.L. Maher, and D. Sriram , "Expert Systems: C.E. Potential", Civil Engineering (ASCE), 54(10), 1984, p.44-47.

24. Franck, B.M. and T. Krauthammer, "An Expert System for Field Inspection of Concrete Dams: Part 1, Engineering Knowledge", Engineering with Computers, 5(1), 1989, p.23-38.

25. Franck, B.M. and T. Krauthammer, "An Expert System for Field Inspection of Concrete Dams: Part 2, Artificial Intelligence Issues", Engineering with Computers, 5(2), 1989, p.119-131.

26. Grahovac, J., and S.P. Simonovic, "Expert System for Budget Preparation and Optimal Operation of an Interconnected Power Utility", ASCE, Optimizing the Resources for Water Management, edt. R.M. Khanbilvardi and T.C. Gooch, pp. 34-40, 1990.

27. Guariso, G., and H. Werthner, Environmental Decision Support Systems, Ellis Horwood Limited, Chichester, England, 1989.

28. Haley, P. and C. Williams, "Expert System Development Requires Knowledge Engineering", Computer Design, 25(2), 1986, p.83-88.

29. Hayes-Roth, F., "The Knowledge-Based Expert System: A Tutorial", Computer, September 1984, p.11-28.

30. Hayes-Roth, F., "Knowledge-Based Expert Systems", Computer, 17(10), 1984, p.263-273.

31. Herzog R.W. and P.E. Barritt-Flatt, Report on HERMES Project User Requirements for Hardware and Software Environment Specification, Manitoba Hydro, 1985.

32. Hogley, J.R. and A.R. Korncoff, "Artificial Intelligence in Engineering: A Revolutionary Change", in Applications of Artificial Intelligence in Engineering Problems, Volume II, D. Sriram and R. Adley (eds.), Computational Mechanics Publications, New York, 1986, p.1155-1160.

33. Jenkins, W.O. and P.W. Jowitt, "Expert Systems in River Basin Management", Civil Engineering Systems, 4(1), 1987, p.31-38.

34. Kahn, G., S. Nowlan, and J. McDermott, "Strategies for Knowledge Acquisition", IEEE Transactions of Pattern Analysis and Machine Intelligence, PAMI-7(5), 1985, p.511-522.

35. Kindler, J., "Water Resources Planning and decision Making: Some Thoughts about the Future", Proceedings of the International Conference on Decision Support Systems in Water Resources, Oslo, Norway, April 1986.

36. Kirby, H.R. and F.O. Montgomery, "Towards a Rule-Based Approach for Traffic Signal Design", Civil Engineering Systems, 4(1), 1987, p.20-26.

37. Knight, B. "An Expert System for Upgrading Small Water Supplies", Civil Engineering Systems, 4(1), 1987, p.27-30.

38. Kolonder, J., "Towards an Understanding of the Role of Experience in the Evolution from Novice to Expert", in Developments in Expert Systems, M.J. Coombs (ed.), Academic Press, Toronto, 1984, p.95-116.

39. Kostem, C.N. and M.L. Maher (eds.), Expert Systems in Civil Engineering, ASCE Publication, New York, 1986.

40. Lenat, D.B., "The Nature of Heuristics", Artificial Intelligence, 19(2), 1982, p.189-249.

41. Loucks, D.P., J. Kindler, and K. Fedra, "Interactive Water Resources Modelling and Model Use; An Overview", Water Resources Research, 21(2), pp. 95-104, 1985.

42. Loucks, D.P., M.R. Taylor, and P.N. French, "Interactive Data Management for Resource Planning and Analysis", Water Resources Research, 21(2), 1985, p. 131-142.

43. Lu, S.C-Y., "Knowledge Map: An Approach to Knowledge Acquisition in Developing Engineering Expert Systems", Engineering with Computers, 3(2), 1987, p.59-68.

44. Maher, M.L., Expert Systems for Civil Engineers: Technology and Application, ASCE Publication, New York, 1987.

45. Nagy, A., K.D. Barlishen, D.H. Burn, and S.P. Simonovic, "Toward an Expert System for Improving the Operations Planning in Manitoba Hydro", ASCE, Water Resources Planning and Management, ed. S.C. Harris, pp. 477-481, 1989.

46. Nau, D.S., "Expert Computer Systems", Computer, February 1983, p.63-84.

47. Palmer, R.N. and K.J. Holmes, "Operational Guidance During Droughts: Expert System Approach", Journal of Water Resources Planning and Management, 114(6), 1988, p.647-666.

48. Palmer, R.N. and B.W. Mar, "Expert Systems Software for Civil Engineering Applications", Civil Engineering Systems, 5(4), 1988, p.170-180.

49. Palmer, R.N. and R.M Tull, "Expert Systems for Drought Management Planning", Journal of Computing in Civil Engineering, 1(4), 1987, p.284-297.

50. Radian Corporation, RuleMaster 2: A software Tool for Building Expert Systems, Reference Manual, 1987.

51. Reznicek, K., "An Application of Successive Linear Programming to the Optimization of an Interconnected Hydro Utility Operation", M.Sc. thesis, University of Manitoba, 1988.

52. Reznicek, K.K. and S.P. Simonovic, "An Improved Algorithm for Hydropower Optimization", Water Resources Research, 26 (2), pp. 189-198, 1990.

53. Reznicek, K.K., D.A. Cormie, P.E. Barritt-Flatt, and S.P. Simonovic, "Hydropower Optimization in Practice and Research, in Proceedings of 4th Canadian Seminar on Systems Theory for Civil Engineer", The University of Manitoba, Department of Civil Engineering, pp. 64-74, 1989.

54. Simonovic, S.P. and D.A. Savic, "Intelligent Decision Support and Reservoir Management and Operations", ASCE, Journal of Computing in Civil Engineering, 3(4), pp. 367-385, 1989.

55. Simonovic, S.P. and J. Grahovac, "HYDRO: An Application of Interactive Computer Graphics in Water Resources", Microcomputers in Civil Engineering, submitted for publication, 1990.

56. Starfield, A.M., K.L. Butala, M.M. England, and K.A. Smith, "Mastering Engineering Concepts by Building an Expert System", Engineering Education, 74(2), 1983, p.104-107.

57. Taylor, M.R. and P.N. French, "Interactive Data Management for Resource Planning and Analysis", Water Resources Research, 21(2), 1985, p. 131-142.

57. Thierauf, R.J., User-Oriented Decision Support Systems, Prentice Hall, Englewood Cliffs, New Jersey, 1988.

58. Virard, M.A. "Transfer of Engineering Knowledge", Engineering Digest, 32(8), 1986, p.32-33.

59. Yeh, W.W-G., "Reservoir Management and Operations Models: A State of-the-Art-Review", Water Resources Research, 21(12), 1985, p.1797-1818.

APPENDIX A

SCREEN OUTPUT OF THE EXPERT SYSTEM

* *

EMMA

The EMMA (Energy Management and Maintenance Analysis) program has been developed by the Informations Systems Division and Energy Resources Section of Manitoba Hydro for use as an operational planning tool for an interconnected hydro thermal utility.

* *

* *

EMMA formulates a plan for energy production in hydro and thermal generating stations, release policy, exports and imports of energy, and maintenance for specified time horizon and flow scenario, with the objective of meeting the system load while maximizing revenues and minimizing costs.

This expert system, developed at the University of Manitoba and Manitoba Hydro, is designed to provide advice and guide-lines for the use of EMMA and to assist in data preparation.

* *

EMMA optimizes operation of an interconnected power utility using linear programming (LP).

Do you want to go through a brief explanation of linear programming?

YES
NO

* LINEAR PROGRAMMING (page 1) *

Linear programming (LP) solves problems where limited resources must be allocated among competing activities the best possible way. LP describes this problem using a mathematical model. The adjective 'linear' refers to the fact that all relationships in the model must be linear. The word 'programming' is a synonym for planning and does not refer to anything related to computer programming.

* LINEAR PROGRAMMING (page 2) *

LP problems are defined by three basic concepts:

1 - All the competing activities in the model are represented by variables that describe their levels or intensities. Different activity levels can be chosen from a set of permitted, or possible, values and are therefore called DECISION VARIABLES.

2 - LP CONSTRAINTS are linear mathematical relationships (or conditions) that one activity or several inter-related activities have to satisfy. Any solution of an LP problem has to satisfy all the constraints, and in that case, it is called feasible. If constraints in a LP problem are conflicting, no feasible solution can be found, and the problem is called infeasible.

3 - The costs and revenues associated with activities provide information that is used in choosing the optimal solution. The sum of products of the activity levels times associated activity costs/revenues give the measure of how good the solution is. This sum is called the OBJECTIVE FUNCTION.

The combination of values of decision variables that satisfies all the constraints in an LP problem while maximizing (or minimizing, if so required) the value of the objective function is called OPTIMAL SOLUTION. LP can be viewed as a set of rules that ensure that the search for the optimal solution will be successful.

* LINEAR PROGRAMMING (page 3) * EXAMPLE:

1 - A System manager has to decide how much energy to produce at each of the existing power plants A,B,and C in the next month in order to meet the estimated demand D. Since the energy productions a, b, and c at stations A, B, and C respectively are a matter of his/her decision, they are DECISION VARIABLES in this case.

2 - If the system manager cannot expect to import from nor to export energy to other utilities, the demand must be satisfied using only A,B,and C. This is represented in this LP problem by the following CONSTRAINT:

$$a + b + c = D \quad (D \text{ is a constant}).$$

Since none of energy productions can be negative, it is obvious that:

$$a >= 0 \quad , \quad b >= 0 \quad , \text{and} \quad c >= 0 .$$

All these productions are also limited by the installed capacities at A, B, and C: $a <= MaxA$, $b <= MaxB$, and $c <= MaxC$

(MaxA, MaxB, and MaxC - maximum energy output of A,B and C for the next month)

3 - If the production costs at A,B, and C are Ca, Cb, and Cc per unit of energy, a simple OBJECTIVE FUNCTION can be defined as:

$$OF = a*Ca + b*Cb + c*Cc.$$

The set of energy productions a,b and c that minimizes the objective function without violating any of the constraints is the OPTIMAL SOLUTION

* LINEAR PROGRAMMING (page 4) *

Although energy production at a hydro generating station does not depend linearly on the flow through the turbines (head also has to be taken into account), the reservoir control and generation allocation problem of a hydroelectric utility is well suited to linear programming.

EMMA overcomes this non-linearity by assuming some initial production coefficients based upon an estimate of the heads for the hydro generating stations of the system. The solution of this linear programming problem is then found and the initial assumption about heads and production coefficients is corrected. The LP problem is formulated again, and the same procedure is repeated till the convergence criteria are met.

EMMA time horizon is the time interval for which the operation of the system is planned (for example: one year). This interval is broken into time steps (for example: months, or weeks), for which the input is specified (flows, loads, etc...) and the results are calculated (optimal releases, power generation for each station, maintenance, etc...). As a computer program, EMMA accepts as input data:

1 - Hydraulic network that consists of lakes and flows. Inflows into lakes are defined by the flow scenario. Lakes can be connected by one or more of the following three flow types: natural outlet, control structure, and hydro generating station. Flows through natural outlets depend on the water levels in the lakes that they connect, while flows through control structures and hydro generating stations are decision variables in EMMA.

2 - Electrical network that consists of: hydro and thermal generating stations, one or more tieline interconnections with adjacent utilities, a system demand defined by the load scenario, and energy import and export market characteristics.

These systems interconnect where a generating station is located on a river. The LP model decides how to operate the hydraulic and electrical network in order to produce the required energy while maximizing the net revenue.

EMMA also provides the possibility of constraining flows, levels, productions, andimposing group performance on selected stations when conditions require to do so.

DECISION VARIABLES in the LP problem in EMMA are calculated for each time step. They include: reservoir releases, energy generation and imports and exports, and maintenance. CONSTRAINTS that are taken into account are:

- continuity of water flow and volume
- equality between the initial and the final storage in a lake (optional)
- equality that defines relation between a turbine flow and energy generation
- bounds on outflows from control structures, spillways, and power-houses
- sum of export and import time on a tieline is not greater than total time
- shaping (generation at a station decreases from on-peak to off-peak load)
- limit on energy imports
- equality between supply and demand
- limit on maximum change in storage in a reservoir
- grouping that imposes group performance on a selected set of stations (optional)
- limit on water temperature in flows that are used for cooling of thermal stations
- limit on increase in water temperature in flows used for cooling thermal stations
- limit on energy production in a power plant
- minimum maintenance of system elements
- constraints that make sure that maintenance schedules respect crews availability.

OBJECTIVE FUNCTION maximizes revenues from selling energy plus value of water in storage at the end of time horizon, minus costs of releases, energy generation and imports, and maintenance. Attributing value to water allows consideration of future value of energy, outside the planning horizon, and also trade off within the planning horizon for competing uses such as recreational benefits or flood damages.

reads "EMMA performs the following functions:

1- Reads data, checks for obvious errors, and produces reports summarizing the data,

2- Formulates the linear programming problem that is defined by the data,

3- Solves the linear programming problem,

4- Presents the solution to the problem in a number of different reports.

* EMMA can deal with different or changing power systems (networks) and different time horizons.

* The only limit to the size of the problem solved is the capability of the computer being used. In general, there is a trade-off between the complexity of the network and the number of time steps to the time horizon that can be handled in a single problem.

* Different contexts for the use of EMMA, as well as each data entry, will be explained in detail during the course of this Expert System consultation.

There are three basic motivations for the use of EMMA - select one:

1 - ANNUAL BUDGET preparation and updates,

2 - preparation of OPERATIONAL PLAN of releases, thermal and hydro generation, maintenance, exports, and imports.

3 - LONG TERM STUDY of the system (benefits from additional capacities, some operational condition or a change in the system).

The following 5 steps are necessary in order to obtain results that can be used in preparation or updates of the annual budget. Select one for further explanation!

1_FIRMNESS_TEST,
2_SEVERE_CASE,
3_EXPECTED_CASE,
4_SEVERE_CASE_RE_RUNS,
5_CONVERGENCE_TEST,
end_consultation

Is the system in a drought?

*System is in drought if reserves are below usual and inflows are not big enough to enable sufficient energy production, thus making it necessary to use water in storage.

yes
no

The following 5 steps are necessary in order to prepare an operational plan of releases and power generation. Select one for further explanation!

1_FIRMNESS_TEST,
2_SEVERE_CASE,
3_EXPECTED_CASE,
4_SEVERE_CASE_RE_RUNS,
5_CONVERGENCE_TEST,
6_end_consultation"

The following 3 steps are necessary in order to prepare an operational plan of releases and power generation. Select one for further explanation!

1_FIRMNESS_TEST,
2_EXPECTED_CASE,
3_CONVERGENCE_TEST,
4_end_consultation"

The following 5 steps are necessary in order to obtain results that can be used in long term studies of the system performance. Select one for further explanation!

1_FIRMNESS_TEST,
2_EXISTING_SYSTEM,
3_CONVERGENCE_TEST,
4_CHANGED_SYSTEM,
5_CONVERGENCE_TEST,
end_consultation"

EMMA SEVERE CASE should be run first. This means assuming a water supply with a low probability (say 5% probability of exedence) and assuming early freeze-up date. This run will formulate a release and energy production plan that will satisfy system demand in case when operating conditions are severe. Keeping firmness test reserves ensures that the system will be capable of coping with contingencies if they should occur on top of the described severe conditions.

EMMA EXPECTED CASE should be run next. This means taking the water supply with 50% probability of exceedence and assuming expected freeze-up date. This run will formulate a release and energy production plan that will satisfy demand in case of the likeliest flow and loads scenario. Note that hydro generation will be larger in this than in the severe case (more water available) and expenses for non hydraulic energy supplies will be smaller.

However, neither operational plan nor annual budget can be prepared based on results of this run because the system should be able to perform if the severe scenario should occur. What can be done, is increase hydro generation and releases in the immediate future for the severe scenario and compensate that with expensive thermal generation in later time steps. This way, the system will perform economically in the immediate future, and in later time steps it has a small probability (probability that the severe scenario really occurs) of paying penalty for this. In most cases (95%) flows will be larger than predicted in the severe case, and the system will avoid penalty.

SEVERE CASE should be re-run iteratively, gradually increasing lower bounds on hydro generation for the first three or four time steps (thus forcing it to increase). This however, must not exceed the corresponding values for hydro generation in the expected case. This incremental increase should be stopped when EMMA starts drawing up on the system's reserves determined in the FIRMNESS TEST.

Generally, utilities face the possibility that contingencies can cause problems in supplying energy to customers. These contingencies include major outages or severe droughts. Utilities should have energy reserves available at any time to cope with these contingencies.

For users of EMMA, energy reserves can be maintained by planning for energy in storage at the end of the planning horizon. The optimal policy formulated by EMMA will not draw up on these reserves which would make it vulnerable to possible future contingencies. Without reserves, EMMA would try to increase revenues by selling additional energy from reservoir storage.

The amount of reserve energy held in reservoir storage at the end of the time horizon is determined by performing the FIRMNESS TEST. These energy reserves are allocated to specific reservoirs in the input data.

FIRMNESS test usually consists of the following steps:

1 - Evaluate the scenario you want your system to be prepared for (Drought, outage, etc)

2 - Evaluate the reserves that should be available to the system if the scenario from Step 1 should occur.

3 - Evaluate the means of coping with it.

Choose the step number for further explanation or 'continue'!

The scenarios to be looked at are major outages or severe droughts:

Outages to generation or transmission facilities that affect the energy supply should be considered when setting energy reserves. The outage should be a low probability event.

Drought will probably be the largest contingency that will affect the energy supply to a hydro utility. The larger the hydraulic component of the system the larger energy reserves for drought. Drought reserves are usually for droughts outside the planning horizon. For outages in the planning period droughts reserve may have to be used. Drought reserves are usually set to withstand the worst drought on record, thus making sure that the system will have enough reserves to meet energy demand even if the worst sequence of low flows on record should repeat. These reserves also protect the system from outages and other contingencies because the probability of simultaneous severe droughts, outages, and spells of extremely cold weather (high consumption) is especially small.

** The worst drought on record is identified through the following steps:

1 - Calculate the total energy production in the system for the recorded period of flow records using only inflows to the system and not water drawn from storage. To this series of hydraulic generation add thermal generation and committed imports.

2 - Compare the energy supply from Step 1 with the demand that has to be met and identify periods when there is deficiency of energy.

3 - The worst drought on record is the one with the largest cumulative energy deficiency (sum of deficiencies from the beginning till the end of the drought).

1 - The occurrence of the scenario from Step 1. should be varied (moved back and forth) in time around the time horizon.

2 - Energy demands have to be estimated for the whole time span covered by the cases in step 1.

3 - For each time variation from Step 1, energy deficiencies for for each time step (usually month) should be calculated using the estimated demands from step 2.

4 - Going backwards in time (adding time step energy deficiencies), it is possible to calculate reserves that are necessary in the system at any point in time in order to cope with the considered scenario.

5 - The values from step 4 that correspond to the time horizon should be compared and the largest required energy reserve corresponds to the worst scenario the system should be protected from. This reserve is the result of the FIRMNESS TEST and it has to be left in the system at the end of time horizon in each EMMA run

The amount of energy evaluated in Step 2 is only available to the system at the end of the planning horizon in EMMA runs.

This usually means planning this energy in storage in the reservoirs of the system.

The reserves can be allocated among the reservoirs of the system in the following order:

1 - Multipurpose reservoirs are targeted to minimum levels equivalent to their normal levels at the end of the time horizon.

2 - Energy in storage between the normal level and the minimum level is calculated and subtracted from the reserve requirement. The remaining energy reserves,if there are any, are allocated iteratively to the other reservoirs of the system, filling them gradually in the way that minimizes the probability of spillage.

* When all the necessary reserves cannot be allocated to the reservoirs of the system the system needs additional generation capacities, new reservoirs, or water diversions in order to firm up the energy supply.

The reserves allocated in the firmness test are given the highest value exceeding any other source of energy or export value in the system. This forces the LP algorithm to use all possible alternatives before it starts using these reserves to meet the energy demands.

If energy reserves are required to meet the firm energy demands reservoirs will be
drawn below target levels at the end of the planning horizon. This is checked by looking at the levels of these reservoirs in EMMA output.

CONVERGENCE TEST:

Levels in the system's reservoirs and forebays operated according to operational rules should be now re-calculated manually, based on the releases from the latest EMMA run.

EMMA should be re-run with these levels in order to make sure that the operating policy converges.

EMMA EXPECTED CASE should be run. This means taking the sequence of flows with 50% probability of exceedence and assuming expected freeze-up date. This run will formulate a release and energy production plan that will satisfy demand in case of the likeliest flow and loads scenario.

System without the examined change is first entered to EMMA and different sequences of flows (5%, 25%, 50%, 75%, 95% probability of exceedence.) or scenarios of interest are examined in order to evaluate performance of the system.

System with the examined change built in is entered to EMMA and different sequences of flows (5%, 25%, 50%, 75%, 95% probability of exceedence.) or scenarios are repeated in order to evaluate the effect of the examined change in the system. Note that this procedure gives a fair comparison of system performance with and without the examined change, because it takes its optimal response in both cases.

Note that the resulting energy productions, reservoir levels, and other physical data obtained for the utility with and without the examined change should be used for further evaluation of benefits and not values of EMMA objective function itself (EMMA objective function gives only relative dollar values that can be compared only within one EMMA run).

In addition to the advice on input data for this EMMA run, this expert system can help create an input data set for running EMMA.

Do you want to create input for EMMA during this consultation?

YES,
NO

Select:

COMPLETE -if you want to go through all data modules, or

NECESSARY -if you are familiar with all modules from the previous SEVERE CASE EMMA run, and want to look only at those that have to be changed for this run.

COMPLETE,
NECESSARY

There are up to 17 different data modules that have to be prepared before running EMMA. This expert system can explain the use and preparation of all these modules. Select:

GUIDE - to let expert system guide you through all necessary modules,
SELF - to choose and go through data modules interactively,
QUIT - to quit advice on input for this EMMA run."

Select a data module for assembly and/or advice, or CONTINUE!

job_title,
report_selection,
LP_parameters,
problem_size,
time_structure,
loads,
lakes,
hydro_stations,
control_structures,
natural_outlets,
inflows,
other_stations,
tielines,
strip_groups,
time_groups,
period_groups,
maintenance,
CONTINUE

Select a data module for assembly and/or advice, or CONTINUE!

time_structure,
loads,
inflows,
CONTINUE

DATA MODULE 1 - JOB TITLE:

The first line of input data set contains the name of the EMMA run as well as any other pertinent information you want to save.

DATA MODULE 2 - REPORT SELECTION (page 1):

EMMA can produce up to 30 different reports. This data module controls which reports are printed or deleted from the output listing. The input of these data consists of the string of numbers from 1 to 30. A report is selected if its number is positive and is deleted if its number is made negative. The whole string must be followed by a zero to indicate the end of the report selection.

- Reports recommended for this EMMA run:

17. Production coefficients report
19. Energy assignment report
20. Capacity assignment report
22. Lake stage report
23. Flow summary report

* Report 17 gives information on convergence of the production coefficients in the system. If all coefficients have not converged (look at the last iteration) with acceptable tolerance then EMMA should be run again with more iterations.

** Report 22 gives information necessary to check whether the reserves determined in the firmness test are present in the system at the end of time horizon.

*** Reports 19, 20, and 23 give good overview of the system's operation. Report 19 is the most important for budgetary runs because energy productions calculated by EMMA enter other models that are used for budget preparation.

DATA MODULE 2 - REPORT SELECTION (page 2):

- In case of infeasibility, EMMA can be re-run for the purpose of creating:

24. Flow continuity report
27. Debug LP report

- In case when binding constraints are of special interest:

25. LP solution report

- When maintenance scheduling is of special interest, the following reports can be created:

28. Capacity availability report
29. Maintenance scheduling report
30. Maintenance graphs

DATA MODULE 2 - REPORT SELECTION (page 1):

EMMA can produce up to 30 different reports. This data module controls which reports are printed or deleted from the output listing. The input of these data consists of the string of numbers from 1 to 30. A report is selected if its number is positive and is deleted if its number is made negative. The whole string must be followed by a zero to indicate the end of the report selection. Reports recommended for this EMMA run:

17. Production coefficients report
19. Energy assignment report
20. Capacity assignment report
22. Lake stage report
23. Flow summary report

* Report 17 gives information on convergence of the production coefficients in the system. If all coefficients have not converged (look at the last iteration) with acceptable tolerance then EMMA should be run again with more iterations.

** Report 22 gives information necessary to check whether the reserves determined in the firmness test are present in the system at the end of time horizon. This is the most important report to be looked at in these EMMA runs. These runs are repeated with gradually higher lower bounds on hydro generation in immediate future, until the system starts spending reserves determined in the firmness test. This is detected by looking at water in storage at the end of the time horizon in report 22.

*** Reports 19, 20, and 23 give good overview of the system's operation.

DATA MODULE 2 - REPORT SELECTION (page 2):

- In case of infeasibility, EMMA can be re-run for the purpose of creating:
  24. Flow continuity report
  27. Debug LP report

- In case when binding constraints are of special interest:
  25. LP solution report

- When maintenance scheduling is of special interest, the following reports can be created:
  28. Capacity availability report
  29. Maintenance scheduling report
  30. Maintenance graphs

DATA MODULE 2 - REPORT SELECTION (page 1):

EMMA can produce up to 30 different reports. This data module controls which reports are printed or deleted from the output listing. The input of these data consists of the string of numbers from 1 to 30. A report is selected if its number is positive and is deleted if its number is made negative. The whole string must be followed by a zero to indicate the end of the report selection.

- Reports recommended for this EMMA run:
  17. Production coefficients report
  19. Energy assignment report
  20. Capacity assignment report
  22. Lake stage report
  23. Flow summary report

* Report 17 gives information on convergence of the production coefficients in the system. If all coefficients have not converged (look at the last iteration) with acceptable tolerance then EMMA should be run again with more iterations.

** Report 22 gives information necessary to check whether the reserves determined in the firmness test are present in the system at the end of time horizon.

*** Reports 19, 20, and 23 give good overview of the system's operation. Report 20 is the most important for the preparation of the operational plan.

DATA MODULE 2 - REPORT SELECTION (page 2):

- In case of infeasibility, EMMA can be re-run for the purpose of creating:
  24. Flow continuity report
  27. Debug LP report

- In case when binding constraints are of special interest:
  25. LP solution report

- When maintenance scheduling is of special interest, the following reports can be created:
  28. Capacity availability report
  29. Maintenance scheduling report
  30. Maintenance graphs

DATA MODULE 2 - REPORT SELECTION (page 1):

EMMA can produce up to 30 different reports. This data module controls which reports are printed or deleted from the output listing. The input of these data consists of the string of numbers from 1 to 30. A report is selected if its number is positive and is deleted if its number is made negative. The whole string must be followed by a zero to indicate the end of the report selection.

- Reports recommended for this EMMA run:
17. Production coefficients report
19. Energy assignment report
20. Capacity assignment report
22. Lake stage report
23. Flow summary report

* Report 17 gives information on convergence of the production coefficients in the system. If all coefficients have not converged (look at the last iteration) with acceptable tolerance then EMMA should be run again with more iterations.
** Report 22 gives information necessary to check whether the reserves determined in the firmness test are present in the system at the end of time horizon.
*** Reports 19, 20, and 23 give good overview of the system's operation.

DATA MODULE 2 - REPORT SELECTION (page 2):

- In case of infeasibility, EMMA can be re-run for the purpose of creating:
    24. Flow continuity report
    27. Debug LP report

- In case when binding constraints are of special interest:
    25. LP solution report

- When maintenance scheduling is of special interest, the following reports can be created:
    28. Capacity availability report
    29. Maintenance scheduling report
    30. Maintenance graphs

DATA MODULE 3 - LP PARAMETERS (page 1): These data control the operation of the LP algorithm. The following are required (recommended values are in brackets):

1. Absolute tolerance for LP zero tests (1E-9)
2. Relative tolerance for LP zero tests (1E-9), 3. and LP zero sums (1E-12)
4. Maximum column coefficient ratio (1E+8)
5. Rational tolerance (1E-7) , 6. Rational factors (7200)
7. Absolute tolerance for change in objective function (100)
8. Max. no. of consecutive changes in objective function factor (.10)
9. Max. number of iterations factor (500)
10. LP coefficient allocation block factor (3.1),11. LP option bits: (10000)
12. LP strategy bits: FF/FO/FF_SS/FO_SS (1111)
13. LP force feasible type (Z, 1, I, D, L) (D)
14. LP force optimal type (Z, 1, I, D, L) (D)
15. Maximum number of re-inversion tries (3)

*The LP package used in EMMA is called MERLIN. It is a general purpose procedure that will accept and solve any LP problem. MERLIN accepts parameters that maintain and monitor the numerical performance of the LP algorithm from EMMA. The use of these parameters requires a substantial knowledge of linear programming and MERLIN in particular. The normal procedure would be to use the values recommended above, and change them only if necessary and with the help of someone familiar with MERLIN.***

DATA MODULE 3 - LINEAR PROGRAMMING PARAMETERS (page 2):

The following data are required (recommended values are given in brackets):

16. Production coefficient convergence tolerance (0.05)
17. Maximum number of stages for production coefficient convergence (4)
18. Output basis at (REST)

* Parameters 16 and 17: EMMA uses an iterative process outside the LP to come to a final solution. This is necessary because the production function is non-linear: POWER(MW) = DISCHARGE(KCFS) x HEAD(m) x COEFF, which makes a direct LP solution impossible. At each iteration, EMMA assumes that the heads are fixed, and then solves thus formulated LP problem using production coefficients calculated from the assumed heads. After that, EMMA calculates the heads that would result from the outflows presented in the solution. It then recalculates the production coefficients and enters a new iteration. Iterations are stopped when all the production coefficients are within convergence tolerance (item 16 above) from those calculated in the preceding iteration, or when the total number of iterations reaches the user-defined maximum number (item 17).*

** Parameter 17: A negative number of stages will result in a partial solution being printed after each iteration. The minus sign will not affect the maximum number of stages for production coefficient convergence.**

DATA MODULE 4 - NETWORK AND CONSTRAINT SIZE:

This module defines the size of the physical problem and the number of grouping constraints. If the size of the problem becomes a problem to EMMA, report #2 (parameter data) gives the size limits that can be accepted by the LP algorithm.

The data are required in the following order (explanations are given in brackets):

1. Number of timesteps - T (This item can be obtained from MODULE 5. It is convenient to complete MODULE 5 before completing MODULE 4. This item is very important for data entry in other data modules, and will be referred to as T, later in this consultation)
2. Number of lakes in the system
3. Number of hydro generating stations in the system
4. Number of control structures in the system
5. Number of natural outlets
6. Number of local inflows
7. Number of generation sources (other than hydro generating stations)

9.Number of generation strip groups (Explained with DATA MODULE 14)
10.Number of generation time groups (Explained with DATA MODULE 15)
11.Number of generation period groups (Explained with DATA MODULE 16)

12. Number of maintenance crews
13. Maintenance allowed flag (YES or NO)

DATA MODULE 5 - TIME STRUCTURE (page 1):

EMMA time horizon is the time interval for which the operation of the system is optimized (for example: one year). This interval is broken into time steps (for example: months, or weeks), for which the input is specified (flows, loads, etc...) and the results are calculated (optimal releases, power generation for each station, maintenance, etc...).

When EMMA is run for annual budget preparation or updates:

- time horizon is till the end of the fiscal year,

- recommended time steps are 1 month each.

Total number of time steps is = 12 for budget preparation, or smaller for budget updates. This total number of time steps will be necessary later, for preparation of DATA MODULE 4 - NETWORK AND CONSTRAINT SIZE.

DATA MODULE 5 - TIME STRUCTURE (page 1):

EMMA time horizon is the time interval for which the operation of the system is optimized (for example: one year). This interval is broken into time steps (for example: months, or weeks), for which the input is specified (flows, loads, etc...) and the results are calculated (optimal releases, power generation for each station, maintenance, etc...).

When EMMA is used for preparation the operational plan of the system:

- time horizon is till the end of the fiscal year,

- recommended time step length:
- first 3-4 steps of 1 week (so that the last one finishes at the end or middle of a month)
- next 2-3 steps of 2 weeks (so that the last one finishes at the end of a full month)
- time steps of 1 month thereafter, till the end of the fiscal year.

Time steps determined as shown above, should be counted to obtain their total number T. This total number of time steps will be necessary later, for preparation of DATA MODULE 4 - NETWORK AND CONSTRAINT SIZE.

DATA MODULE 5 - TIME STRUCTURE (page 1):

EMMA time horizon is the time interval for which the operation of the system is optimized (for example: one year). This interval is broken into time steps (for example: months, or weeks), for which the input is specified (flows, loads, etc...) and the results are calculated (optimal releases, power generation for each station, maintenance, etc...).

When EMMA is run for long term study of the system:

- time horizon is one year,

- recommended time steps are 1 month each.

Total number of time steps is T=12. This information will be necessary later, for preparation of DATA MODULE 4 - NETWORK AND CONSTRAINT SIZE.

DATA MODULE 5 - TIME STRUCTURE (page 2): For each time step, the following data items are required (explanations are given in brackets):

1. Name (Usually the name of the month, or any other convenient string. Use only alphanumeric characters and underscores, no blanks are allowed.)

2. Length (Enter number of days in the timestep)

3. Season type (This keyword may be either OPEN_WATER or ICE_COVER. This is important because tailwater curves for the hydro generating stations, and therefore energy production, depend upon the season type.)

4. Shaped (YES or NO. This flag controls whether any shaping constraints invoked by the generating stations and generation sources will be allowed in the time step. It allows making shaping constraints inactive in the time steps when they are not important, in order to save computer time. Shaping constraints are explained in DATA MODULES 8 and 12)

* The four items described above are entered as many times as there are time steps. EMMA has the option of solving a shorter problem than specified. If any of the time step names (item 1) is preceded by an asterisk (*) the corresponding time step is considered to be the first, and the time steps before it are ignored in the LP problem.

DATA MODULE 5 - TIME STRUCTURE (page 3):

ADVICE:

Item 3 - Assume severe winter with an early freeze-up date, the ICE-COVER time steps start earlier, and last longer during EMMA time horizon!

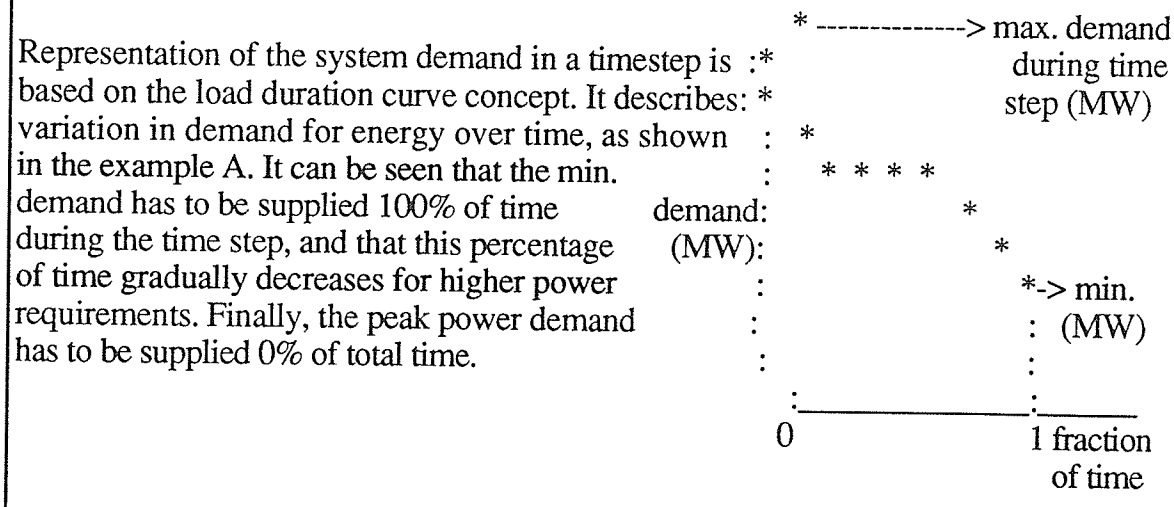
DATA MODULE 5 - TIME STRUCTURE (page 3):

ADVICE:

Item 3 - Assume a winter with the most probable freeze-up date, the ICE-COVER time steps start and last till expected dates!

DATA MODULE 6 - LOAD DURATION CURVE (page 1):

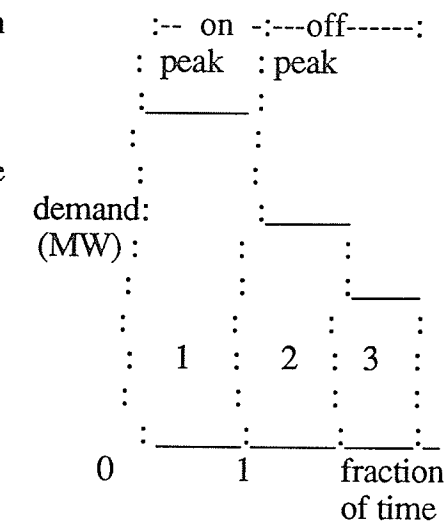
Representation of the system demand in a timestep is based on the load duration curve concept. It describes: variation in demand for energy over time, as shown in the example A. It can be seen that the min. demand has to be supplied 100% of time during the time step, and that this percentage of time gradually decreases for higher power requirements. Finally, the peak power demand has to be supplied 0% of total time.



Example A - Typical Load Duration Curve

DATA MODULE 6 - LOAD DURATION CURVE (page 2):

The load duration curve in EMMA is represented in linear form, using a set of vertical strips. The height of each strip represents the average system demand for the corresponding percent of time. The strip designation is also used to divide the load curve into on peak and off peak strips. A minimum one on peak and one off peak strip is required. This is necessary for the differentiation in energy costs (it is more expensive during on peak hours) and values in the import and export markets. The Example B shows linearization of a load duration curve using two on peak and one off peak strip. On peak and off peak hours are by by definition and determine the pricing policy for interutility transactions. Generally, on peak hours are Monday to Saturday 6.00 a.m. to 10.00 p.m. (16 hours). Of peak are all other hours plus 6 specified US holidays.



Example B - Strip Representation of Load Duration Curve

DATA MODULE 6 - LOAD DURATION CURVE (page 3):

This data module describes the reserve capacity, the load duration curve for each time step, and the maximum fraction of energy demand allowed from imports. The following items have to be entered (explanations are given in brackets):

1. Reserve capacity (In MW. This maintains capacity reserve in the system. It is usually set to be equal to the largest single unit in the system, so that the system can operate safely in case of a major outage. The other way to maintain sufficient reserve in the system is to set this item to zero and enter bigger loads with smaller probabilities. The latter is recommended and explained later during this consultation! A reserve capacity has to be entered for each time step, so it is necessary to enter a sequence of T numbers, or use the key word ALL followed by the constant value of capacity reserve for all timesteps.)

2. Number of on peak strips (At least 1, recommended value 2! The number of strips is usually determined by the minimum that can adequately represent the nonlinear load duration curve. This item is required for each time step, so it is necessary to enter T values, or use the key word ALL followed by the number of on peak strips for all timesteps.)

3. Number of off peak strips (At least 1, recommended value 1! The number of strips is usually determined by the minimum that can adequately represent the nonlinear load duration curve. This item is required for each time step, so it is necessary to enter T values, or use the key word ALL followed by the number of off peak strips for all time steps.)

DATA MODULE 6 - LOAD DURATION CURVE (page 4):

4. Strip width (Fraction of total time within a time step that corresponds to the strip of the load duration curve. Number of values that are required is equal to the sum of the items 1 and 2, and all strip widths have to sum up to 1.0! The sequence of values of strip widths has to be entered T times, in order to cover all time steps.)

5. Peak instant load and 6. System demand (For each time step: peak instant load in MW's and demands in MW's for all strips, in descending order, have to be entered. This has to be repeated T times in order to cover all the time steps.)

7. Firm system value (T values in $/GWh for all the time steps may be specified, or a single constant value for all time steps can be entered preceded by the key word ALL)

8. Maximum fraction of system demand allowed from imports (This item is used to limit the dependance on other utility generation. T fraction values, from 0.0 to 1.0, may be entered, or only one constant value preceded by the key word ALL. Note that 0 is a coded value which allows unlimited imports, and that entering 0.0001 or a similar small value actually eliminates imports.)

DATA MODULE 6 - LOAD DURATION CURVE (page 5) - ADVICE:

On peak loads can generally be modelled using 2 strips. The system demand, item 6, for the first strip can be taken to be the average load of the 4 highest loaded hours for all on peak days. The strip width, item 4, for the first strip in this case would be: $4x6 / 7x24 = 0.14$.

Off peak hours are Sundays, some US holidays, and all other days from 10.00 p.m. to 6.00 a.m. Because of less non linearity, these hours can be modelled using only one strip. The strip width in this case would be: $(24 + 6x8) / 24 \times 7 = 0.43$, and the system demand is average load during these hours.

Obviously, for the second on peak strip, the width is $1.0 - 0.17 - 0.43 = 0.40$, and the system demand is the corresponding average load.

* More detailed representation of the on peak is desirable because the surplus on peak capacity determines how much capacity and energy can be sold to neighboring utilities. In the off peak, one strip is required just to determine total energy.
** Item 8: It is always preferred to identify an external source of energy and model it directly using tieline data, module 13. than to use this item to limit imports of energy. Even so, this item can be used to limit the dependence on one supplier.

DATA MODULE 6 - LOAD DURATION CURVE (page 5):

CONSTRUCTION OF THE LOAD DURATION CURVE - ADVICE:

It is possible, based on 52 occurrences in the past year, to find the relationship between temperature and demand for energy, for each hour of the week.

Using that relationship, it is possible to calculate energy consumptions that correspond to different historic sequences of temperature.

This enables calculation the mean and standard deviation for energy consumption, as well as energy demands with different probabilities for each EMMA time step.

The hourly loads can be adjusted to account for the growth of consumption between the last year and the year EMMA is run for.

For the severe case, time step energy demands have to be chosen so that the cumulative probability for the overall energy demand for the whole time horizon is 95%.

DATA MODULE 6 - LOAD DURATION CURVE (page 5):

CONSTRUCTION OF THE LOAD DURATION CURVE - ADVICE:

It is possible, based on 52 occurrences in the past year, to find the relationship between temperature and demand for energy, for each hour of the week.

Using that relationship, it is possible to calculate energy consumptions that correspond to different historic sequences of temperature.

This enables calculation the mean and standard deviation for energy consumption, as well as energy demands with different probabilities for each EMMA time step.

The hourly loads can be adjusted to account for the growth of consumption between the last year and the year EMMA is run for.

For the expected case, the time step energy demands with 50% probability of exceedence have to be chosen.

DATA MODULE 6 - LOAD DURATION CURVE (page 5):

CONSTRUCTION OF THE LOAD DURATION CURVE - ADVICE:

It is possible, based on 52 occurrences in the past year, to find the relationship between temperature and demand for energy, for each hour of the week.

Using that relationship, it is possible to calculate energy consumptions that correspond to different historic sequences of temperature.

This enables calculation the mean and standard deviation for energy consumption, as well as energy demands with different probabilities for each EMMA time step.

The hourly loads can be adjusted to account for the growth of consumption between the last year and the year EMMA is run for.

For the long term study, time step energy demands have to be chosen in accordance with the examined scenario of loads in the system.

DATA MODULE 7 - LAKES (page 1): This module has to be repeated for each lake in the network. A lake can be either:

- a reservoir controlled by a hydro station or a control structure, or
- a lake with a natural outlet.

Each lake is described by data as follows (explanations are given in brackets):

1. Name of lake (Enter the name of the lake using only alphanumeric characters and underscores - no blanks are allowed)
2. To be printed (Enter flag YES or NO. This controls whether the lake appears in the report)
3. Number of storage segments M (It may be necessary to model the lake with several segments to account for its non linear storage discharge relationship. This data item specifies the number of segments, and is called M later in these explanations. If the number M is made negative then the storage at the end of the final time step may be subdivided into figments. Figments are finer subdivisions of segments which only exist at the end of the time horizon and are used to reflect more closely the the non linear value of water in storage. Figment data are further discussed under item 13.)
4. Initial stage in feet (This is the initial elevation of the reservoir at the beginning of the problem. This may be the first timestep or the timestep marked with an asterisk in DATA MODULE 5. If the initial level is to be fixed at a certain elevation then the keyword FIX is followed by that elevation. If the initial level is to be left unconstrained in order to find its optimal value then the keyword FREE is used. This keyword must not be followed by an elevation.)

DATA MODULE 7 - LAKES (page 2):

5. Final stage in feet (This is the final elevation of the reservoir at the end of the last time step. If the final level is to be fixed at a certain elevation then the key word FIX is followed by that elevation. If the final level is to be left unconstrained in order to find its optimal value then the keyword FREE is used. The final stage can also be constrained to be equal to the initial stage by entering the keyword SAME. The keywords FREE and SAME must not be followed by an elevation.)

6. Minimum stage, and 7. Maximum stage, both in feet (All elevations in EMMA are at the time step boundaries, so there is always one more boundary than the total number of time steps. This means that item 6 has to be repeated T+1 times, and then the same has to be done with item 7. The levels of the lake are constrained to be between these minimum and maximum values at each time step boundary. This item can be used to constrain levels in the reservoirs that are operated according to operating rules, or constrained by some other factor.)

8. Maximum storage change in kcfs-days (This data constrains the change in storage allowed within a time step. This item can prevent fast and large fluctuations of the lake level when necessary. Zero is a coded value taken to mean no limit on change of storage.)

DATA MODULE 7 - LAKES (page 3):

9. Maximum segment stage in feet (This item describes the maximum elevation to which each segment of the storage is bound. An input of 0 can be used for the last segment, and it is taken to mean no maximum limit. M values, one for each segment, have to be specified.)

10. Slope of stage-storage curve in kcfs-days/foot (This item has to be entered for each segment, M values are required.)

11. Intercept of stage-storage curve in kcfs-days (this item has to be entered for each segment, M values are required. For each segment, the corresponding pair of items 10 and 11 is used to define the stage-storage curve within that lake segment.)

12. Value of storage in $/kcfs-day (Water in storage in each segment of a lake may be given a value (positive number) or a cost (negative number). Item 12 is usually the expected value of water in storage, calculated based on the total amount of energy that the system can produce using this water. This value has to be discounted by the probability of spillage which can be calculated for each segment of the lake and for each time step based on historic data. A cost could be assigned to a segment if it is a part of a flood reserve.)

DATA MODULE 7 - LAKES (page 4):

13. For each segment at the last time bound, if the number of segments M is made negative, the following data are required:

a) number of figments F

b) maximum figment stage in feet (Repeated F times)

c) figment storage value in $/kcfs-day (Repeated F times)

(a), (b), and c) has to be repeated for each segment. This subdivision of segments is introduced in order to model more closely the value of water in storage. These data are used only for the segments that are divided into figments at the end of the time horizon. For all other segments, that are either full or empty at the end of the time horizon, data from item 12 are used.)

DATA MODULE 7 - LAKES (page 4):

ADVICE:

Item 4 - The initial water level has to be adjusted to reflect the linear approximation of the stage storage curve. Use the elevations that correspond to the actual volume of water in storage in the reservoirs of the system with respect to their linearized stage storage curves. This defines total amount of energy stored in the system. Use the keyword FIX followed by the elevation in feet for each lake.

Item 5 - Normally left free. This way, the final water level is determined by EMMA based on the value of water in storage.

For the forebays in the system, which are generally regulated for local or real time considerations, minimum and maximum levels (items 6 and 7) can be fixed to the median levels determined from historic data and initial and final levels (items 4 and 5) are set free.

DATA MODULE 7 - LAKES (page 4):

ADVICE:

item 4 - Use free initial water level. This way, EMMA can choose optimal initial lake level and the corresponding release policy. This means using the keyword FREE for each lake of the system.

item 5 - Constrain final water levels to be equal to initial levels in the system. This way the energy supply is determined from inflows and storages are not mined during the study period. This means using the keyword SAME for each lake of the system.

DATA MODULE 7 - LAKES (page 4):

ADVICE:

For any forebay whose level is dependent upon the flow released, actual levels have to be updated by referring to the appropriate hydraulic relationship. The reservoir levels should be constrained using items 6 and 7, and EMMA should be run again in order to ensure that the release policy converges. The main effect will be on the head of the generating station whose forebay is constrained. PRESS RETURN TO CONTINUE

DATA MODULE 8 - HYDRO GENERATING STATIONS (page 1): This module must be repeated for each hydro generating station in the network. The data are required in the following order (explanations are given in brackets):

1. Name of the station (alphanumerics and underscores only, no blanks .)
2. To be printed? (Enter flag YES or NO)
3. Upstream lake (Enter the name of the forebay of the generating station, The corresponding lake data module (DATA MODULE 7) under that name must be present in the data set.)
4. Downstream lake (This is the name of the lake into which the generating station discharges. The corresponding lake data module (MODULE 7) under that name can be present in the data set. If the generating station discharge leaves the network, a name for the destination should be given.
5. Shaping constraint - Shaping of the generation is controlled by this keyword:

- When UNSHAPED, no shaping constraint is written.
- When SHAPED a shaping constraint is written, which ensures the capacity of the hydro station utilized decreases or remains uniform moving from on peak to off peak strips. This avoids random changes in loading the generating station.
- When BASE_LOADED the generation is forced to a uniform output i.e. no cycling of generation will be possible. The facility of shaping generation within a time step is subject to the status of the shaping flag for each time step set in DATA MODULE 5)

DATA MODULE 8 - HYDRO GENERATING STATIONS (page 2):

6. Run of river (When set to YES, this flag will force the operation of the station to match the operation of an upstream generation station strip by strip on the load duration curve. This is used for hydro plants that are close to each other, which means that the travel time is small relative to the time step length, and that do not have much storage capacity in the lake between them. The run of river constraint maintains flow continuity with the upstream plant on a strip basis, rather than on the normal time step basis. A run of river plant must always be downstream of another single hydro plant, and it cannot be base loaded. It cannot have a parallel plant, and it cannot be downstream from two parallel plants. When the hydro station is not run of river, enter flag NO.)
7. Number of units - U (Enter integer number of the plant units.)
8. Unit rated capacity in MW's (This is the maximum capacity under normal operating conditions. U numbers (item 7) have to be entered, one for each unit.)
9. Scheduled maintenance - in days (This is the length of time that the unit will be out of operation. EMMA is allowed to choose the best time for this maintenance. It is assumed that a crew will be working on the unit for the same period of time. U numbers (item 7) have to be entered, one for each unit.)

DATA MODULE 8 - HYDRO GENERATING STATIONS (page 3):

10. Number of maintenance timesteps, 11. Maintenance time step, and 12. Maintenance preference (Items 10 and 11 define a subset of all time steps during which scheduled maintenance is done. The scheduled maintenance for a plant can be constrained to occur in specific time steps, but not unit by unit. Item 10 defines total number of steps to which EMMA will be allowed to place scheduled maintenance. For these time steps a preference could be given, 1 being the highest priority. Any relative scheme can be used, provided that it is commensurate among all plants. Pairs of items 11 and 12 have to be repeated N times, once for each allowed maintenance timestep.)

13. Pre-scheduled maintenance - in days (The maintenance for units may be pre-scheduled, usually by the operator of the plant. This maintenance is fixed in time and EMMA is not allowed to move it. The number of days is assigned for each unit, which means that U numbers have to be entered for the first time step, and that the same has to be repeated for all other time steps.)

14. Maximum maintenance outage in MW (This is the upper limit on the capacity that maintenance can displace. It is usually set to be the maximum unit size so that maintenance on most one unit at a time is scheduled. T numbers have to be entered, one for each time step.)

15. Minimum capacity in MW (This is the minimum load from the hydro plant. It is necessary because all plants need to be heated and energized. T numbers have to be entered, one for each time step.)

DATA MODULE 8 - HYDRO GENERATING STATIONS (page 4):

16. Capacity derate in MW (It is entered for each strip of the LDC, and it models factors that cannot be controlled, such as ice conditions causing higher tailwater levels, high temperature of cooling water, or any other factor that reduces generating capacity. S numbers, one for each strip, are repeated for each time step, which means T times.)

17. Forced outage rate MW/MW (This is the number of forced outage hours divided by the sum of the forced outage hours and the running hours. Item 17 can be calculated based on historic data for the plant. In effect, this item reduces the hours available for generation and the energy generation ability of the plant.)

18. Forced maintenance days (This is the number of days a maintenance crew must allocate to forced maintenance. This reduces the time the crew can work on scheduled maintenance during the maintenance period)

19. Production coefficient in MW/KCFS (EMMA uses an iterative technique for determining the production coefficient. Initial estimate based upon normal plant head is required. When the keyword ALL is used, it is not necessary to define production coefficients for all strips and timesteps. If not, enter S (number of strips) values T times, to cover all load duration curve strips and time steps. The better this initial estimate the faster the EMMA solution will converge. When EMMA finishes an iteration it saves the updated production coefficients. These are used in the next iteration or are saved and used when the problem is run next. If a set of production coefficients is available in the production coefficients file it is used instead of the values entered here.)

DATA MODULE 8 - HYDRO GENERATING STATIONS (page 5):

20. Operating cost in $/GWh (Incremental cost of running the hydro station. It is equal to the amount the utility has to pay for the use of water and may include operation and maintenance cost. S values, one for each LDC strip, have to be entered T times, once for each time step. If this is constant, keyword ALL followed by the cost can be used.)

21. Spill cost in $/KCFS-DAYS (Penalty for spillage, that can be used to affect the linear programming algorithm. Normally, this item is zero as the LP knows that spillage is not beneficial. However, if there is a preference in where water is spilled a value can be given to reflect the preference. T values, one for each time step, are required.)

22. Powerhouse minimum flow slope in KCFS/KCFS-DAYS

23. Powerhouse minimum flow intercept in KCFS

24. Powerhouse maximum flow slope in KCFS/KCFS-DAYS

25. Powerhouse maximum flow intercept in KCFS

26. Spill minimum flow slope in KCFS/KCFS-DAYS

27. Spill minimum flow intercept in KCFS

28. Spill maximum flow slope in KCFS/KCFS-DAYS

29. Spill maximum flow intercept in KCFS (22-29 are used to constrain the minimum and maximum outflow through the powerhouse and spillway as a function of the upstream lake storage. Intercepts for all but the first segment are zero as the segments of the flow-storage curve are non-cumulative. All the items are entered M times, once for each segment of the upstream lake, and that is repeated T times, to cover all the time steps. If the discharge limit is constant the slope of the storage discharge relationship is zero.)

DATA MODULE 8 - HYDRO GENERATING STATIONS (page 6):

30. Powerhouse efficiency curve number of data pairs N1

31. Data pairs (KCFS, decimal %)

(These data define the efficiency curve. N1 data pairs should follow item 30)

32. Number of generating station open water tailrace curves N2

33. Number of data pairs per curve N3

34. Data pairs: flow in KCFS, and elevation in FEET (N3 points define each curve, so N3 data pairs have to be specified N2 times)

35. Number of generating station ice cover tailrace curves M2

36. Number of data pairs per curve M3

37. Data pairs: flow in KCFS, and elevation in FEET (M3 points define each curve, so M3 data pairs have to be specified M2 times)

* The tailwater levels at a hydro plant can be a function of flow, and the downstream lake level. The tailwater curve may also depend upon whether there is ice causing additional backwater. The tailwater curves are used in the calculation of the production coefficients. Items 23 to 25 describe the open water curves while items 26 to 28 describe the ice cover curves. For a particular downstream lake level the data pairs describing the curve to the required accuracy are needed. Items 23 and 26 specify the number of downstream lake level curves modelled.

DATA MODULE 9 - CONTROL STRUCTURES (page 1):

This module has to be repeated for each control structure in the system. Each control structure is described as follows (explanations are given in brackets):

1. Name (The structure name can consists only of alphanumeric characters and underscores. No blanks are allowed.)
2. To be printed (Enter flag YES or NO)
3. Upstream lake (This is the name of the lake that the control structure is impounding. The corresponding lake data module (MODULE 7) under that name has to be present in the data set.)
4. Downstream lake (This is the name of the lake into which the control structure discharges. If the control structure discharge leaves the network, a name for the destination, and not a lake in the network, should be given.)
5. Spill cost in $/KCFS-DAYS (T values are expected, one for each time step. This is a penalty that can be used to reflect loss of energy to the system. Normally, no spillage costs need be assigned as the linear programming algorithm knows that spillage is not beneficial. However, if there is a preference in where water is spilled a value can be given to reflect this preference. If a lake has more than one outlet, a larger spill cost can be assigned to the the outlet where spillage results in a larger energy loss for the system.)

DATA MODULE 9 - CONTROL STRUCTURES (page 2):

6. Minimum flow slope in KCFS/KCFS-DAYS

7. Minimum flow intercept in KCFS

8. Maximum flow slope in KCFS/KCFS-DAYS

9. Maximum flow intercept in KCFS

(All these data items can be used to constrain the minimum or maximum outflow through the control structure as a function of the upstream lake storage. M values, one corresponding to each segment of the upstream lake, have to be entered T times, once for each time step. This means that each of the items 6 - 9 has to be entered TxM times. The intercepts for all but the first segment of the upstream lake should be equal to zero. Other intercepts are equal to zero because the segments of the flow-storage curve are non-cumulative. If all items 6-9 are made equal to zero the flow is unconstrained. If discharge is predetermined then the rating curve is overwritten with that discharge.

DATA MODULE 10 - NATURAL LAKE OUTLETS (page 1):

This module has to be repeated for each natural lake outlet in the system. This data may also be used to describe a fixed crest weir. Each outlet is described as follows:

1. Name (Only alphanumeric characters and underscores can be used. No blanks are allowed)
2. To be printed (Enter YES or NO flag)
3. Upstream lake (This is the name of the lake that is controlled by the natural outlet. The corresponding data module 7 (lake data) under that lake name has to be present in the data set.)
4. Downstream lake (This is the name of the lake into which the natural outlet discharges. If the natural outlet discharge leaves the network, a name for the destination that is not part of the network should be given.)
5. Flow/storage slope in KCFS/KCFS-DAYS
6. Flow/storage intercept in KCFS

(Items 5 and 6 are used to define the natural outflow rating curve as a function of storage. Each item should be specified M times, once for each segment of the upstream lake, and all that should be repeated T times, once for each time step. That means that item 5 has to be entered TxM times, and then the same is repeated for item 6. The intercepts for all but the first segment of the flow/storage curve are equal to zero because these segments are non-cumulative.)

DATA MODULE 11 - LOCAL INFLOWS (page 1):

This section must be repeated for each local inflow. Each local inflow is described as follows:

1. Name (Use only alphanumeric characters and underscores. No blanks are allowed.)

2. To be printed? (Enter YES or NO flag.)

3. Flows into (Enter a lake name. There must be a corresponding module 7 with the lake data under that lake name in the data set.)

4. Flow in KCFS (Enter T flows, one for each time step.)

DATA MODULE 11 - LOCAL INFLOWS (page 1):

An inflow time series of a specified low probability (for example 5%) is required. Note that this is the cumulative probability over the entire time horizon, and that the time step inflows have to be adjusted to that!

DATA MODULE 11 - LOCAL INFLOWS (page 1):

The likeliest inflow scenario is required

DATA MODULE 11 - LOCAL INFLOWS (page 1):

The inflow scenario that corresponds to the operational context of the system that is examined should be entered!

DATA MODULE 12 - GENERATING SOURCES (page 1):

This module must be repeated for each generation source in the network. A generation source is any energy source, either hydraulic or thermal, that is isolated from the hydraulic system. Each generation source is described as follows:

1. Name of the source (Use only alphanumeric characters and underscores. Blanks are not allowed.)
2. To be printed? (Enter flag YES or NO)
3. Shaping constraint - keyword (Shaping of the generation from the source is controlled by this keyword:
- When UNSHAPED, no shaping constraint is written.
- When SHAPED a shaping constraint is written, which ensures the capacity of the source utilized decreases moving from on peak to off peak loads.
- When BASE_LOADED the generation is forced to a uniform output i.e. no cycling of generation will be possible. The facility of shaping generation within a time step is subject to the status of the shaping flag for each time step set in DATA MODULE 5)
4. Number of units - U (Enter integer number of the plant units.)
5. Unit rated capacity in MW's (This is the maximum capacity under normal operating conditions. U numbers (item 4) have to be entered, one for each unit.)

DATA MODULE 12 - GENERATION SOURCES (page 2):

6. Scheduled maintenance - in days (This is the length of time that the unit will be out of operation. It is assumed that a crew will be working on the unit for the same period of time. U numbers (item 4) have to be entered, one for each unit.)

7. Number of maintenance timesteps, 8. Maintenance time step, and

9. Maintenance preference (The maintenance for a plant can be constrained to occur in specific time steps, but not unit by unit. For those time steps in which maintenance may occur a preference could be given, with 1.0 being the highest priority. Any relative scheme can be used, provided that it is commensurate among all plants. Items 8 and 9 have to be repeated for each maintenance timestep.)

10. Pre-scheduled maintenance - in days (The maintenance for units may be pre-scheduled. The number of days is assigned for each unit, which means that U numbers have to be entered for the first time step, and that the same has to be repeated for all other time steps.)

11. Maximum maintenance outage in MW (This is the upper limit on the capacity that maintenance can displace. T numbers have to be specified, one for each time step.)

12. Minimum capacity in MW (This is the minimum load from the generation source. T numbers have to be entered, one for each time step.)

13. Capacity derate in MW (This item has to be entered for each strip of the load duration curve. A sequence of S numbers, one for each strip, has to be repeated for each time step, which means T times.)

DATA MODULE 12 - GENERATION SOURCES (page 3):

14. Forced outage rate (This is the number of forced outage hours divided by the sum of the forced outage hours and the running hours. In effect, this item derates the energy generation ability of the plant. T values are expected, one for each time step.)

15. Forced maintenance days (This is the number of days a maintenance crew must allocate to forced maintenance. This reduces the time the crew can work on scheduled maintenance during the maintenance period.)

16. Station service load in MW/MW (This is the portion of the source's generation that is required for meeting the plant's own loads i.e. grinder, conveyors, cooling water pumps etc. Only the net output of the generating source, equal to the station output minus station service, is available to meet the load. This item has to be entered T times, once for each time step.)

17. Cooling water source (Each generation source must be associated with a cooling water source. This source can be the name of one of the following parts of the hydraulic system:

-a) a hydro generation station
-b) a control structure
-c) a local inflow.

If the cooling water source is not in the hydraulic network, a special, local inflow can be created. This inflow does not have to be connected to the rest of the network.)

DATA MODULE 12 - GENERATION SOURCES (page 4):

18. Maximum load slope in MW/deg.F/KCFS-DAY (T values are expected, one for each time step.)

19. Maximum load intercept in MW (T values are expected, one for each time step.)

20. Maximum downstream temperature in degrees F (T values are expected, one for each time step)

21. Maximum change in temperature in degrees F (T values are expected, one for each time step)

* The waste heat constraint will limit plant output when the temperature rise or the maximum temperature of the fully mixed water in the cooling water source downstream from the plant exceeds the limits given in the data. If both a maximum temperature change and a maximum temperature limit are given EMMA uses the most constraining condition to determine the limit. If the maximum temperature change is zero this limit is ignored and the maximum temperature limit will govern. *

DATA MODULE 12 - GENERATION SOURCES (page 5):

22. Maximum load slope in MW/degF (T values are expected, one for each time step)

23. Maximum load intercept in MW (T values are expected, one for each time step)

* The temperature limit constraint reflects the loss in plant capacity due to the cooling water. As cooling water temperatures rise higher cooling water velocities through the condensers are required to carry away the waste heat. At higher velocities erosion occurs in the condensor so to limit the erosion a maximum velocity is placed on the flow. This constraint is a function of the cooling water temperature and not of the cooling water resource discharge. Therefore, the constraint becomes an upper bound on capacity rather than a constraint. *

24. Cooling water source temperature in degrees F (A time series forecast of water temperatures at the prescribed probabilities is provided. This is used for the waste heat and temperature limit constraint. T values are expected, one for each time step)

25. Operating cost in $/GWh (This is the incremental cost of running the station. S values, one for each load duration curve strip, have to be entered T times, once for each time step. If the cost is constant, the keyword ALL followed by the cost can be used.)

DATA MODULE 13 - TIELINES (page 1):

This module must be repeated for each tieline in the system. Each tieline connects the system to an external energy source and/or energy demand. Each tieline is described as follows:

1. Name (Use only alphanumeric characters and underscores. No blanks are allowed.)
2. To be printed? (Enter YES or NO flag)
3. Firm imports in MW (EMMA permits a firm import or export of energy as long as these two do not occur simultaneously. A firm import contract has effect of reducing the system load and occupying a portion of the tieline. This contract has to be specified for each strip of the load duration curve, which means S times, and that has to be repeated for T time steps in EMMA. TxS values are therefore required.)
4. Cost of firm imports in $/GWh (The cost of the firm import may be specified for each strip of the load duration curve, which means S times, and that has to be repeated for T time steps.
5. Import overall minimum in GWh (An overall minimum energy import for all time steps may be specified. This amount of energy will be allocated over all strips of all load duration curves to produce the best overall net revenue.)

DATA MODULE 13 - TIELINES (page 2):

6. Cost of import overall minimum $/GWh (Cost of the overall minimum import energy may be specified. The distribution of this energy will depend on this value as it will compete with the interruptible energy purchases.)
7. Maximum import load in MW (A tieline may have different maximum import load capabilities depending upon electrical operating constraints and transmission outages. These capabilities are defined at the boundaries of the utility's jurisdiction.)
8 Number of import segments N, and
9. Import segment capacity in MW (Tieline capacity may be divided into segments to permit the representation of non linear import cost function as piece wise linear. Segmentation of the capacity occurs after the capacity required for firm loads and overall minimum loads are accounted for. Figure A demonstrates the composition of a tieline. The number of segments depends upon how close a representation of the cost curve is desired. Item 9 has to be specified for each tieline segment, which means N times, for each of S strips of the LDC. All that has to be repeated for each time step, which means that total number of entries for item 9 is TxSxN.

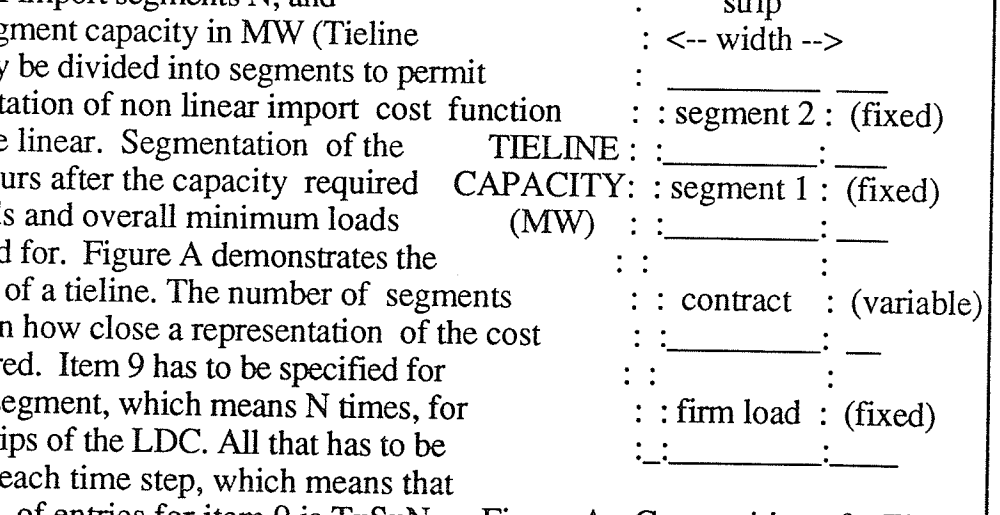


Figure A - Composition of a Tieline

DATA MODULE 13 - TIELINES (page 3):

10. Import segment cost in $/GWh (Figure B demonstrates the segmentation of the cost function. Item 10 has to be specified for each tieline segment, which means N times, for each of S strips of the load duration curve. All that has to be repeated for each time step, which means that the total number of inputs is TxSxN)

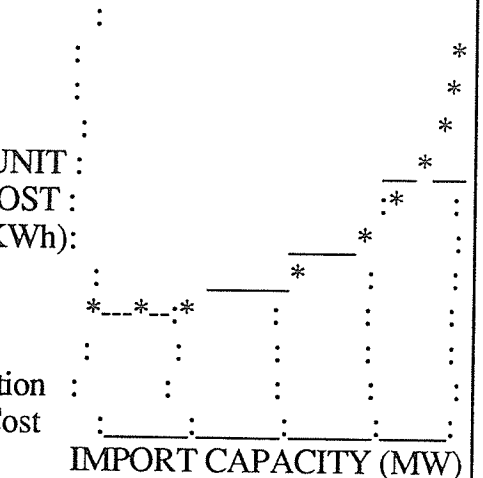


Figure B - Segmentation Import Cost Function

11. Import losses MW/MW (Average import tieline losses can be accounted for with this item. Import losses are expressed as a fraction of the import load at its source, i.e. the border between the importing and exporting systems.)

12. Firm exports in MW (EMMA permits a firm import or export of energy as long as these two do not occur simultaneously. A firm export contract has effect of increasing the system load and occupying a portion of the tieline. This contract has to be specified for each strip of the load duration curve, which means S times, and that has to be repeated for T time steps in EMMA. TxS values are therefore required.)

DATA MODULE 13 - TIELINES (page 4):

13. Value of firm exports in $/GWh (The value of the firm export may be specified for each strip of the load duration curve, which means S times, and that has to be repeated for T time steps.)

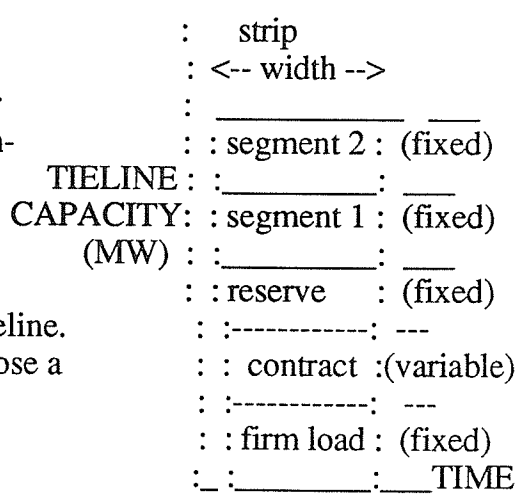
14. Export overall minimum in GWh (An overall minimum export for all time steps may be specified. This amount of energy will be allocated over all strips of all load duration curves to produce the best overall net revenue.)

15. Value of export overall minimum $/GWh (The value of the overall minimum export energy may be specified. The distribution of this energy will depend on this value as it will compete with the interruptible energy sales.)

16. Maximum export load in MW (A tieline may have different maximum export load capabilities depending upon electrical operating constraints and transmission outages. These capabilities are defined at the boundaries of the utility's jurisdiction.)

DATA MODULE 13 - TIELINES (page 5):

17. Number of export segments N, and
18. Export segment capacity in MW
(Tieline capacity may be divided into a number of segments to permit the representation of non-linear export value function as piece wise linear. Segmentation of the capacity occurs after the capacity required for firm loads and overall minimum loads are accounted for. Figure A demonstrates the composition of a tieline. The number of segments depends upon how close a representation of the value curve is desired. Item 9 has to be specified for each tieline segment, which means N times, for each of S strips of the LDC. All that has to be repeated for each time step,which means that total number of entries for item 9 is TxSxN.



FigureA-Composition of a Tieline

DATA MODULE 13 - TIELINES (page 6):

19. Export segment value in $/GWh
(Figure B demonstrates the segmentation of the value function. Item 10 has to be specified for each tieline segment, which means N times, for each of S strips of the load duration curve. All that has to be repeated for each time step, which means that the total number of inputs is TxSxN)

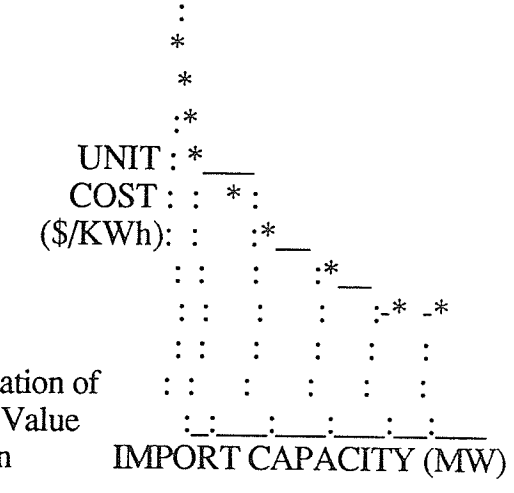


Figure B - Segmentation of Export Value Function

20. Export losses MW/MW (Average export tieline losses can be accounted for with this item. Export losses are expressed as a fraction of the loading at the source.)

DATA MODULE 14 - GENERATION STRIP GROUPS (page 1): This module enables grouping of energy sources by strip of LDC. Grouping is used to define a minimum energy output from the membership of a group over a strip of the LDC. For each generation strip group, the following data are required:

1. Generation group name (Only alphanumerics & underscores, no blanks.)
2. To be printed? (Enter YES or NO flag.)
3. Minimum generation slope in MW/MW, and
4. Minimum generation intercept in MW (3 and 4 define the function of the system load for all strips in each time step. T values, one for each time step, are required for both. If the constraint is independent of the load, the slope is zero.)
5. Number of the generation strip group members N (This defines how many members are involved in the group.)
6. Name (This defines the name of the generating station, generating source, or tieline. It must be one from the electrical network)
7. Type of energy source or sink (Four keywords can be used here:

GEN_STATION - for hydro generating stations,
GEN_SOURCE - for generation sources,
TIELINE_EXPORT - for tieline exports, and
TIELINE_IMPORT - for tieline imports.

8. Membership (This is a fraction and it describes what contribution the member will make to meeting the constraint by timestep. This may vary by time step and strip, and has to be entered T times. 6,7 ,8 have to be repeated for all N (item 5) group members.)

DATA MODULE 14 - GENERATION STRIP GROUPS (page 2):

Note that maximum generation can also be imposed on the group. This is accomplished by entering negative values for items 3,4 and 8!

DATA MODULE 15 - GENERATION TIME GROUPS (page 1): This module enables the grouping of energy sources in all strips of the load duration curve by time step. The grouping is used to define a minimum energy output from the membership of the group over a time step. For each generation time group, the following data items are required:

1. Generation group name (Alphanumerics and underscores, no blanks.)
2. To be printed? (Enter YES or NO flag.)
3. Minimum generation slope in MW/MW, and
4. Minimum generation intercept in MW (3 and 4 define the function of the system load for all strips and time steps. T values, are required for 3 and 4)
5. Number of the generation time group members N (This defines how many members are involved in the group.)
6. Name (This defines the name of the generating station, generating source, or tieline. It must be one from the electrical network)
7. Type of energy source or sink (Four keywords that can be used here:

GEN_STATION - for hydro generating stations,
GEN_SOURCE - for generation sources,
TIELINE_EXPORT - for tieline exports, and
TIELINE_IMPORT - for tieline imports.

8. Membership (This is a fraction and it describes what contribution the member will make to meeting the constraint by timestep. This may vary by time step, and has to be entered T times. Items 6,7 and 8 have to be repeated for all N (item 5) group members.)

DATA MODULE 15 - GENERATION TIME GROUPS (page 2):

Note that maximum generation can also be imposed on the group. This is accomplished by entering negative values for items 3,4 and 8!

DATA MODULE 16 - GENERATION PERIOD GROUPS (page 1) This module enables the grouping of energy sources in all strips of the load duration curve and all time steps by period. The period is over all time steps. The grouping is used to define a minimum energy output from the membership of the group over the period. For each generation period group, the following data items are required:

1. Generation group name (Alphanumerics and underscores, no blanks)
2. To be printed? (Enter YES or NO flag.)
3. Minimum generation slope in MW/MW, and
4. Minimum generation intercept in MW (The minimum generation of the group is a linear function of the system load. 3 and 4 describe this relationship.)
5. Number of the generation period group members N (This defines how many members are involved in the group.)
6. Name (This defines the name of the generating station, generating source, or tieline. It must be one from the electrical network)
7. Type of energy source or sink (Four keywords that can be used here:

GEN_STATION - for hydro generating stations,
GEN_SOURCE - for generation sources,
TIELINE_EXPORT - for tieline exports, and
TIELINE_IMPORT - for tieline imports.

8. Membership (This is a fraction and it describes what contribution the member will make to meeting the constraint by timestep. This may vary by time step, and has to be entered T times. Items 6,7 and 8 have to be repeated for all N (item 5) group members.)

DATA MODULE 16 - GENERATION PERIOD GROUPS (page 2):

Note that maximum generation can also be imposed on the group. This is accomplished by entering negative numbers for items 3,4 and 8!

DATA MODULE 17 - MAINTENANCE CREWS (page 1):

This module must be repeated for each maintenance crew. Each maintenance crew is described as follows (explanations of data items are given in brackets):

1. Name (Alphanumeric characters and underscores. No blanks.)
2. To be printed? (Enter YES or NO flag.)
3. Maximum time step availability in days (This is the actual number of days that the crew would normally have units out of service during a time step. T values, one for each time step are required.)
4. Maximum period availability in days (This the actual number of days that the crew would normally have units out of service during the period defined by items 5 & 6.)
5. Number of time steps in the period (This defines how many time steps are there in the period that needs to be defined for maintenance purposes.)
6. Period timesteps names (List the names of all the time steps that belong to the period. Total number of entries has to be equal to the number entered for 5.)
7. Number of assigned plants (This is the total number of hydro and thermal stations the crew is assigned to maintain.)
8. Plant name (List all the hydro and thermal stations that the crew is assigned to maintain. Total number of entries must be equal to the number entered under item 7. The corresponding modules for the hydro generating stations data or generating source data under all listed names must be present in the data set.)

APPENDIX B

SAMPLE LISTING OF EXPERT SYSTEM MODULE

```
MODULE: emma

INTENT: "Give advice | to EMMA users|"
    CHILD: intro_emma
    CHILD: firmness
    CHILD: sequence
    LOCAL: string {sys_name sys_title sys_help}
LOCAL: string {motivation step first drought cont}


      STATE: begin ("e.exe" -> sys_name;
        "EMMA advice" -> sys_title;
        "general.hlp" -> sys_help;
        intro_emma, context)


      STATE: context
        IF (ask "There are three basic motivations for the use of EMMA - select one:

1 - ANNUAL BUDGET preparation and updates,

2 - preparation of OPERATIONAL PLAN of releases, thermal and hydro
generation, maintenance, exports, and imports.

3 - LONG TERM STUDY of the system ( benefits from additional capacities, some
operational condition or a change in the system). "

"budget,operational_plan,long_term_study") IS
            "budget" : ( "budget" -> motivation, budget_how )
            "operational_plan" : ( "operational_plan" -> motivation,
                                   operational_how )
            ELSE ( "long_term_study" -> motivation, study_how )


      STATE: budget_how
IF (ask "The following 5 steps are necessary in order to obtain results that can be
used in preparation or updates of the annual budget. Select one for further
explanation!

1_FIRMNESS_TEST, 2_SEVERE_CASE, 3_EXPECTED_CASE,
4_SEVERE_CASE_RE_RUNS, 5_CONVERGENCE_TEST, end_consultation "
"1_FIRMNESS_TEST,2_SEVERE_CASE,3_EXPECTED_CASE,4_SEVERE_C
ASE_RE_RUNS,5_CONVERGENCE_TEST,end_consultation")
IS
            "1_FIRMNESS_TEST" : ( firmness, budget_how )
```

"2_SEVERE_CASE" : ("severe" -> step;
reads " EMMA SEVERE CASE should be run first. This means assuming a water supply with a low probability (say 95% probability of exceedence) and assuming early freeze-up date. This run will formulate a release and energy production plan that will satisfy system demand in case when operating conditions are severe. Keeping firmness test reserves ensures that the system will be capable of coping with contingencies if they should occur on top of the described severe conditions.
PRESS RETURN TO CONTINUE" -> cont;
sequence motivation step, budget_how)
"3_EXPECTED_CASE" : ("expected" -> step; reads " EMMA EXPECTED CASE should be run next. This means taking the water supply with 50% probability of exceedence and assuming expected freeze-up date. This run will formulate a release and energy production plan that will satisfy demand in case of the likeliest flow and loads scenario. Note that hydro generation will be larger in this than in the severe case (more water available) and expenses for non hydraulic energy supplies will be smaller.

However, neither operational plan nor annual budget can be prepared based on results of this run because the system should be able to perform if the severe scenario should occur. What can be done, is increase hydro generation and releases in the immediate future for the severe scenario and compensate that with expensive thermal generation in later time steps. This way, the system will perform economically in the immediate future, and in later time steps it has a small probability (probability that the severe scenario really occurs) of paying penalty for this. In most cases (95%) flows will be larger than predicted in the severe case, and the system will avoid penalty.

PRESS RETURN TO CONTINUE" -> cont;
sequence motivation step, budget_how)
"4_SEVERE_CASE_RE_RUNS" : ("re_runs" -> step; reads " SEVERE CASE should be re-run iteratively, gradually increasing lower bounds on hydro generation for the first three or four time steps (thus forcing it to increase). This however, must not exceed the corresponding values for hydro generation in the expected case. This incremental increase should be stopped when EMMA starts drawing up on the system's reserves determined in the FIRMNESS TEST.

PRESS RETURN TO CONTINUE" -> cont;
sequence motivation step, budget_how)
"5_CONVERGENCE_TEST" : (reads " ---------- CONVERGENCE TEST: ----------
Levels in the system's reservoirs and forbays operated according to operational rules should be now re-calculated manually, based on the releases from the latest EMMA run.

EMMA should be re-run with these levels in order to make sure that the operating policy converges.

PRESS RETURN TO CONTINUE" -> cont, budget_how)
ELSE (null, GOAL)

STATE: operational_how
IF (ask "Is the system in a drought?

*System is in drought if reserves are below usual and inflows are not big enough to enable sufficient energy production, thus making it necessary to use water in storage" "yes,no")
IS
"yes" : (null, drought_operation)
ELSE (null, no_drought_operation)

STATE: drought_operation
IF (ask "The following 5 steps are necessary in order to prepare an operational plan of releases and power generation. Select one for further explanation!

1_FIRMNESS_TEST, 2_SEVERE_CASE, 3_EXPECTED_CASE, 4_SEVERE_CASE_RE_RUNS, 5_CONVERGENCE_TEST, 6_end_consultation" "1,2,3,4,5,6")
IS
"1" : (firmness, drought_operation)
"2" : ("severe" -> step;
reads " EMMA SEVERE CASE should be run first. This means assuming a low water supply (say 95% of exceedence) and assuming early freeze-up date. This run will formulate a release and energy production plan that will satisfy system demand in case when operating conditions are severe. Keeping firmness test reserves ensures that the system will be capable of coping with contingencies if they should occur on top of the described severe conditions.

PRESS RETURN TO CONTINUE" -> cont;
sequence motivation step, drought_operation)
"3" : ("expected" -> step; reads " EMMA EXPECTED CASE should be run next. This means taking the water supply with 50% probabilityof exceedence and assuming expected freeze-up date. This run will formulate a release and energy production plan that will satisfy demand in case of the likeliest flow and loads scenario. Note that hydro generation will be larger in this than in the severe case (more water available) and expenses for non hydraulic energy supplies will be smaller.

However, neither operational plan nor annual budget can be prepared based on results of this run because the system should be able to perform if the severe scenario should occur. What can be done, is increase hydro generation and releases in the immediate future for the severe scenario and compensate that with expensive thermal generation in later time steps. This way, the system will perform economically in the immediate future, and in later time steps it has a small probability (probability that the severe scenario really occurs) of paying penalty for this. In most cases (95%) flows will be larger than predicted in the severe case, and the system will avoid penalty.

PRESS RETURN TO CONTINUE" -> cont;
sequence motivation step, drought_operation)
"4" : ("re_runs" -> step; reads " SEVERE CASE should be re-run iteratively, gradually increasing lower bounds on hydro generation for the first three or four time steps (thus forcing it to increase). This however, must not exceed the corresponding values for hydro generation in the expected case. This incremental increase should be stopped when EMMA starts drawing up on the system's reserves determined in the FIRMNESS TEST.

PRESS RETURN TO CONTINUE" -> cont; sequence motivation step, drought_operation)
"5" : (reads " ---------- ---------- CONVERGENCE TEST: ---------- ----------
Levels in the system's reservoirs and forbays operated according to operational rules should be now re-calculated manually, based on the releases from the latest EMMA run.

EMMA should be re-run with these levels in order to make sure that the operating policy converges.

PRESS RETURN TO CONTINUE" -> cont, drought_operation)
ELSE (null, GOAL)

STATE: no_drought_operation
IF (ask "The following 3 steps are necessary in order to prepare an operational plan of releases and power generation. Select one for further explanation!

1_FIRMNESS_TEST, 2_EXPECTED_CASE, 3_CONVERGENCE_TEST, 4_end_consultation" "1,2,3,4")
IS
"1" : (firmness, no_drought_operation)
"2" : ("expected_n" -> step;
reads "

EMMA EXPECTED CASE should be run. This means taking the flows with 50% probability of exceedence and assuming expected freeze-up date. This run will formulate a release and energy production plan that will satisfy demand in case of the likeliest flow and loads scenario.

PRESS RETURN TO CONTINUE" -> cont; sequence motivation step, no_drought_operation)
"3" : (reads " ---------- CONVERGENCE TEST: ----------
Levels in the system's reservoirs and forbays operated according to operational rules should be now re-calculated manually, based on the releases from the latest EMMA run.

EMMA should be re-run with these levels in order to make sure that the operating policy converges.

PRESS RETURN TO CONTINUE" -> cont, no_drought_operation)
ELSE (null, GOAL)

STATE: study_how
IF (ask "The following 5 steps are necessary in order to obtain results that can be used in long term studies of the system performance. Select one for further explanation!

1_FIRMNESS_TEST, 2_EXISTING_SYSTEM, 3_CONVERGENCE_TEST, 4_CHANGED_SYSTEM, 5_CONVERGENCE_TEST, end_consultation" "1,2,3,4,5,6")
IS
"1" : (firmness, study_how)
"2" : ("existing" -> step;
reads " System without the examined change is first entered to EMMA and different sequences of flows (95%, 75%, 50%, 25%, and 5% probability of exceedence) or scenarios of interest are examined in order to evaluate performance of the system.

PRESS RETURN TO CONTINUE" -> cont; sequence motivation step, study_how)
"3" : (reads " ---------- CONVERGENCE TEST: ----------
Levels in the system's reservoirs and forbays operated according to operational rules should be now re-calculated manually, based on the releases from the latest EMMA run.

EMMA should be re-run with these levels in order to make sure that the operating policy converges.

PRESS RETURN TO CONTINUE" -> cont, study_how)
"4" : ("changed" -> step; reads " System with the examined change built in is entered to EMMA and different sequences of flows (5%, 25%, 50%, 75%, 95% probability of exceedence.) or scenarios are repeated in order to evaluate the effect of the examined change in the system. Note that this procedure gives a fair comparison of system performance with and without the examined change, because it takes its optimal responce in both cases.

PRESS RETURN TO CONTINUE" -> cont; sequence motivation step; reads " Note that the resulting energy productions, reservoir levels, and other physical data obtained for the utility with and without the examined change should be used for further evaluation of benefits and not values of EMMA objective function itself (EMMA objective function gives only relative dollar values that can be compared only within one EMMA run).

PRESS RETURN TO CONTINUE" -> cont, study_how)
"5" : (reads " ---------- CONVERGENCE TEST: ----------
Levels in the system's reservoirs and forbays operated according to operational rules should be now re-calculated manually, based on the releases from the latest EMMA run.

EMMA should be re-run with these levels in order to make sure that the operating policy converges.

PRESS RETURN TO CONTINUE" -> cont, study_how)
ELSE (null, GOAL)

GOAL OF emma

APPENDIX C
LISTING OF GRAPHICS PROGRAM

```
#nolist
#include <stdio.h>
 #include <math.h>
#include "/sys/ins/base.ins.c"
 #include "/sys/ins/gmr.ins.c"
#include "/sys/ins/pfm.ins.c"
#list

/* declaring necessary globals */
short file_id, viewport_id1, viewport_id2;
gm_$segment_id_t segment_id, s1, s2, s3, s4, s5, s6;

main()

{ extern int select_from_menu();

int choice;
short high_plane , font_family_id;
gm_$boundsreal_t bounds;
status_$t st;
gm_$point16_t bitmap_size ;
gm_$color_entry_t color_array[8];

      high_plane = 8; bitmap_size.x = 1024; bitmap_size.y = 1024;

/* Initialize 2D GMR. */

      gm_$init( gm_$direct,
                  stream_$stdout, bitmap_size, high_plane, st);

/* Define colours */

   color_array[0].red = 1.0;
   color_array[0].green = 1.0;
   color_array[0].blue = 0.0;

      gm_$display_set_color_map((long)1,(short)1,color_array,st);

   color_array[0].red =1.0;  color_array[0].green =0.2;  color_array[0].blue =0.3;
   /* red 16 */
   color_array[1].red =0.3;  color_array[1].green =1.0; color_array[1].blue =0.3;
   /* green 17 */
   color_array[2].red =0.3;  color_array[2].green =0.2; color_array[2].blue =1.0;
    /* blue 18 */
   color_array[3].red =0.3;  color_array[3].green =1.0; color_array[3].blue =1.0;
   /* b-g 19 */
```

```
    color_array[4].red =1.0;  color_array[4].green =1.0; color_array[4].blue 0.0;
    /* yellow 20 */
    color_array[5].red =1.0;  color_array[5].green =0.4; color_array[5].blue =1.0;
    /* purple 21 */
    color_array[6].red =1.0;  color_array[6].green =0.6;  color_array[6].blue =0.0;
    /* orange 22 */
    color_array[7].red =1.0;  color_array[7].green =1.0; color_array[7].blue =1.0;
     /* white 23 */

       gm_$display_set_color_map((long)16,(short)8,color_array,st);

/* Create/update and name metafile. */

       gm_$file_create("EMMA_ES_gmr",
                                (short)11, gm_$write, gm_$1w, file_id, st);
       if( st.all == status_$ok)  {
 /* Get the right fonts (necessary for writing text). */
       gm_$font_family_include("FONTS",(short)5,gm_$pixel,font_family_id,st);
                get_it_started();
                        /* make all necessary segments */
                        gm_$file_close(true,st);
                        };

       gm_$file_open("EMMA_ES_gmr",

       identify_segments();

/* Coerce REAL data to INTEGER16. */
       gm_$data_coerce_set_real(gm_$16,st);


/* Move viewport 1 to the left for menu use */
         bounds.xmin =    0.0;
         bounds.xmax =    0.24;
         bounds.ymin =    0.0;
         bounds.ymax =    0.75;
       gm_$viewport_set_bounds (bounds,
                                           st);

/* Create viewport where everything will be going on */
         bounds.xmin =    0.244;
         bounds.xmax =    1.0;
         bounds.ymin =    0.0;
         bounds.ymax =    1.0;
       gm_$viewport_create(bounds,
                                  viewport_id1, st);
```

```
/* Create viewport for messages */
        bounds.xmin =    0.0;
        bounds.xmax =    0.24;
        bounds.ymin =    0.755;
        bounds.ymax =    1.0;
      gm_$viewport_create(bounds,
                                  viewport_id2, st);

/* Function that draws the scheme of the system */

      select:
            please_draw_scheme();

/* Function that draws the main menu and returns a picked value */

      choice = select_from_menu(1);
      if( choice == 2 ) edit_mode();
      if( choice == 3 ) view_mode();
      if( choice != 1 ) goto select;

/* Close the metafile and terminate the whole thing. */

      gm_$file_close( true,
                              st);

/* Set the cursor to the original color */

      color_array[0].red = 1.0;
      color_array[0].green = 0.0;
      color_array[0].blue = 0.0;

      gm_$display_set_color_map((long)1,(short)1,color_array,st);
      gm_$terminate(st);
}
```

```
/*-------------------------------------------------------------------------*/
add_control()
{
extern gm_$segment_id_t s5;
extern int select_from_menu();
extern int picklake();
extern int get_name();

status_$t                   st;
gm_$point_array16_t         pt;
gm_$point16_t               pt1,pt2;
gm_$pointreal_t             ptrl,position;
short                       radius1, radius2 , n1, delta_y, delta_x, width,
                              lake1_length,  lake2_length;
int                         choice,count,n;
gm_$string_t                str, lake1, lake2;
boolean                     fill;
char                        call_str[32];
double                      fact,dist;

/* Prompt to get station name */

  n= get_name("ENTER THE NAME OF","CONTROLLED OUTLET",call_str);
     if ( n > 32 ) n = 32;
     n1 = (short)n;
     count = 0;
     while( count <= n ){
                                 str[count] = call_str[count];
                                 ++count;};
again1:

     tell_user("PICK UPSTREAM","LAKE");
     please_draw_scheme();
     pick_something(0.5,0.5);
     choice = picklake();

     if( choice == 1 ) {
                                 gm_$inq_circle_16(pt1,radius1,fill,st);
                                 gm_$pick_command(gm_$step,st);
                                 gm_$inq_tag(lake1,lake1_length,st);
                                 gm_$segment_close(false,st);
     } else
                                { choice = select_from_menu(6);
                                  if (choice == 1) return;
                                  goto again1;
                                };
```

```
again2:

    tell_user("PICK DOWNSTREAM","LAKE");
    please_draw_scheme();
    pick_something(0.5,0.5);
    choice = picklake();

    if( choice == 1 ) {
                        gm_$inq_circle_16(pt2,radius2,fill,st);
                        gm_$pick_command(gm_$step,st);
                        gm_$inq_tag(lake2,lake2_length,st);
                        gm_$segment_close(false,st);
    } else
                       { choice = select_from_menu(6);
                         if (choice == 1) return; goto again2;
                       };

/* Construct the rectangle */
    width = 5;
    delta_x = pt2.x - pt1.x;
    delta_y = pt2.y - pt1.y;
    dist = delta_x * delta_x + delta_y * delta_y;
    fact = radius1 / sqrt( dist );

    gm_$segment_open(s5,st);
    gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
    check(st);
    gm_$fill_value((long)18,st);
    gm_$tag(lake1,lake1_length,st);
    gm_$tag(lake2,lake2_length,st);

size_choice:

    pt[0].x = pt1.x + delta_y/width; pt[0].y = pt1.y - delta_x/width;
    pt[1].x = pt1.x - delta_y/width; pt[1].y = pt1.y + delta_x/width;
    pt[2].x = pt2.x - delta_y/width; pt[2].y = pt2.y + delta_x/width;
    pt[3].x = pt2.x + delta_y/width; pt[3].y = pt2.y - delta_x/width;

    gm_$polyline_2d16((short)4,pt,true,true,st);
    check(st);
    gm_$segment_close(true,st);
    please_draw_scheme();

/* Select the station size */
     choice = select_from_menu(5);
     gm_$segment_open(s5,st);
```

```
        if(choice == 3) {
                gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
                gm_$fill_value((long)23,st);

                width = width * 0.9;
                pt1.x = pt1.x + delta_x * fact;
                pt1.y = pt1.y + delta_y * fact;
                pt[0].x = pt1.x + delta_y/width;
                pt[0].y = pt1.y - delta_x/width;
                pt[1].x = pt1.x - delta_y/width;
                pt[1].y = pt1.y + delta_x/width;
                pt[2].x = pt[1].x + delta_x / (width * 2.5);
                pt[2].y = pt[1].y + delta_y / (width * 2.5);
                pt[3].x = pt[0].x + delta_x / (width * 2.5);
                pt[3].y = pt[0].y + delta_y / (width * 2.5);

                gm_$polyline_2d16((short)4,pt,true,true,st);
                gm_$tag(str,n1,st);
                gm_$segment_close(true,st);
                please_draw_scheme();}
        else {
                if( choice == 1 )
                width = width * 0.9 ;
                if( choice == 2 ) width = width * 1.1 + 1 ;
                gm_$modelcmd_set_mode(gm_$modelcmd_replace,st);
                goto size_choice; };
}
```

```
/*-------------------------------------------------------------------------*/
/* Edit the scheme */
add_element()
{
extern int select_from_menu();
int choice;

/* Select from edit menu */
again: choice = select_from_menu(2);

/* Go to the appropriate module */
        if (choice == 2) add_lake();
        if (choice == 3) add_inflow();
        if (choice == 4) add_hydro();
        if (choice == 5) add_outlet();
        if (choice == 6) add_control();
        if (choice == 7) add_thermal();
        if (choice != 1) goto again;
}
```

```
add_hydro()
{
extern gm_$segment_id_t s2;
extern int select_from_menu();
extern int picklake();
extern int get_name();

/* declare variables */
status_$t                st;
gm_$point_array16_t      pt;
gm_$point16_t            pt1,pt2;
gm_$pointreal_t          ptrl,position;
short                    radius1, radius2 , n1, delta_y, delta_x, width,
                         lake1_length, lake2_length;
int                      choice,choice1,count,n;
gm_$string_t             str, lake1, lake2;
boolean                  fill;
char                     call_str[32];
double                   fact,dist;


/* Prompt to get station name */
    n= get_name("ENTER THE NAME OF","THE HYDRO STATION",call_str);
    if ( n > 32 ) n = 32;
    n1 = (short)n;
    count = 0;
    while( count <= n ){
                              str[count] = call_str[count];
                              ++count;};
again1:
    tell_user("PICK UPSTREAM","LAKE");
    please_draw_scheme();
    pick_something(0.5,0.5);
    choice = picklake();

    if( choice == 1 ) {
                             gm_$inq_circle_16(pt1,radius1,fill,st);
                             gm_$pick_command(gm_$step,st);
                             gm_$inq_tag(lake1,lake1_length,st);
                             gm_$segment_close(false,st);
    } else
                            { choice = select_from_menu(6);
                              if (choice == 1) return;
                              goto again1;
                            };
```

```
again2:
    tell_user("PICK DOWNSTREAM","LAKE");
    please_draw_scheme();
    pick_something(0.5,0.5);
    choice = picklake();
    if( choice == 1 ) {
                                gm_$inq_circle_16(pt2,radius2,fill,st);
                                gm_$pick_command(gm_$step,st);
                                gm_$inq_tag(lake2,lake2_length,st);
                                gm_$segment_close(false,st);
    } else
                               { choice = select_from_menu(6);
                                 if (choice == 1) return;
                                 goto again2;
                               };

/* Construct the rectangle */
    width = 5;
    delta_x = pt2.x - pt1.x;
    delta_y = pt2.y - pt1.y;
    dist = delta_x * delta_x + delta_y * delta_y;
    fact = radius1 / sqrt( dist );

    gm_$segment_open(s2,st);
    gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
    check(st);
    gm_$fill_value((long)18,st);
    gm_$tag(lake1,lake1_length,st);
    gm_$tag(lake2,lake2_length,st);

size_choice:
    pt[0].x = pt1.x + delta_y/width;
    pt[0].y = pt1.y - delta_x/width;
    pt[1].x = pt1.x - delta_y/width;
    pt[1].y = pt1.y + delta_x/width;
    pt[2].x = pt2.x - delta_y/width;
    pt[2].y = pt2.y + delta_x/width;
    pt[3].x = pt2.x + delta_y/width;
    pt[3].y = pt2.y - delta_x/width;

    gm_$polyline_2d16((short)4,pt,true,true,st);
    check(st);
    gm_$segment_close(true,st);

    please_draw_scheme();
```

```
/* Select the station size */

      choice = select_from_menu(5);
      gm_$segment_open(s2,st);
      if(choice != 3)
                              {
                               if( choice == 1 ) width = width * 0.9 ;
                               if( choice == 2 ) width = width * 1.1 + 1 ;
                               gm_$modelcmd_set_mode(gm_$modelcmd_replace,st);
                               goto size_choice; }
   else                       {
                               gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
                               gm_$fill_value((long)17,st);
                              width = width * 0.9;
                              pt1.x = pt1.x + delta_x * fact;
                              pt1.y = pt1.y + delta_y * fact;
            plant_size:
                              pt[0].x = pt1.x + delta_y/width;
                              pt[0].y = pt1.y - delta_x/width;
                              pt[1].x = pt1.x - delta_y/width;
                              pt[1].y = pt1.y + delta_x/width;
                              pt[2].x = pt[1].x + delta_x / (width * 0.8);
                              pt[2].y = pt[1].y + delta_y / (width * 0.8);
                              pt[3].x = pt[0].x + delta_x / (width * 0.8);
                              pt[3].y = pt[0].y + delta_y / (width * 0.8);

                              gm_$polyline_2d16((short)4,pt,true,true,st);
                              gm_$segment_close(true,st);
                              please_draw_scheme();

                              choice1 = select_from_menu(5);
                              gm_$segment_open(s2,st);
                              if(choice1 != 3) {
                                if (choice1 == 1) width = width * 0.9;
                                if (choice1 == 2) width = width * 1.1 +1;
                                gm_$modelcmd_set_mode(gm_$modelcmd_replace,st)
                                ; goto plant_size;}
                              else          {
                                      gm_$tag(str,n1,st);
                                      gm_$segment_close(true,st);};

                              choice = select_from_menu(7);
```

```
                    if(choice == 1){
                            pt1.x=pt1.x+delta_x/(width*0.5);
                            pt1.y=pt1.y+delta_y/(width*0.5);

        n = get_name("ENTER NAME FOR RUN","OF RIVER
        STATION",call_str);
                            if ( n > 32 ) n = 32;
                            n1 = (short)n;
                            count = 0;
                            while( count <= n ){
                                    str[count] = call_str[count];
                                    ++count;};
                        gm_$segment_open(s2,st);
                        gm_$modelcmd_set_mode(gm_$modelcmd_insert,st)
                        ; goto plant_size;};
};
}
```

```
/*-------------------------------------------------------------------------*/
add_inflow()
{
extern gm_$segment_id_t s3;
extern int select_from_menu();
extern gm_$pointreal_t pick_position();
extern int picklake();
extern int get_name();

/* declare variables */
status_$t               st;
gm_$point_array16_t     pt;
gm_$point16_t           pt1;
gm_$pointreal_t         ptrl,position;
short                   radius, n1, delta_y, delta_x, width,lake_length;
int                     choice,choice1,count,n;
gm_$string_t            str,lake;
boolean                 fill;
char                    call_str[32];

/* Prompt to get inflow name */
    n = get_name("ENTER INFLOW NAME","up to 32 characters",call_str);
    if ( n > 32 ) n = 32;
    n1 = (short)n;
    count = 0;
    while( count <= n ){
                            str[count] = call_str[count];
                            ++count;};

/* Select the inflow the inflow goes to */

again1:
    tell_user("PICK DESTINATION","LAKE FOR THE INFLOW");
    please_draw_scheme();
    pick_something(0.5,0.5);
    choice = picklake();
    if ( choice == 1 ){
                            gm_$inq_circle_16(pt1,radius,fill,st);
                            gm_$pick_command(gm_$step,st);
                            gm_$inq_tag(lake,lake_length,st);
                            gm_$segment_close(false,st);
    } else
                           { choice1 = select_from_menu(6);
                             if (choice1 == 1) return;
                             goto again1;
                           };
```

```
/* Prompt to pick the source position */
     tell_user("CHOOSE THE SOURCE","POSITION");
     position = pick_position();
     pt[0].x = position.x;
     pt[0].y = position.y;

/* Construct the triangle */
     width = 5;
     delta_x = pt[0].x - pt1.x;
     delta_y = pt[0].y - pt1.y;
     gm_$segment_open(s3,st);
     gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
     check(st);
     gm_$tag(lake,lake_length,st);

size_choice:

     pt[1].x = pt1.x + delta_y/width;
     pt[1].y = pt1.y - delta_x/width;
     pt[2].x = pt1.x - delta_y/width;
     pt[2].y = pt1.y + delta_x/width;

     gm_$polyline_2d16((short)3,pt,true,true,st);
     check(st);
     gm_$segment_close(true,st);

     please_draw_scheme();

/* Select the inflow size */
      choice = select_from_menu(5);
      gm_$segment_open(s3,st);

      if(choice == 3) {
            gm_$tag(str,n1,st);
            gm_$segment_close(true,st); }
      else {
            if( choice == 1 )
            width = width * 0.9 ;
            if( choice == 2 ) width = width * 1.1 + 1 ;
            gm_$modelcmd_set_mode(gm_$modelcmd_replace,st);
            goto size_choice; };
}
```

```
/*-------------------------------------------------------------------------*/
add_lake()
{
extern gm_$segment_id_t s1;
extern int select_from_menu();
extern gm_$pointreal_t pick_position();
extern int get_name();

/* declare variables */
status_$t               st;
gm_$point16_t           pt1;
gm_$pointreal_t         ptrl,position;
short                   radius,n1;
int                     choice,count,n;
gm_$string_t            str;
long                    fill;
char                    call_str[32];

/* Prompt to get lake name */
    n = get_name("ENTER LAKE NAME","up to 32 characters",call_str);
    if ( n > 32 ) n = 32;
    n1 = (short)n;
     count = 0;
    while( count <= n ){str[count] = call_str[count]; ++count;};

/* Select from PURPOSE_MENU segment */
      choice = select_from_menu(4);
      if( choice == 1 )          fill = 16;
      if( choice == 2 )          fill = 22;
      if( choice == 3 )          fill = 20;
      if( choice == 4 )          fill = 18;

/* Prompt to pick lake position */
    tell_user("CHOOSE THE LAKE","POSITION");
    position = pick_position();

/* Draw the lake */
    pt1.x = position.x;     pt1.y = position.y;
    radius = 48;
    gm_$segment_open(s1,st);
    gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
    gm_$fill_value(fill,st);
size_choice:
    gm_$circle_16(pt1,radius,true,st);
    gm_$segment_close(true,st);
    please_draw_scheme();
```

```
/* Select the lake size */
        choice = select_from_menu(5);
        gm_$segment_open(s1,st);
        if(choice == 3) {
                gm_$tag(str,n1,st);
                gm_$segment_close(true,st); }
        else {
                if( choice == 1 ) radius = radius * 1.1 ;
                if( choice == 2 ) radius = radius * 0.9 ;
                gm_$modelcmd_set_mode(gm_$modelcmd_replace,st);
                goto size_choice; };

}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the add element menu segment in the metafile */
int add_menu()
{
gm_$segment_id_t segment_id;

 status_$t st;
gm_$point16_t pt1;
int seg;

/* Create ADD_MENU segment */
      gm_$segment_create("ADD_MENU",
                                    (short)8, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT ELEMENT TYPE","FOR ADDING");

/* Make 7 blue menu boxes */
      menu_boxes(7);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);
      pt1.x = 2;
      pt1.y = -6;

/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"go back to",(short)10,st);
      pt1.y = pt1.y -5;
      gm_$text_2d16(pt1,0.0,"EDIT MENU",(short)9,st);
      pt1.y = pt1.y - 9;
      gm_$text_2d16(pt1,0.0,"LAKES",(short)5,st);

      pt1.y = pt1.y - 10;
      gm_$text_2d16(pt1,0.0,"INFLOWS",(short)7,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"into lakes",(short)10,st);

      pt1.y = pt1.y - 8;
      gm_$text_2d16(pt1,0.0,"HYDRO STATIONS",(short)14,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"between lakes",(short)13,st);
```

```
        pt1.y = pt1.y - 8;
        gm_$text_2d16(pt1,0.0,"NATURAL OUTLETS",(short)15,st);
        pt1.y = pt1.y - 4;
        gm_$text_2d16(pt1,0.0,"between lakes",(short)13,st);

        pt1.y = pt1.y - 8;
        gm_$text_2d16(pt1,0.0,"CONTROLLED OUTLETS",(short)18,st);
        pt1.y = pt1.y - 4;
        gm_$text_2d16(pt1,0.0,"between lakes",(short)13,st);

        pt1.y = pt1.y - 10;
        gm_$text_2d16(pt1,0.0,"THERMAL/OTHER ST.",(short)17,st);

        seg = segment_id; return seg;
}
```

```
/*-------------------------------------------------------------------------*/
add_outlet()
{
extern gm_$segment_id_t s4;
 extern int select_from_menu();
extern int picklake();
extern int get_name();

/* declare variables */
status_$t                   st;
gm_$point_array16_t         pt;
gm_$point16_t               pt1,pt2;
gm_$pointreal_t             ptrl,position;
short                       radius, n1, delta_y, delta_x, width, lake1_length,
                            lake2_length;
int                         choice,count,n;
gm_$string_t                str, lake1, lake2;
boolean                     fill;
char                        call_str[32];

/* Prompt to get outlet name */
    n = get_name("ENTER OUTLET NAME","up to 32 characters",call_str);
    if ( n > 32 ) n = 32;
    n1 = (short)n;
    count = 0;
    while( count <= n ){
                              str[count] = call_str[count];
                              ++count;};

/* Select the outlet the outlet goes to */

again1:
    tell_user("PICK DESTINATION","LAKE OF THE OUTLET");
    please_draw_scheme();
    pick_something(0.5,0.5);
    choice = picklake();
    if( choice == 1 ) {
                               gm_$inq_circle_16(pt2,radius,fill,st);
                               gm_$pick_command(gm_$step,st);
                               gm_$inq_tag(lake2,lake2_length,st);
                               gm_$segment_close(false,st);
    } else
                              { choice = select_from_menu(6);
                                if (choice == 1) return;
                                goto again1;
                              };
```

```
/* Prompt to pick the source position */

again2:
     tell_user("CHOOSE THE POSITION","OF THE SOURCE");
     please_draw_scheme();
     pick_something(0.5,0.5);
     choice = picklake();
     if( choice == 1 ) {
                                  gm_$inq_circle_16(pt1,radius,fill,st);
                                  gm_$pick_command(gm_$step,st);
                                  gm_$inq_tag(lake1,lake1_length,st);
                                  gm_$segment_close(false,st);
     } else
                                 { choice = select_from_menu(6);
                                   if (choice == 1) return;
                                   goto again2;
                                 };

/* Construct the rectangle */
     width = 5;
     delta_x = pt2.x - pt1.x;
     delta_y = pt2.y - pt1.y;
     gm_$segment_open(s4,st);
     gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);
     check(st);
     gm_$tag(lake1,lake1_length,st);
     gm_$tag(lake2,lake2_length,st);

size_choice:
     pt[0].x = pt1.x + delta_y/width;
     pt[0].y = pt1.y - delta_x/width;
     pt[1].x = pt1.x - delta_y/width;
     pt[1].y = pt1.y + delta_x/width;
     pt[2].x = pt2.x - delta_y/width;
     pt[2].y = pt2.y + delta_x/width;
     pt[3].x = pt2.x + delta_y/width;
     pt[3].y = pt2.y - delta_x/width;

     gm_$polyline_2d16((short)4,pt,true,true,st);
     check(st);
     gm_$segment_close(true,st);

     please_draw_scheme();
```

```
/* Select the outlet size */
        choice = select_from_menu(5);
        gm_$segment_open(s4,st);

        if(choice == 3) {
                gm_$tag(str,n1,st);
                gm_$segment_close(true,st); }
        else {
                if( choice == 1 ) width = width * 0.9 ;
                if( choice == 2 ) width = width * 1.1 + 1 ;
                gm_$modelcmd_set_mode(gm_$modelcmd_replace,st);
                goto size_choice; };

}
```

```
/*-------------------------------------------------------------------------*/
add_thermal()
{
 extern gm_$segment_id_t s6;
extern int select_from_menu();
extern gm_$pointreal_t pick_position();
extern int get_name();

/* declare variables */
status_$t                st;
gm_$point16_t            pt1,pt2;
gm_$pointreal_t          ptrl,position;
short                    radius,n1;
int                      choice,count,n;
gm_$string_t             str;
long                     fill;
char                     call_str[32];


/* Prompt to get thermal name */
     n = get_name("ENTER STATION NAME","up to 32 characters",call_str);
     if ( n > 32 ) n = 32;
     n1 = (short)n;
     count = 0;
               while( count <= n ){str[count] = call_str[count];
                                   ++count;};

/* Prompt to pick thermal position */
     tell_user("CHOOSE THE STATION","POSITION");
     position = pick_position();

/* Draw the thermal */
     pt1.x = position.x;
     pt1.y = position.y;
     radius = 48;
     gm_$segment_open(s6,st);
     gm_$modelcmd_set_mode(gm_$modelcmd_insert,st);

size_choice:

     pt2.x = pt1.x + 1.6 * radius ;
     pt2.y = pt1.y + radius;
     gm_$rectangle_16(pt1,pt2,true,st);
     gm_$segment_close(true,st);
     please_draw_scheme();
```

```
/* Select the thermal size */
        choice = select_from_menu(5);
        gm_$segment_open(s6,st);

        if(choice == 3) {
                gm_$tag(str,n1,st);
                gm_$segment_close(true,st); }
        else {
                if( choice == 1 ) radius = radius * 1.1 ;
                if( choice == 2 )
                radius = radius * 0.9 ;
                gm_$modelcmd_set_mode(gm_$modelcmd_replace,st);
                goto size_choice; };
}
```

```
/*--------------------------------------------------------------------------*/
/* This routine sets the right pick aperture size */

aperture()
{
status_$t st;
gm_$pointreal_t aperture_size;


/* Initialize aperture size */

     aperture_size.x = 0.0001;
     aperture_size.y = 0.0001;

gm_$pick_set_size(aperture_size,st);
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the cancel/proceed menu segment in the metafile */
int cancel_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create CANCEL_MENU segment */
      gm_$segment_create("CANCEL_MENU",
                                    (short)11, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("LAKE NOT PICKED","SUCCESSFULLY");

/* Make 2 blue menu boxes */
      menu_boxes(2);=

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1; pt1.y = -6;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"CANCEL INPUT",(short)12,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"(no lake available)",(short)19,st);

      pt1.y = pt1.y - 8;
      gm_$text_2d16(pt1,0.0,"RETRY PICKING",(short)13,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"A LAKE",(short)6,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* I'll tell you what's going on if there's an error! */
check(status)

status_$t status;
{

        if(status.all != status_$ok)

pfm_$error_trap(status);

}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view_control menu segment in the metafile */
int control_vw()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create CONTROL_VIEW segment */
      gm_$segment_create("CONTROL_VIEW",
                                    (short)12, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT GRAPHS FOR","DISPLAY");

/* Make 2 blue menu boxes */
      menu_boxes(2);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -8;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"BACK TO MAIN MENU",(short)17,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"HISTORIC DISCHARGES",(short)19,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* This routine makes the cursor active and puts it to (x,y) */
cursor(x,y)

float x,y;
{
status_$t st;
gm_$pointreal_t cursor_position;

/* make cursor active */
        gm_$cursor_set_active(true,st);

/* Put the cursor to a visible place in the viewport */
        cursor_position.x = x;
        cursor_position.y = y;
        gm_$cursor_set_position(cursor_position,st);
}
```

```
/*-------------------------------------------------------------------------*/
delete_control()
{
extern int pickcontrol();

long                    value;
status_$t               st;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
int                     ind;

/* Draw the picture */
        please_draw_scheme();

/* Pick ! */
      pick_something(0.5,0.5);

/* pick control for deleting */
      ind = pickcontrol();

/* If picked successfully - delete it */
      if ( ind == 1 ) {
                        again: gm_$pick_command(gm_$step,st);
                        if(st.all != status_$ok) goto delete;
                        gm_$inq_command_type(command_type,data_type,st);
                        if( command_type != gm_$tfillvalue ) goto again;
                        gm_$inq_fill_value(value,st);
                        if (value != (long)18) goto again;

                        delete: gm_$command_delete(st);
                        gm_$inq_command_type(command_type,data_type,st);
                        if( command_type != gm_$tfillvalue ) goto delete;
                        gm_$inq_fill_value(value,st);
                        if (value != (long)18) goto delete;
                        };
      gm_$segment_close(true,st);

please_draw_scheme();
}
```

```
/*-----------------------------------------------------------------*/
/* Edit the scheme */
delete_element()
{
extern int select_from_menu();
int choice;

/* Select from edit menu */
again: choice = select_from_menu(8);

/* Go to the appropriate module */
        if (choice == 2) delete_lake();
        if (choice == 3) delete_inflow();
        if (choice == 4) delete_hydro();
        if (choice == 5) delete_outlet();
        if (choice == 6) delete_control();
        if (choice == 7) delete_thermal();
        if (choice != 1) goto again;
}
```

```
/*-------------------------------------------------------------------------*/
delete_hydro()
{
extern int pickhydro();

long                    value;
status_$t               st;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
int                     ind;

/* prepare window 1 for picking */
      please_draw_scheme();

/* Pick */
      pick_something(0.5,0.5);

/* pick hydro for deleting */
      ind = pickhydro();

/* If picked successfully - delete it */
      if ( ind == 1 ) {
            again:
            gm_$pick_command(gm_$step,st);
            if(st.all != status_$ok) goto delete;
            gm_$inq_command_type(command_type,data_type,st);
            if( command_type != gm_$tfillvalue ) goto again;
            gm_$inq_fill_value(value,st);
            if (value != 18) goto again;

            delete:
            gm_$command_delete(st);
            gm_$inq_command_type(command_type,data_type,st);
            if( command_type != gm_$tfillvalue ) goto delete;
            gm_$inq_fill_value(value,st);
            if (value != 18) goto delete;
            };

      gm_$segment_close(true,st);

please_draw_scheme();
}
```

```
/*-------------------------------------------------------------------------*/
delete_inflow()
{
extern int pickinflow();

status_$t st;
 int ind;


/* Draw the picture */
        please_draw_scheme();

/* Pick ! */
      pick_something(0.5,0.5);

/* pick inflow for deleting */
      ind = pickinflow();

/* If picked successfully - delete it */

      if ( ind == 1 ) {
                                  gm_$command_delete(st); gm_$command_delete(st);
                                  gm_$pick_command(gm_$step,st);
                                  gm_$command_delete(st);
                                 };

      gm_$segment_close(true,st);


/*  Let's see the picture now ! */

please_draw_scheme();
}
```

```
/*-------------------------------------------------------------------------*/
delete_lake()
{
extern int picklake();
status_$t st; int ind;

/* prepare window 1 for picking */
      please_draw_scheme();

/* Pick */
      pick_something(0.5,0.5);

/* pick lake for deleting */
      ind = picklake();

/* If picked successfully - delete it */
      if ( ind == 1 ) {
                              gm_$command_delete(st);
                              gm_$command_delete(st);
                              gm_$pick_command(gm_$step,st);
                              gm_$command_delete(st);
                             };
      gm_$segment_close(true,st);

/*  Let's see the picture now ! */
please_draw_scheme();
}
```

```
/*-------------------------------------------------------------------------*/
 /* I shall make the delete element menu segment in the metafile */
int delete_menu()
{
gm_$segment_id_t segment_id;
status_$t st; gm_$point16_t pt1;
int seg;

/* Create DELETE_MENU segment */
      gm_$segment_create("DELETE_MENU",
                                    (short)11, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT ELEMENT TYPE","FOR DELETING");

/* Make 7 blue menu boxes */
      menu_boxes(7);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 2;
      pt1.y = -6;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"go back to",(short)10,st);
      pt1.y = pt1.y -5;
      gm_$text_2d16(pt1,0.0,"EDIT MENU",(short)9,st);

      pt1.y = pt1.y - 9;
      gm_$text_2d16(pt1,0.0,"LAKES",(short)5,st);

      pt1.y = pt1.y - 10;
      gm_$text_2d16(pt1,0.0,"INFLOWS",(short)7,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"into lakes",(short)10,st);

      pt1.y = pt1.y - 8;
      gm_$text_2d16(pt1,0.0,"HYDRO STATIONS",(short)14,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"between lakes",(short)13,st);
```

```
        pt1.y = pt1.y - 8;
        gm_$text_2d16(pt1,0.0,"NATURAL OUTLETS",(short)15,st);
        pt1.y = pt1.y - 4;
        gm_$text_2d16(pt1,0.0,"between lakes",(short)13,st);

        pt1.y = pt1.y - 8;
        gm_$text_2d16(pt1,0.0,"CONTROLLED OUTLETS",(short)18,st);
        pt1.y = pt1.y - 4;
        gm_$text_2d16(pt1,0.0,"between lakes",(short)13,st);

        pt1.y = pt1.y - 10;
        gm_$text_2d16(pt1,0.0,"THERMAL/OTHER ST.",(short)17,st);

        seg = segment_id; return seg;
}
```

```
/*-------------------------------------------------------------------------*/
delete_outlet()
{
 extern int pickoutlet();

status_$t st;
int ind;

/* Draw the picture */
        please_draw_scheme();

/* Pick ! */
      pick_something(0.5,0.5);

/* pick outlet for deleting */
      ind = pickoutlet();

/* If picked successfully - delete it */

      if ( ind == 1 ) {
                                gm_$command_delete(st);
                                gm_$command_delete(st);
                                gm_$command_delete(st);
                                gm_$pick_command(gm_$step,st);
                                gm_$command_delete(st);
                               };
      gm_$segment_close(true,st);

/*  Let's see the picture now ! */
please_draw_scheme(); }
```

```
/*-------------------------------------------------------------------------------*/
delete_thermal()
{
extern int pickthermal();
status_$t st; int ind;

/* Draw the picture */
        please_draw_scheme();

/* Pick ! */
      pick_something(0.5,0.5);

/* pick thermal for deleting */
      ind = pickthermal();

/* If picked successfully - delete it */
      if ( ind == 1 ) {
                                  gm_$command_delete(st);
                                  gm_$pick_command(gm_$step,st);
                                  gm_$command_delete(st);
                                 };
      gm_$segment_close(true,st);

/*  Let's see the picture now ! */

please_draw_scheme();
}
```

```
/* ------------------------------------------------------------------------ */
/* Finding the right format for display of values between two extremes */
 double dist(ymin,ymax)

double ymin, ymax;
{
 double dd, ed;

ed = 1.0;
dd = ymax - ymin;
if( dd == 0.0 ) goto out;

while ( ed < dd ) {ed = 10.0 * ed;};
while ( ed > dd ) {
                                ed = 0.1 * ed;};

out:
        return ed;
}
```

```
/*-------------------------------------------------------------------------*/
 /* I shall make the add_delete_menu segment in the metafile */
int edit_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg,n_instances;

/* Create EDIT_MENU segment */
      gm_$segment_create("EDIT_MENU",
                                    (short)9, segment_id, st);check(st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);

/* Red box prompt */
    message("EDITING THE","SCHEME");

/* Make 3 blue menu boxes */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 2;
      pt1.y = -8;
      gm_$text_2d16(pt1,0.0,"END EDITING",(short)11,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"ADD ELEMENT",(short)11,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"DELETE ELEMENT",(short)14,st);

      seg = segment_id; return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* Edit the scheme */
edit_mode()
{
extern int select_from_menu();
int choice;

/* Select from edit menu */
again: choice = select_from_menu(3);

/* Go to the appropriate module */
        if (choice == 2) add_element();
        if (choice == 3) delete_element();
        if (choice != 1) goto again;

}
```

```
/*------------------------------------------------------------------------*/
/* start the keyboard */
enable_keyboard()
{
gm_$keyset_t keyset;
status_$t st;

/* Enable all letters.  */
        lib_$add_to_set(keyset,SETSIZE,ret);
        lib_$add_to_set(keyset,SETSIZE,lf);
        lib_$add_to_set(keyset,SETSIZE,CR);
        lib_$add_to_set(keyset,SETSIZE,LF);
        lib_$add_to_set(keyset,SETSIZE,BS);
        lib_$add_to_set(keyset,SETSIZE,'A');
        lib_$add_to_set(keyset,SETSIZE,'a');
        lib_$add_to_set(keyset,SETSIZE,'B');
        lib_$add_to_set(keyset,SETSIZE,'b');
        lib_$add_to_set(keyset,SETSIZE,'C');
        lib_$add_to_set(keyset,SETSIZE,'c');
        lib_$add_to_set(keyset,SETSIZE,'D');
        lib_$add_to_set(keyset,SETSIZE,'d');
        lib_$add_to_set(keyset,SETSIZE,'E');
        lib_$add_to_set(keyset,SETSIZE,'e');
        lib_$add_to_set(keyset,SETSIZE,'F');
        lib_$add_to_set(keyset,SETSIZE,'f');
        lib_$add_to_set(keyset,SETSIZE,'G');
        lib_$add_to_set(keyset,SETSIZE,'g');
        lib_$add_to_set(keyset,SETSIZE,'H');
        lib_$add_to_set(keyset,SETSIZE,'h');
        lib_$add_to_set(keyset,SETSIZE,'I');
        lib_$add_to_set(keyset,SETSIZE,'i');
        lib_$add_to_set(keyset,SETSIZE,'J');
        lib_$add_to_set(keyset,SETSIZE,'j');
        lib_$add_to_set(keyset,SETSIZE,'K');
        lib_$add_to_set(keyset,SETSIZE,'k');
        lib_$add_to_set(keyset,SETSIZE,'L');
        lib_$add_to_set(keyset,SETSIZE,'l');
        lib_$add_to_set(keyset,SETSIZE,'M');
        lib_$add_to_set(keyset,SETSIZE,'m');
        lib_$add_to_set(keyset,SETSIZE,'N');
        lib_$add_to_set(keyset,SETSIZE,'n');
        lib_$add_to_set(keyset,SETSIZE,'O');
        lib_$add_to_set(keyset,SETSIZE,'o');
        lib_$add_to_set(keyset,SETSIZE,'P');
        lib_$add_to_set(keyset,SETSIZE,'p');
        lib_$add_to_set(keyset,SETSIZE,'Q');
```

```
        lib_$add_to_set(keyset,SETSIZE,'q');
        lib_$add_to_set(keyset,SETSIZE,'R');
        lib_$add_to_set(keyset,SETSIZE,'r');
        lib_$add_to_set(keyset,SETSIZE,'S');
        lib_$add_to_set(keyset,SETSIZE,'s');
        lib_$add_to_set(keyset,SETSIZE,'T');
        lib_$add_to_set(keyset,SETSIZE,'t');
        lib_$add_to_set(keyset,SETSIZE,'U');
        lib_$add_to_set(keyset,SETSIZE,'u');
        lib_$add_to_set(keyset,SETSIZE,'V');
        lib_$add_to_set(keyset,SETSIZE,'v');
        lib_$add_to_set(keyset,SETSIZE,'W');
        lib_$add_to_set(keyset,SETSIZE,'w');
        lib_$add_to_set(keyset,SETSIZE,'X');
        lib_$add_to_set(keyset,SETSIZE,'x');
        lib_$add_to_set(keyset,SETSIZE,'Y');
        lib_$add_to_set(keyset,SETSIZE,'y');
        lib_$add_to_set(keyset,SETSIZE,'Z');
        lib_$add_to_set(keyset,SETSIZE,'z');
        lib_$add_to_set(keyset,SETSIZE,'1');
        lib_$add_to_set(keyset,SETSIZE,'2');
        lib_$add_to_set(keyset,SETSIZE,'3');
        lib_$add_to_set(keyset,SETSIZE,'4');
        lib_$add_to_set(keyset,SETSIZE,'5');
        lib_$add_to_set(keyset,SETSIZE,'6');
        lib_$add_to_set(keyset,SETSIZE,'7');
        lib_$add_to_set(keyset,SETSIZE,'8');
        lib_$add_to_set(keyset,SETSIZE,'9');
        lib_$add_to_set(keyset,SETSIZE,'0');
        lib_$add_to_set(keyset,SETSIZE,'_');
        lib_$add_to_set(keyset,SETSIZE,'-');

gm_$input_enable(gm_$keystroke,keyset,st); }
```

```
/*  ------------------------------------------------------------------------*/
enable_mouse()
{
status_$t st;
gm_$keyset_t keyset;

/* Enable all three buttons on the mouse. */
        lib_$init_set(keyset, SETSIZE);

lib_$add_to_set(keyset, SETSIZE, 'a');
lib_$add_to_set(keyset, SETSIZE, 'b');
lib_$add_to_set(keyset, SETSIZE, 'c');

        gm_$input_enable(gm_$buttons,
                                keyset, st);
}
```

```
/*-------------------------------------------------------------------*/
/* I shall start scheme segment of the metafile */
get_it_started()
{
extern gm_$segment_id_t segment_id, s1, s2, s3, s4, s5, s6;

status_$t                                   st;
gm_$point16_t                               translate, pt1, pt2;

/* Create and name the main segment. */
      gm_$segment_create("scheme",
                                  (short)6, segment_id, st);

/* Get the right text size. */
      gm_$text_size(28.0,st);
      gm_$draw_value((long)20,st);
      gm_$text_value((long)20,st);

/* make a frame */
      pt1.x = 50 ;
      pt1.y = 0 ;
      pt2.x = 1100;
      pt2.y = 1000;
      gm_$rectangle_16(pt1,pt2,false,st);

/* Close the segment.*/
      gm_$segment_close(true,
                                  st);

/* Create and name the lake segment. */
      gm_$segment_create("lakes",
                                  (short)5, s1, st);
      gm_$segment_close(true,st);

/* Create and name the stations segment. */
      gm_$segment_create("stations",
                                  (short)8, s2, st);
      gm_$segment_close(true,
                                  st);

/* Create and name the inflows segment. */
      gm_$segment_create("inflows",
                                  (short)7, s3, st);
      gm_$fill_value((long)18,st);
      gm_$segment_close(true,
                                  st);
```

```
/* Create and name the outlets segment. */
        gm_$segment_create("outlets",
                                        (short)7, s4, st);
        gm_$fill_value((long)18,st);
        gm_$segment_close(true,
                                    st);

/* Create and name the controlled outlets segment. */
        gm_$segment_create("controlled",
                                        (short)10, s5, st);
        gm_$segment_close(true,
                                    st);

/* Create and name the generation sources segment. */
        gm_$segment_create("gens",
                                        (short)4, s6, st);
        gm_$fill_value((long)21,st);
        gm_$segment_close(true,
                                    st);

/* Open again the main segment. */
        gm_$segment_open(
                                  segment_id, st);

/* Instance all other segments */
        translate.x = 0  ;  translate.y = 0 ;
        gm_$instance_translate_2d16(s6,translate,st);check(st);
        gm_$instance_translate_2d16(s5,translate,st);check(st);
        gm_$instance_translate_2d16(s4,translate,st);check(st);
        gm_$instance_translate_2d16(s3,translate,st);check(st);
        gm_$instance_translate_2d16(s2,translate,st);check(st);
        gm_$instance_translate_2d16(s1,translate,st);check(st);


/* Close the segment.*/
        gm_$segment_close(true,
                                    st);
}
```

```
/*-------------------------------------------------------------------------------*/
/* Get an element's name str3, using a prompt that consist of str1 and str2. */
/* This routine returns number of characters in the str3. */
int get_name(str1,str2,str3)

char        str1[], str2[], str3[];
{
gm_$string_t str;
status_$t st;
gm_$segment_id_t segment_id;
gm_$boundsreal_t bounds;
gm_$pointreal_t ptrl,position;
gm_$point16_t pt1;
gm_$event_t response;
char seq;
short count, viewport_id;
int count1;

/* Select viewport no 1 */
     gm_$viewport_select((short)1,st);

/* Make cursor active and put it in the right place */
     cursor(0.005,0.43);

/* Open a temporary segment for the message  */
     gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
     bounds.xmin = -2.0;
     bounds.xmax = 42.0;
     bounds.ymin = -36.0;
     bounds.ymax = 17.0;
     pt1.x = 0;
     pt1.y = -2;

/* Enter prompts to the picture */
     message(str1,str2);

/* Display the whole thing */
     gm_$text_background_value((long)0,st);
     gm_$segment_close(true,st);
     gm_$display_segment_part(segment_id,bounds,st);

/* Get the name */
      enable_keyboard();
      count = 0;
      count1 = 0;
```

```
again:

gm_$input_event_wait(true,response, seq, ptrl, viewport_id, position, st);
if(seq != ret && seq != lf && seq != CR && seq != LF)
                { gm_$segment_open(segment_id,st);

                  if(seq == BS) { --count1;
                                  str[count1] = ' ';
                                  gm_$text_2d16(pt1,0.0,str,count,st);
                                  --count ; }
                  else {
                                  str[count1] = seq;
                                  str3[count1] = seq;
                                  ++count1;
                                  ++count;
                                  gm_$text_2d16(pt1,0.0,str,count,st);}
                                  ;

                  gm_$display_segment_part(segment_id,bounds,st);
                  gm_$segment_close(true,st);
                  goto again;};

/* Disble input, destroy segment, de-activate cursor */
      gm_$input_disable(gm_$keystroke,st);
      gm_$segment_open(segment_id,st);
      gm_$segment_delete(st);
      gm_$cursor_set_active(false,st);

      return count1;
}
```

```
/* ------------------------------------------------------------------------------ */
/* Routine that draws graphs reading data from file *filename */
 graph(filename)

char *filename;
{
 extern short viewport_id1;
FILE *ld;

char                    ch;
double                  x[100],y[100][6];
float                   xx,yy[6],hi;
status_$t               st;
int                     i,j,numb,np,count,col;
gm_$point16_t           pt;
gm_$point_array16_t     grph;
short                   n_short;
gm_$string_t            tag;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;

/* Open the temporary segment for display */
     hi = 48.0;
     bounds.xmin = -150.0;
     bounds.xmax = 900.0;
     bounds.ymin = -150.0;
     bounds.ymax = 850.0;
                    gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
                    gm_$text_size(hi,st);

/* Prepare filename and open it */
       ld = fopen(filename,"r");

/* Read the titles */
       count = 0;
title:
       j = fscanf(ld,"%c",&ch);
       if( j > 0 && ch > 31 ) {tag[count] = ch;
                               ++count;
                               goto title;};
       n_short = count;
       pt.x = 50;
       pt.y = 820;
       gm_$text_2d16(pt,(float)0.0,tag,n_short,st);
       count = 0;
```

```
x_title:
        j = fscanf(ld,"%c",&ch);
        if( j > 0 && ch > 31 ){
                tag[count] = ch;
                ++count;
                goto x_title;};
        n_short = count;
        pt.x = 0;
        pt.y = -60;
        gm_$text_2d16(pt,(float)0.0,tag,n_short,st);
        count = 0;
y_title:
        j = fscanf(ld,"%c",&ch);
        if( j > 0 && ch > 31 ){
                tag[count] = ch;
                ++count;
                goto y_title;};
        n_short = count;
        pt.x = -100;
        pt.y = 0;
        gm_$text_2d16(pt,(float)-90.0,tag,n_short,st);

/* Number of curves  */
        j = fscanf(ld,"%d",&numb);
        j = fscanf(ld,"%c",&ch);
         if( numb > 6 ) numb = 6;

/* Curve label */
        i = 0;
        col = 23;
        pt.y = -100;
        pt.x = -280;
        while ( i < numb ) {
                count = 0;

                curve_labels:
                    j = fscanf(ld,"%c",&ch);
                    if( j > 0 && ch > 31 ){
                            tag[count] = ch;
                            ++count;
                            goto curve_labels;};
                 n_short = count;
                 pt.x = pt.x + 150;
                 gm_$text_value(col,st);
                 gm_$text_2d16(pt,(float)0.0,tag,n_short,st);
                  ++i,--col;};
```

```
        col = 20;
        gm_$text_value(col,st);
        count = 0; i = 0;

curves:
        j = fscanf(ld,"%f",&xx);
        if( j > 0 ) {count = 0;
                x[i] = xx;
                 while( count < numb ){
                        j = fscanf(ld,"%f",&xx);
                        y[i][count] = xx;
                        ++count;};
                 ++i;
                  goto curves;};

        np = i; fclose(ld);

         make_y_axis(y,np,numb);
         make_x_axis(x,np);

/* Plus number of decimal spaces */
        count = 0;
        col = 23;

        while (count < numb){
                i = 0;
                while ( i < np ){
                        grph[i].y = y[i][count] * 800;
                        grph[i].x = x[i] * 800;
                             /* Plus number of decimal spaces */
                         ++i;};
                gm_$draw_value(col,st);
                n_short = np;
                gm_$polyline_2d16(n_short,grph,false,false,st);
                ++count;
                --col;};

gm_$segment_close(true,st);
gm_$viewport_select(viewport_id1,st);
gm_$display_segment_part(segment_id,bounds,st);
gm_$segment_open(segment_id,st);
gm_$segment_delete(st);
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view_hydro menu segment in the metafile */
int hydro_vw()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
 int seg;

/* Create LAKE_VIEW segment */
      gm_$segment_create("HYDRO_VIEW",(short)1, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT GRAPHS FOR","DISPLAY");

/* Make 4 blue menu boxes */
      menu_boxes(4);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -8;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"BACK TO MAIN MENU",(short)17,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"EFFICIENCY CURVE",(short)16,st);

      pt1.y = pt1.y - 10;
      gm_$text_2d16(pt1,0.0,"OPEN WATER TAILRACE",(short)19,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"CURVES",(short)6,st);

      pt1.y = pt1.y - 8;
      gm_$text_2d16(pt1,0.0,"ICE COVER TAILRACE",(short)18,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"CURVES",(short)6,st);

      seg = segment_id;
return seg;      }
```

```
/*-----------------------------------------------------------------------*/
/* Identify all the important segments by their numbers for further use. */
identify_segments()
{
extern gm_$segment_id_t segment_id, s1, s2, s3, s4, s5, s6;

status_$t                                   st;
long                                        n_instances;

/* Create and name the main segment. */
      gm_$segment_inq_id("scheme",
                                    (short)6, segment_id, n_instances, st);
      gm_$segment_inq_id("lakes",
                                    (short)5, s1, n_instances, st);
      gm_$segment_inq_id("stations",
                                    (short)8, s2, n_instances, st);
      gm_$segment_inq_id("inflows",
                                    (short)7, s3, n_instances, st);
      gm_$segment_inq_id("outlets",
                                    (short)7, s4, n_instances, st);
      gm_$segment_inq_id("controlled",
                                    (short)10, s5, n_instances, st);
      gm_$segment_inq_id("gens",
                                    (short)4, s6, n_instances, st);

}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view_inflow menu segment in the metafile */
int inflow_vw()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create INFLOW_VIEW segment */
      gm_$segment_create("INFLOW_VIEW",
                                    (short)11, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT GRAPHS FOR","DISPLAY");

/* Make 3 blue menu boxes */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -8;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"BACK TO MAIN MENU",(short)17,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"HISTORIC FLOWS",(short)14,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"EMMA FLOWS",(short)10,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view_lake menu segment in the metafile */
int lake_vw()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create LAKE_VIEW segment */
      gm_$segment_create("LAKE_VIEW",
                                    (short)9, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT GRAPHS FOR","DISPLAY");

/* Make 4 blue menu boxes */
      menu_boxes(4);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -8;
/* define place for text, and write command */
       gm_$text_2d16(pt1,0.0,"BACK TO MAIN MENU",(short)17,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"RATING CURVE",(short)12,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"VALUE OF WATER",(short)14,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"HISTORIC LEVELS",(short)15,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------*/
 /* I shall make the main menu segment in the metafile */
int main_menu()
{
gm_$segment_id_t segment_id;

status_$t st;
gm_$point16_t pt1;
int seg;

/* Create MAIN_MENU segment */
      gm_$segment_create("MAIN_MENU",
                                    (short)9, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);

/* Red box prompt */
    message("SELECT A BLUE BOX"," ");

/* Make 3 blue menu boxes */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);
      pt1.x = 2;          pt1.y = -8;

/* define place for text, and write command */
gm_$text_2d16(pt1,0.0,"END",(short)3,st);
      pt1.y = pt1.y - 10;
      gm_$text_2d16(pt1,0.0,"EDIT SCHEME",(short)11,st);
      pt1.x = 1;
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"add/delete elements",(short)19,st);
      pt1.x = 2;
      pt1.y = pt1.y - 8;
      gm_$text_2d16(pt1,0.0,"PICK ELEMENT",(short)12,st);
      pt1.x = 1;
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"view related data",(short)17,st);
      seg = segment_id;

return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* Draw the x-axis */
make_x_axis(y,n)

double y[]; int n;
{
extern double min_array(),max_array(),dist();

status_$t                st;
gm_$point16_t            pt;
gm_$point_array16_t      point_arr;
double                   ymin,ymax,ed,lg,cut;
int                      i,ii,j,main_count, count,ct,ns,incr,nn,dd;
short                    radius, nspaces;
char                     c[80];
gm_$string_t             str;
float                    hight;

/* initialize what you'l need later */
     nn = 10;
     dd = 80;
     main_count = 0;
     count = 0;
     pt.x = 0;
     pt.y = 0;
     radius= 4;
     hight = 40.0;

/* draw the axis line */
          point_arr[0].x = 0;
          point_arr[1].x = 800;
          point_arr[0].y = 0;
          point_arr[1].y = 0;
     gm_$polyline_2d16((short)2,point_arr,false,false,st);

/* Plus number of decimal spaces */
     ymin = min_array(y,n);
     ymax = max_array(y,n);
     ed = dist(ymin,ymax);
     ymax = (ymax-ymin)/5.0;
     if ( ymax <= ed ) { nn = 5;
                         dd = 160 ;};
     ymax = ymax*2.0;
     if (ymax <= ed ) { nn = 3;
                         dd = 267; };
```

```
/*-------------------------------------------------------------------------*/
/* Draw the x-axis */
make_x_axis(y,n)

double y[]; int n;
{
extern double min_array(),max_array(),dist();

status_$t                st;
gm_$point16_t            pt;
gm_$point_array16_t      point_arr;
double                   ymin,ymax,ed,lg,cut;
int                      i,ii,j,main_count, count,ct,ns,incr,nn,dd;
short                    radius, nspaces;
char                     c[80];
gm_$string_t             str;
float                    hight;

/* initialize what you'l need later */
    nn = 10;
    dd = 80;
    main_count = 0;
    count = 0;
    pt.x = 0;
    pt.y = 0;
    radius= 4;
    hight = 40.0;

/* draw the axis line */
        point_arr[0].x = 0;
        point_arr[1].x = 800;
        point_arr[0].y = 0;
        point_arr[1].y = 0;
    gm_$polyline_2d16((short)2,point_arr,false,false,st);

/* Plus number of decimal spaces */
    ymin = min_array(y,n);
    ymax = max_array(y,n);
    ed = dist(ymin,ymax);
    ymax = (ymax-ymin)/5.0;
    if ( ymax <= ed ) { nn = 5;
                dd = 160 ;};
    ymax = ymax*2.0;
    if (ymax <= ed ) { nn = 3;
                dd = 267; };
```

```
/* Plus number of decimal spaces */
     incr = ed;
     ns = -1;
     lg = log10(ed);
     if(lg < 0)      { ns = - lg;
                       incr = incr/ed;};
     ii = ymin/ed;
     ii = ii * ed;
     cut = ii;
     if(lg < 0) ii = ii/ed;
     while ( main_count < nn ){ii = ii + incr;
                                count = 0;
                                if( ii == 0 )
                                     { c[0] = '0';
                                     count = 1; }
                                else {
                                      i = abs(ii);
                                      while ( i != 0 ){
                                         if(count == ns){c[count] = '.'; ++count;};
                                         j = i;
                                         i = i/10;
                                         j = j - i*10;
                                          c[count] = j + 48; ++count;
                                      };
                                };

     if ( ii < 0 )   { c[count] = '-';
                               ++count;};
     nspaces = count;
     ct = 0;
     while (ct < count){
                                str[ct] = c[count -1 - ct];
                                ++ct;};
    pt.y = 0; pt.x = pt.x + dd;
    gm_$circle_16(pt,radius,true,st);
    pt.y = -30 ;
    gm_$text_2d16(pt,(float)45.0,str,nspaces,st);
    ++main_count; };

   ed = nn * ed;
   count = 0;
   while(count < n){
                          y[count] = (y[count] - cut)/ed;
                           ++count;
                    };
}
```

```
/* ------------------------------------------------------------------------ */
/* Draw the y-axis */
make_y_axis(y,n,numb)

double y[100][6]; int n,numb;
{
extern double min_matrix(),max_matrix(),dist();

status_$t                st;
gm_$point16_t            pt;
gm_$point_array16_t      point_arr;
double                   ymin,ymax,ed,lg,cut;
int                      i,ii,j,main_count, count,ct,ns,incr,dd,nn;
short                    radius, nspaces;
char                     c[80];
gm_$string_t             str;
float                    hight;

/* initialize what you'l need later */
     nn = 10;
     dd = 80;
     main_count = 0;
     count = 0;
     pt.x = 0;
     pt.y = 0;
     radius= 4;
     hight = 40.0;

/* draw the axis line */
     point_arr[0].x = 0;
     point_arr[1].x = 0;
     point_arr[0].y = 0;
     point_arr[1].y = 800;
     gm_$polyline_2d16((short)2,point_arr,false,false,st);

/* try it */
     ymin = min_matrix(y,n,numb);
     ymax = max_matrix(y,n,numb);
     ed = dist(ymin,ymax);
     ymax = (ymax-ymin)/5.0;
     if ( ymax <= ed ) { nn = 5;
                         dd = 160 ;};
     ymax = ymax*2.0;
     if (ymax <= ed ) { nn = 3;
                         dd = 267; };
```

```
/* Plus number of decimal spaces */
     incr = ed;
     ns = -1;
     lg = log10(ed);
     if(lg < 0){ ns = - lg;
                 incr = incr/ed;};
     ii = ymin/ed;
     ii = ii * ed;
     cut = ii;
     if(lg < 0) ii = ii/ed;
     while ( main_count < nn ) {
                 ii = ii + incr;
                 count = 0;
                 i = abs(ii);
                 if( ii == 0 ) { c[0] = '0';
                                 count = 1;}
                 else {while ( i != 0 ){
                                  if(count == ns){c[count] = '.'; ++count;};
                                  j = i;
                                  i = i/10;
                                  j = j - i*10;
                                  c[count] = j + 48;
                                   ++count;};
                    };
                  if ( ii < 0 )   { c[count] = '-';++count;};

                  nspaces = count;
                  ct = 0;
                  while (ct < count){str[ct] = c[count -1 - ct]; ++ct;};
                  pt.x = 0;   pt.y = pt.y + dd;
                  gm_$circle_16(pt,radius,true,st);
                  pt.x = - nspaces * hight/2;
                  gm_$text_2d16(pt,0.0,str,nspaces,st);

/* Plus number of decimal spaces */
                  ++main_count; };

   ed = nn * ed; ct = 0;
   while(ct < numb){
           count = 0;
           while(count < n){
                   y[count][ct] = (y[count][ct] - cut)/ed;
                   /* Plus number of decimal spaces */
                   ++count;};
            ++ct;};
}
```

```
/* ------------------------------------------------------------------------ */
/* Finding a maximum of an array */

double max_array(y,n)

double y[]; int n;
{
double maxi;
int count;

maxi = y[0];
 count= 0;

while ( count < n ){
                                if ( y[count] > maxi ) maxi = y[count];
                                ++count;
                                };
return maxi;
}
```

```
/* ------------------------------------------------------------------------ */
/* Finding a maximum of an array */

double max_matrix(y,n,numb)

double y[100][6];
int n,numb;
{
double maxi;
int count,ct;

maxi = y[0][0];

ct = 0;
while(ct < numb){
        count= 0;
        while ( count < n ) {
                        if ( y[count][ct] > maxi ) maxi = y[count][ct];
                        ++count; };
        ++ct;};

return maxi;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make up to 9 menu boxes */
menu_boxes(n) int n;
{
 short count=1;
 int m;

 status_$t st;
gm_$point16_t pt1, pt2;
gm_$string_t tag;

/* Set colors */
      gm_$draw_value((long)20,st);
      gm_$fill_value((long)18,st);

/* Not more than 9 */
      m = n;
      if( n >= 9 ) m = 9;

       pt1.x = 0;      /* define starting co-ordinates */
       pt2.x = 40;
       pt1.y = 0;
       pt2.y = 10;
while ( count <= m ){     /* count the boxes, prepare, and draw them */
                pt1.y = pt1.y - 12;
                pt2.y = pt2.y - 12;

                tag[0]=(char)count;
                ++count;
                gm_$rectangle_16( pt1,pt2, true, st);
                gm_$tag(tag,(short)1,st);
                gm_$rectangle_16( pt1,pt2, false, st);
                gm_$tag(tag,(short)1,st); }
}
```

```
/*-------------------------------------------------------------------------*/
/*Give a message in the red box.  Has to be called from an open segment */
message(str1,str2)

char        str1[], str2[];
{
 short n1, count;
gm_$string_t message;
status_$t st;
gm_$point16_t pt1;

/* Draw a red box background, and define the place for writing */
     red_box_prompt();
     pt1.x = 1;
     pt1.y = 10;
     gm_$text_background_value((long)16,st);

/* Prepare and write first string */
     count = 0;
     while ( str1[count] != '\0' )
               { message[count] = str1[count];
                 ++count;};
     n1 = count;
     gm_$text_2d16(pt1,0.0,message,n1,st);

/* Prepare and write second string */
     count = 0;
     while ( str2[count] != '\0' )
               { message[count] = str2[count];
                 ++count;};
     n1 = count;

     pt1.y = 5;
     gm_$text_2d16(pt1,0.0,message,n1,st);

}
```

```
/* ------------------------------------------------------------------------ */
/* Finding a minimum of an array */

double min_array(y,n)

double y[]; int n;
{
double mini;
int count;

mini = y[0];
 count= 0;

while ( count < n ) {
                                if ( y[count] < mini ) mini = y[count];
                                ++count;
                                };
return mini;
}
```

```
/* ------------------------------------------------------------------------ */
/* Finding a minimum of an array */

double min_matrix(y,n,numb)

double y[100][6]; int n,numb;
{
double mini; int count,ct;

mini = y[0][0];

ct = 0;
while(ct < numb){
        count= 0;
        while ( count < n ) {
                                if ( y[count][ct] < mini ) mini = y[count][ct];
                                ++count; };
        ++ct;};

return mini;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view_lake menu segment in the metafile */
int outlet_vw()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create OUTLET_VIEW segment */
      gm_$segment_create("OUTLET_VIEW",
                                    (short)11, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT GRAPHS FOR","DISPLAY");

/* Make 3 blue menu boxes */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -8;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"BACK TO MAIN MENU",(short)17,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"HISTORIC FLOWS",(short)14,st);

      pt1.y = pt1.y - 10;
      gm_$text_2d16(pt1,0.0,"STAGE-DISCHARGE",(short)15,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"CURVES",(short)6,st);

      seg = segment_id;
      return seg;
}
```

```
 /*-------------------------------------------------------------------------*/
pickcontrol()
{
 extern gm_$segment_id_t segment_id, s5;
extern short viewport_id1;

status_$t                st;
short                    tag_length;
gm_$string_t             tag;
int                      ind;
gm_$command_type_t       command_type;
gm_$data_type_t          data_type;

      ind = 0;

/* open the segment */
      gm_$segment_open(s5,st);

/* Pick a rectangle command */
      gm_$pick_command(gm_$start,st);
      try_next:
                        gm_$pick_command(gm_$cnext,st);
                       if(st.all != status_$ok) goto out;
                       gm_$inq_command_type(command_type,data_type,st);
      if( command_type != gm_$tpolyline_2d ) goto try_next;
      ind = 1;

/* disable input from mouse */
out:
      gm_$input_disable(gm_$buttons,st);
      gm_$cursor_set_active(false,st);
      return ind;
}
```

```
/*-------------------------------------------------------------------------*/
pickhydro()
{
extern gm_$segment_id_t segment_id, s2;
extern short viewport_id1;

status_$t               st;
short                   tag_length;
gm_$string_t            tag;
int                     ind;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;

        ind = 0;

/* open the segment */
        gm_$segment_open(s2,st);

/* Pick a rectangle command */
        gm_$pick_command(gm_$start,st);

        try_next:
        gm_$pick_command(gm_$cnext,st);
         if(st.all != status_$ok) goto out;
         gm_$inq_command_type(command_type,data_type,st);
        if( command_type != gm_$tpolyline_2d ) goto try_next;
        ind = 1;

/* disable input from mouse */ out:
        gm_$input_disable(gm_$buttons,
        st); gm_$cursor_set_active(false,st); return ind;
}
```

```
/*-------------------------------------------------------------------------*/
pickinflow()
{
 extern gm_$segment_id_t segment_id, s3;
extern short viewport_id1;

status_$t                st;
short                    tag_length;
gm_$string_t             tag;
int                      ind;
gm_$command_type_t       command_type;
gm_$data_type_t          data_type;


      ind = 0;

/* open the segment */
      gm_$segment_open(s3,st);

/* Pick a rectangle command */
      gm_$pick_command(gm_$start,st);
      try_next: gm_$pick_command(gm_$cnext,st);

                        if(st.all != status_$ok) goto out;

                        gm_$inq_command_type(command_type,data_type,st);
      if( command_type != gm_$tpolyline_2d ) goto try_next;
      ind = 1;

/* disable input from mouse */

out:
      gm_$input_disable(gm_$buttons,st);
      gm_$cursor_set_active(false,st);

      return ind;
}
```

```
/*-------------------------------------------------------------------------*/
/* I make it possible to pick menu items */
int pick_item()
{
status_$t st;
gm_$event_t response;
gm_$pointreal_t ptrl, position;
short viewport_id, tag_length;
gm_$string_t tag;
char ignorance;
int item;
gm_$command_type_t command_type;
gm_$data_type_t data_type;

/* Pick something */

again:
    pick_something(0.12,0.7);

/* Pick a rectangle command */
      gm_$pick_command(gm_$start,st);

try_next:
      gm_$pick_command(gm_$cnext,st);
                        if(st.all != status_$ok) goto again;
                        gm_$inq_command_type(command_type,data_type,st);
      if( command_type != gm_$trectangle )      goto try_next;

/* Read the subsequent tag */
      gm_$pick_command(gm_$step,st);
      gm_$inq_command_type(command_type,data_type,st);
      if( command_type == gm_$ttag )
         {
          gm_$inq_tag(tag,tag_length,st);
           item = (int)tag[0];}
      else goto again;

/* disable input from mouse */
      gm_$input_disable(gm_$buttons,
                                st);

/* make cursor inactive */
      gm_$cursor_set_active(false,st);
      return item;
}
```

```
/*-------------------------------------------------------------------------*/
 picklake()
{
extern gm_$segment_id_t segment_id, s1;
extern short viewport_id1;

status_$t                   st;
short                       tag_length;
gm_$string_t                tag;
int                         ind;
gm_$command_type_t          command_type;
gm_$data_type_t             data_type;

      ind = 0;

/* open the segment */
      gm_$segment_open(s1,st);

/* Pick a rectangle command */
      gm_$pick_command(gm_$start,st);
      try_next: gm_$pick_command(gm_$cnext,st);
                  if(st.all != status_$ok) goto out;
                  gm_$inq_command_type(command_type,data_type,st);
      if( command_type != gm_$tcircle_2d ) goto try_next;
      ind = 1;

/* disable input from mouse */
out:
      if(ind != 1) gm_$segment_close(false,st); gm_$input_disable(gm_$buttons,
      st); gm_$cursor_set_active(false,st); return ind;
}
```

```
/*-------------------------------------------------------------------------*/
pickoutlet()
{
extern gm_$segment_id_t segment_id, s4;
extern short viewport_id1;

status_$t                st;
short                    tag_length;
gm_$string_t             tag;
int                      ind;
gm_$command_type_t       command_type;
gm_$data_type_t          data_type;

        ind = 0;

/* open the segment */
        gm_$segment_open(s4,st);

/* Pick a rectangle command */
        gm_$pick_command(gm_$start,st);

try_next:
        gm_$pick_command(gm_$cnext,st);
                        if(st.all != status_$ok) goto out;

                        gm_$inq_command_type(command_type,data_type,st);
        if( command_type != gm_$tpolyline_2d ) goto try_next;
        ind = 1;

/* disable input from mouse */
out:
        gm_$input_disable(gm_$buttons,st);
        gm_$cursor_set_active(false,st);

        return ind;
}
```

```
/*-------------------------------------------------------------------------*/
/* I make it possible to pick a position from the scheme window */

gm_$pointreal_t  pick_position()
{
status_$t                st;
gm_$event_t              response;
gm_$pointreal_t          ptrl,position;
short                    viewport_id;
char                     ignorance;

/* start the wqhole thing */
      please_draw_scheme();
      enable_mouse();

again:
      cursor(0.5,0.5);

/* Wait for YOU to click the mouse */
  gm_$input_event_wait(true,response, ignorance, ptrl, viewport_id, position, st);

/* Is it inside the window ? */
      if ( ptrl.x <= 0.244 ) goto again;

/* disable input from mouse */
      gm_$input_disable(gm_$buttons,
                                st);

/* make cursor inactive */
      gm_$cursor_set_active(false,st);

return position;
}
```

```
/*. ------------------------------------------------------------------- */
/* Routine that picks a position and sets aperture for picking */

pick_something(r1,r2)

float r1,r2;
{
 status_$t st;
gm_$event_t response;
gm_$pointreal_t ptrl, position ;
short viewport_id;
char ignorance;


/* make cursor active */
        cursor(r1,r2);
        enable_mouse();

/* Wait for YOU to click a mouse button */
      again:  gm_$input_event_wait(true,
                              response, ignorance, ptrl, viewport_id, position, st);
                              check(st);

/* Pick aperture definition */
      gm_$pick_set_center(position,st);
      aperture();
}
```

```
/*-------------------------------------------------------------------------*/
pickthermal()
{
extern gm_$segment_id_t segment_id, s6;
extern short viewport_id1;

status_$t                st;
short                    tag_length;
gm_$string_t             tag;
int                      ind;
gm_$command_type_t       command_type;
gm_$data_type_t          data_type;

      ind = 0;

/* open the segment */
      gm_$segment_open(s6,st);

/* Pick a rectangle command */
      gm_$pick_command(gm_$start,st);
      try_next: gm_$pick_command(gm_$cnext,st);
                     if(st.all != status_$ok) goto out;
                     gm_$inq_command_type(command_type,data_type,st);
      if( command_type != gm_$trectangle ) goto try_next;
      ind = 1;

/* disable input from mouse */ out:
      gm_$input_disable(gm_$buttons,st);
      gm_$cursor_set_active(false,st);
      return ind;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall draw the scheme of the system! */
please_draw_scheme()
{
extern short viewport_id1;
extern gm_$segment_id_t segment_id;

/* declare all those variables. */
status_$t st;
long n_instances;
gm_$boundsreal_t bounds;

/* Select the right viewport */
      gm_$viewport_select(viewport_id1,st);check(st);

/* Display the scheme */
      bounds.xmin = 50. ;
      bounds.xmax = 1100.;
      bounds.ymin = 0. ;
      bounds.ymax = 1000.;
      gm_$display_segment_part(segment_id,bounds,st);
      }
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the lake purpose menu segment in the metafile */
int purpose_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create PURPOSE_MENU segment */
      gm_$segment_create("PURPOSE_MENU",
                                    (short)12, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);

/* Red box prompt */
      message("CHOOSE THE LAKE","PURPOSE");

/* Make 4 blue menu boxes */
      menu_boxes(4);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.y = -8;
      pt1.x = 1;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"REGULATED FOR HYDRO",(short)19,st);

      pt1.y = pt1.y - 10;
      gm_$text_2d16(pt1,0.0,"REGULATED FOR",(short)13,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"NON HYDRO",(short)9,st);

      pt1.y = pt1.y - 8;
      gm_$text_2d16(pt1,0.0,"UNREGULATED",(short)11,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"DESTINATION OUT OF",(short)18,st);
      pt1.y = pt1.y - 4;
      gm_$text_2d16(pt1,0.0,"THE SYSTEM",(short)10,st);

      seg = segment_id;
      return seg;   }
```

```
/*-------------------------------------------------------------------------*/
/* red box for prompts - three prompts up to 18 chars ! */
red_box_prompt()
{
status_$t st;
gm_$point16_t pt1, pt2;

/* Set colors */
        gm_$draw_value((long)20,st);
        gm_$fill_value((long)16,st);

/* box */
         pt1.x = 0;
         pt2.x = 40;
         pt1.y = 3;
         pt2.y = 15;
        gm_$rectangle_16( pt1,
                                    pt2, true, st);
        gm_$rectangle_16( pt1, pt2,
                                    false, st);
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the cancel/proceed menu segment in the metafile */
int run_of_river_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create run_of_river_menu segment */
      gm_$segment_create("RUN_OF_RIVER",
                                    (short)12, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("MORE RUN OF RIVER","PLANTS DOWNSTREAM?");

/* Make 2 blue menu boxes */
      menu_boxes(2);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -10;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"YES",(short)3,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"NO",(short)2,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view menu segment in the metafile */
int scroll_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create view_menu segment */
      gm_$segment_create("SCROLL",
                                    (short)6, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SCROLL THE GRAPH","OR QUIT");

/* Make 3 blue menu box */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -10;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"QUIT VIEWING",(short)12,st);

      pt1.y = pt1.y - 12;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"SCROLL RIGHT",(short)12,st);

      pt1.y = pt1.y - 12;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"SCROLL LEFT",(short)11,st);

      seg = segment_id;
      return seg;
}
```

```
/*-----------------------------------------------------------------*/
/* I shall make the main menu and let YOU select an item */
int select_from_menu(i)
int i;
{
extern int pick_item();
extern int main_menu();
extern int add_menu();
extern int edit_menu();
extern int purpose_menu();
extern int size_menu();
extern int cancel_menu();
extern int run_of_river_menu();
extern int delete_menu();
extern int view_menu();
extern int lake_vw();
extern int hydro_vw();
extern int inflow_vw();
extern int thermo_vw();
extern int control_vw();
extern int outlet_vw();
extern int scroll_menu();

extern gm_$segment_id_t segment_id; int choice;

/* declare all the wierdows */
gm_$segment_id_t segmen;
status_$t st;
gm_$boundsreal_t bounds;
long n_instances;
int seg;
gm_$point16_t trans, size;

/* viewport 1 */
      gm_$viewport_select((short)1,st);

/* Create and display appropriate MENU segment */
      switch(i) {
      case 1:
                  seg = main_menu(); break;
      case 2:
                  seg = add_menu(); break;
      case 3:
                  seg = edit_menu(); break;
      case 4:
                  seg = purpose_menu(); break;
```

```
        case 5:
                seg = size_menu(); break;
        case 6:
                seg = cancel_menu(); break;
        case 7:
                seg = run_of_river_menu(); break;
        case 8:
                seg = delete_menu(); break;
        case 9:
                seg = view_menu(); break;
        case 10:
                seg = lake_vw(); break;
        case 11:
                seg = hydro_vw(); break;
        case 12:
                seg = inflow_vw(); break;
        case 13:
                seg = thermo_vw(); break;
        case 14:
                seg = control_vw(); break;
        case 15:
                seg = outlet_vw(); break;
        case 16:
                seg = scroll_menu(); break;
          };

/* Display the MAIN_MENU segment */
        bounds.xmin = -2.0 ; bounds.xmax = 42.0 ;
        bounds.ymin = -85.0 ; bounds.ymax = 16.0 ;
        segmen = seg; gm_$display_segment_part(segmen,bounds,st);

/* Pick an item */
        gm_$segment_open(segmen,st); choice = pick_item();

/* Close it */
        gm_$segment_delete(st);check(st);

return choice;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the size menu segment in the metafile */
int size_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create SIZE_MENU segment */
      gm_$segment_create("SIZE_MENU",
                                    (short)9, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("ADJUST THE SIZE OF","THE ELEMENT");

/* Make 3 blue menu boxes */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -8;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"GROW",(short)4,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"DECREASE SIZE",(short)13,st);

      pt1.y = pt1.y - 12;
      gm_$text_2d16(pt1,0.0,"ACCEPT CURRENT SIZE",(short)19,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------------*/
/* prompt that consist of str1 and str2. */
int tell_user(str1,str2)

char        str1[], str2[];
{
 status_$t st;
gm_$segment_id_t segment_id;
 gm_$boundsreal_t bounds;

/* Select viewport no 1 */
     gm_$viewport_select((short)1,st);

/* Open a temporary segment for the message  */
     gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
     bounds.xmin = -2.0;
     bounds.xmax = 42.0;
     bounds.ymin = -36.0;
     bounds.ymax = 17.0;

/* Enter prompts to the picture */
     message(str1,str2);

/* Display the whole thing */
     gm_$segment_close(true,st);
     gm_$display_segment_part(segment_id,bounds,st);

       gm_$segment_open(segment_id,st);
       gm_$segment_delete(st);
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view_thermo menu segment in the metafile */
int thermo_vw()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create THERMO_VIEW segment */
      gm_$segment_create("THERMO_VIEW",
                                    (short)11, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("SELECT GRAPHS FOR","DISPLAY");

/* Make 1 blue menu boxes */
      menu_boxes(1);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1; pt1.y = -8;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"BACK TO MAIN MENU",(short)17,st);

      seg = segment_id;
      return seg;
}
```

```
/* ------------------------------------------------------------------------------ */
/* Routine that draws graphs reading data from file *filename */
time_graph(sc,filename)

int sc;
char *filename;
{
extern short viewport_id1;

FILE                    *ld;
char                    ch;
double                  y[100][6],yhelp[100][6];
float                   xx,hi;
status_$t               st;
int                     col,plotp,i,j,numb,np,count,scr,ct,mm,yy,
                        month[100],year[100];
gm_$point16_t           pt;
gm_$point_array16_t     grph;
short                   n_short;
gm_$string_t            tag;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;

/* Open the temporary segment for display */
    hi = 48.0;
    bounds.xmin = -150.0;  bounds.xmax = 900.0;
    bounds.ymin = -150.0;  bounds.ymax = 850.0;
gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
gm_$text_size(hi,st);

/* Prepare filename and open it */
      ld = fopen(filename,"r");

/* Read the titles */
      count = 0;
title:
      j = fscanf(ld,"%c",&ch);
      if( j > 0 && ch > 31 ){
                  tag[count] = ch;
                  ++count;
                  goto title;};
      n_short = count;
      pt.x = 50;  pt.y = 820;
      gm_$text_2d16(pt,(float)0.0,tag,n_short,st);
      count = 0;
```

```
y_title:

        j = fscanf(ld,"%c",&ch);

        if( j > 0 && ch > 31 ){
                tag[count] = ch;
                ++count;
                goto y_title;};

        n_short = count;
        pt.x = -100;
        pt.y = 0;
        gm_$text_2d16(pt,(float)-90.0,tag,n_short,st);

/* Number of curves  */
        j = fscanf(ld,"%d",&numb);
        j = fscanf(ld,"%c",&ch);
        if( numb > 6 ) numb = 6;

/* Curve label */
        i = 0;
        col = 23;
        pt.y = -100;
         pt.x = -280;
        while ( i < numb ) {
                count = 0;

                curve_labels:

                j = fscanf(ld,"%c",&ch);

                if( j > 0 && ch > 31 ){
                        tag[count] = ch;
                        ++count;
                        goto curve_labels;};

                n_short = count;
                pt.x = pt.x + 150;
                gm_$text_value(col,st);
                gm_$text_2d16(pt,(float)0.0,tag,n_short,st);
                 ++i,
                --col;};

        col = 20;
        gm_$text_value(col,st);
        i = 0;
```

```
curves:

        if( j > 0 ) {
                count = 0;

                while( count < numb ){
                                j = fscanf(ld,"%f",&xx);
                                 y[i][count] = xx;
                                ++count;};

                 j = fscanf(ld,"%d%d",&mm,&yy);
                if( j >= 1 ){month[i] = mm;
                                year[i] = yy;};
                ++i;
                goto curves;};

        np = i;
        fclose(ld);

/* Make y axis */
        make_y_axis(y,np,numb);
        scr = sc;

/* Prepare 24 points to be shown */
        ct= scr*10;
        i = 0;
        while( ct < np && i < 24){
                        count = 0;

                        while( count < numb ){
                                yhelp[i][count] = y[ct][count];
                                ++count;};
                        ++ct;
                        ++i;};

        ct = scr*10;
        if(ct >= np) ct = np - 1;
        plotp = i;
        mm = month[ct];
        yy = year[ct];

        time_x_axis(mm,yy);

        count = 0;
        col = 23;
```

```
        while (count < numb){
                i = 0;

                while ( i < plotp ){
                        grph[i].y = yhelp[i][count] * 800;
                        grph[i].x = i * 35;
/* Plus number of decimal spaces */
                        ++i;};

                 gm_$draw_value(col,st);
                n_short = plotp;
                if(plotp > 1) gm_$polyline_2d16(n_short,grph,false,false,st);
                ++count;
                --col;};

gm_$viewport_select(viewport_id1,st);
check(st);
gm_$display_segment_part(segment_id,bounds,st);
check(st);
gm_$segment_close(true,st);
check(st);
gm_$segment_open(segment_id,st);
check(st);
gm_$segment_delete(st);
check(st);
}
```

```
/*-------------------------------------------------------------------------*/
/* Draw the time x-axis, given the starting month and the year */
time_x_axis(month,year)

int month,year;
{
status_$t                   st;
gm_$point16_t               pt;
gm_$point_array16_t         point_arr;
int                         i,j,m,main_count,count,ct;
char                        c[12], ch[4];
gm_$string_t                str;
float                       hight;

/* initialize what you'l need later */
    main_count = 0; pt.x = 0; pt.y = 0; hight = 40.0; m = month;
    c[0] = 'J'; c[1] = 'F'; c[2] = 'M'; c[3] = 'A'; c[4] = 'M'; c[5] = 'J'; c[6] = 'J';
    c[7] = 'A'; c[8] = 'S'; c[9] = 'O'; c[10]= 'N'; c[11]= 'D';

/* draw the axis line */
        point_arr[0].x = 0;
        point_arr[1].x = 840;
        point_arr[0].y = 0;
        point_arr[1].y = 0;
    gm_$polyline_2d16((short)2,point_arr,false,false,st);

/* try it */
   while ( main_count < 24 ){pt.x = pt.x + 35;
                             pt.y = 0;
                             gm_$circle_16(pt,(short)4,true,st);
                             if(m == 12) { m = 0;
                                           year = year + 1;
                                           i = year;
                                           count = 0;
                                           while ( i != 0 ){ j= i;
                                                             i = i/10;
                                                             j = j - i*10;
                                                             ch[count] = j + 48;
                                                             ++count;
                                                             ct = 0; };
                                           while (ct < 4){str[ct] = ch[3 - ct];
                                                             ++ct;};
                                         pt.y = -60;
                                         gm_$text_2d16(pt,90.0,str,(short)4,st);
                             };
```

```
                str[0] = c[m];
                pt.y = -30;
                gm_$text_2d16(pt,90.0,str,(short)1,st);
                ++main_count;
                ++m; };
}
```

```
/* ------------------------------------------------------------------------* /
/* Viewing controlled outlets related data */
view_control()
{
extern int pickcontrol();
extern short viewport_id1;

FILE                    *gf;
char                    fln[80];
status_$t               st;
int                     ind,n,typ,count,scr,mn,ind1;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;
gm_$string_t            name;
short                   name_length,n_short;
gm_$point16_t           pt;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
float                   hi;

/* Initialize string */
      fln[0] = 'c'; fln[1] = 'o'; fln[2] = 'n'; fln[3] = 't'; fln[4] = 'r';
      fln[5] = 'o'; fln[6] = 'l'; fln[7] = '_'; fln[8] = 's'; fln[9] = 't';
      fln[10] = 'r'; fln[11] = '/';

/* pick control for deleting */
      ind = pickcontrol();

/* If picked successfully - delete it */
try_again:
 if ( ind == 1 ) {
                        gm_$pick_command(gm_$step,st);
                        gm_$inq_command_type(command_type,data_type,st);
                        if(command_type == gm_$ttag) {
                                    gm_$inq_tag(name,name_length,st); }
                        else {goto try_again;};
                        }
                  else { ind = 0; goto out;};
      gm_$segment_close(true,st);

/* Open a temporary segment for the picture */
     gm_$viewport_select(viewport_id1,st);
     gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
     bounds.xmin = -150.0; bounds.xmax = 900.0;
     bounds.ymin = -100.0; bounds.ymax = 900.0;
     pt.x = 20; pt.y = 840; hi = 50.0; gm_$text_size(hi,st);
```

```
/* Put all that i snecessary in it */
    gm_$text_2d16(pt,0.0,name,name_length,st);
    pt.y = 870;
    gm_$text_2d16(pt,0.0,"CONTROLLED OUTLET:",(short)18,st);

/* Display the whole thing */
    gm_$segment_close(true,st);
    gm_$display_segment_part(segment_id,bounds,st);
    gm_$segment_open(segment_id,st);
    gm_$segment_delete(st);
    scr = 0;
    again:
        mn = 14;
        ind = select_from_menu(mn);
    move_gr:
        if(ind != 1) {
            if(ind == 2) {
                fln[12] = 'h'; fln[13] = 'i'; fln[14] = 's';
                fln[15] = 't'; fln[16] = 'o'; fln[17] = 'r';
                fln[18] = 'i'; fln[19] = 'c'; fln[20] = '_';
                fln[21] = 'f'; fln[22] = 'l'; fln[23] = 'o';
                fln[24] = 'w'; fln[25] = 's'; fln[26] = '/';
                n = 27;};
            count = 0;
            while(count < name_length){
                fln[n + count] = name[count]; ++count;};
            n = n + name_length;
            fln[n] = 0;
            time_graph(scr,fln);
            mn = 9;
            ind1 = select_from_menu(mn);
            if( ind1 != 1 ){
                if( ind1 == 2) scr = scr + 1;
                if( ind1 == 3) scr = scr -1;
                if(scr < 0) scr = 0;
                goto move_gr;};
            goto again; };
        ind = 1;
    out:
        return ind;
}
```

```
/* ------------------------------------------------------------------------* /
 /* Viewing hydro stations related data */
 view_hydro()
{
extern int pickhydro();
extern short viewport_id1;

char                    fln[80];
status_$t               st;
int                     ind,n,typ,count,scr,mn,ind1;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;
gm_$string_t            name;
short                   name_length,n_short;
gm_$point16_t           pt;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
float                   hi;


/* Initialize string */
      fln[0] = 'h'; fln[1] = 'y'; fln[2] = 'd'; fln[3] = 'r'; fln[4] = 'o'; fln[5] = '/';

/* pick hydro */
      ind = pickhydro();
      name_length = 0;


/* If picked successfully  */

try_next:

      if ( ind == 1 ) {
                gm_$pick_command(gm_$step,st);
                gm_$inq_command_type(command_type,data_type,st);
                if(command_type == gm_$ttag) {
                                gm_$inq_tag(name,name_length,st); }
                else {
                                gm_$pick_command(gm_$cnext,st);
                                if( st.all == status_$ok ) goto try_next;};
                }
      else { ind = 0;
                goto out;};

      gm_$segment_close(true,st);
```

```
/* Open a temporary segment for the picture */
        gm_$viewport_select(viewport_id1,st);
        gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
        bounds.xmin = -150.0;
        bounds.xmax = 900.0;
        bounds.ymin = -100.0;
        bounds.ymax = 900.0;
        pt.x = 20;
        pt.y = 840;
        hi = 50.0;
        gm_$text_size(hi,st);

/* Put all that i snecessary in it */
        gm_$text_2d16(pt,0.0,name,name_length,st);
        pt.y = 870;
        gm_$text_2d16(pt,0.0,"HYDRO STATION:",(short)13,st);

/* Display the whole thing */
        gm_$segment_close(true,st);
        gm_$display_segment_part(segment_id,bounds,st);
        gm_$segment_open(segment_id,st);
        gm_$segment_delete(st);
        scr = 0;

again:
        mn = 11;
        ind = select_from_menu(mn);

move_gr:
        if(ind != 1) {
                if(ind == 2) {
                   fln[6] = 'e'; fln[7] = 'f'; fln[8] = 'f'; fln[9] = 'i'; fln[10] = 'c';
                   fln[11] = 'i'; fln[12] = 'e'; fln[13] = 'n'; fln[14] = 'c'; fln[15] = 'y';
                   fln[16] = '_'; fln[17] = 'c'; fln[18] = 'u'; fln[19] = 'r'; fln[20] = 'v';
                   fln[21] = 'e'; fln[22] = '/'; n = 23;};
                if(ind == 3) {
                    fln[6] = 'o'; fln[7] = 'p'; fln[8] = 'e'; fln[9] = 'n'; fln[10] = '_';
                    fln[11] = 'w'; fln[12] = 'a'; fln[13] = 't'; fln[14] = 'e'; fln[15] = 'r';
                    fln[16] = '_'; fln[17] = 't'; fln[18] = 'a'; fln[19] = 'i'; fln[20] = 'l';
                    fln[21] = 'r'; fln[22] = 'a'; fln[23] = 'c'; fln[24] = 'e'; fln[25] = '/';
                    n= 26;};
                if(ind == 4) {
                    fln[6] = 'i'; fln[7] = 'c'; fln[8] = 'e'; fln[9] = '_'; fln[10] = 'c';
                    fln[11] = 'o'; fln[12] = 'v'; fln[13] = 'e'; fln[14] = 'r'; fln[15] = '_';
                    fln[16] = 't'; fln[17] = 'a'; fln[18] = 'i'; fln[19] = 'l'; fln[20] = 'r';
                    fln[21] = 'a'; fln[22] = 'c'; fln[23] = 'e'; fln[24] = '/'; n = 25;};
```

```
                count = 0;

                while(count < name_length){
                    fln[n + count] = name[count];
                    ++count;};

                n = n + name_length;
                fln[n] = 0;
                graph(fln);
                goto again; };

        ind = 1;
        out:
        return ind;
}
```

```
/* ------------------------------------------------------------------------* /
/* Viewing inflow related data */
view_inflow()
{
extern int pickinflow();
extern short viewport_id1;

char                    fln[80];
status_$t               st;
int                     ind,n,typ,count,scr,mn,ind1;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;
gm_$string_t            name;
short                   name_length,n_short;
gm_$point16_t           pt;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
float                   hi;

/* Initialize string */
      fln[0] = 'i'; fln[1] = 'n'; fln[2] = 'f'; fln[3] = 'l'; fln[4] = 'o'; fln[5] = 'w';
      fln[6] = 's'; fln[7] = '/';

/* pick inflow for deleting */
      ind = pickinflow();

/* If picked successfully */
      if ( ind == 1 ) {
                  gm_$pick_command(gm_$step,st);
                  gm_$inq_command_type(command_type,data_type,st);
                  if(command_type == gm_$ttag) {
                              gm_$inq_tag(name,name_length,st); }; }
      else { ind = 0;
                  goto out;};
      gm_$segment_close(true,st);

/* Open a temporary segment for the picture */

again:
    gm_$viewport_select(viewport_id1,st);
    gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
    bounds.xmin = -150.0; bounds.xmax = 900.0;
    bounds.ymin = -100.0; bounds.ymax = 900.0;
    pt.x = 20; pt.y = 850;
    hi = 50.0;
    gm_$text_size(hi,st);
```

```
/* Put all that i snecessary in it */
    gm_$text_2d16(pt,0.0,name,name_length,st);
    pt.x = -100;
    gm_$text_2d16(pt,0.0,"INFLOW:",(short)7,st);

/* Display the whole thing */
    gm_$segment_close(true,st);
    gm_$display_segment_part(segment_id,bounds,st);
    gm_$segment_open(segment_id,st);
    gm_$segment_delete(st);
    scr = 0; mn = 12;
    ind = select_from_menu(mn);

move_gr:  if(ind != 1) {
                  if(ind == 2) {
                      fln[8] = 'h'; fln[9] = 'i'; fln[10] = 's'; fln[11] = 't';
                      fln[12] = 'o'; fln[13] = 'r'; fln[14] = 'i'; fln[15] = 'c';
                      fln[16] = '_'; fln[17] = 'f'; fln[18] = 'l'; fln[19] = 'o';
                      fln[20] = 'w'; fln[21] = 's'; fln[22] = '/'; n = 23; };
                  if(ind == 3) {
                      fln[8] = 'E'; fln[9] = 'M'; fln[10] = 'M'; fln[11] = 'A';
                      fln[12] = '_'; fln[13] = 'f'; fln[14] = 'l'; fln[15] = 'o';
                      fln[16] = 'w'; fln[17] = 's'; fln[18] = '/'; n = 19;};
                      count = 0;
                      while(count < name_length){
                              fln[n + count] = name[count];
                              ++count;};
                      n = n + name_length;
                      fln[n] = 0;
                      time_graph(scr,fln);
                      mn = 9;
                      ind1 = select_from_menu(mn);
                      if( ind1 != 1 ){
                            if( ind1 == 2) scr = scr + 1;
                            if( ind1 == 3) scr = scr -1;
                            if(scr < 0) scr = 0;
                            goto move_gr;};

                      goto again; };
ind = 1;
out:
          return ind;
}
```

```
/* ------------------------------------------------------------------------ */
 /* Viewing lake related data */
view_lake()
{
extern int picklake();
extern short viewport_id1;

FILE                    *gf;
char                    fln[80];
status_$t               st;
int                     ind,n,typ,count,scr,mn,ind1;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;
gm_$string_t            name;
short                   name_length,n_short;
gm_$point16_t           pt;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
float                   hi;

/* Initialize string */
      fln[0] = 'l'; fln[1] = 'a'; fln[2] = 'k'; fln[3] = 'e'; fln[4] = 's'; fln[5] = '/';

/* pick lake for deleting */
      ind = picklake();

/* If picked successfully - delete it */
      if ( ind == 1 ) {
                  gm_$pick_command(gm_$step,st);
                  gm_$inq_command_type(command_type,data_type,st);
                  if(command_type == gm_$ttag) {
                                    gm_$inq_tag(name,name_length,st); };
      } else {
               ind = 0; goto out;};
      gm_$segment_close(true,st);

/* Open a temporary segment for the picture */
     gm_$viewport_select(viewport_id1,st);
     gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
     bounds.xmin = -150.0; bounds.xmax = 900.0;
     bounds.ymin = -100.0; bounds.ymax = 900.0;

     pt.x = 20; pt.y = 850;
     hi = 50.0;
     gm_$text_size(hi,st);
```

```
/* Put all that i snecessary in it */
     gm_$text_2d16(pt,0.0,name,name_length,st);
     pt.x = -100;
     gm_$text_2d16(pt,0.0,"LAKE:",(short)5,st);

/* Display the whole thing */
     gm_$segment_close(true,st);
     gm_$display_segment_part(segment_id,bounds,st);
     gm_$segment_open(segment_id,st);
     gm_$segment_delete(st);
     scr = 0;
again:
     mn = 10;
     ind = select_from_menu(mn);
move_gr:  if(ind != 1) {
               if(ind == 2) {
                    fln[6] = 's'; fln[7] = 't'; fln[8] = 'a'; fln[9] = 'g'; fln[10] = 'e';
                    fln[11] = 's'; fln[12] = 't'; fln[13] = 'o'; fln[14] = 'r';
                    fln[15] = 'a'; fln[16] = 'g'; fln[17] = 'e'; fln[18] = 'c';
                     fln[19] = 'u'; fln[20] = 'r'; fln[21] = 'v'; fln[22] = 'e';
                    fln[23] = '/'; n = 24;};
               if(ind == 3) {
                    fln[6] = 's'; fln[7] = 't'; fln[8] = 'o'; fln[9] = 'r'; fln[10] = 'a';
                    fln[11] = 'g'; fln[12] = 'e'; fln[13] = 'v'; fln[14] = 'a';
                    fln[15] = 'l'; fln[16] = 'u'; fln[17] = 'e'; fln[18] = '/'; n = 19;};
               if(ind == 4) {
                    fln[6] = 'h'; fln[7] = 'i'; fln[8] = 's'; fln[9] = 't'; fln[10] = 'o';
                    fln[11] = 'r'; fln[12] = 'i'; fln[13] = 'c'; fln[14] = '_';
                    fln[15] = 'l'; fln[16] = 'e'; fln[17] = 'v'; fln[18] = 'e';
                    fln[19] = 'l'; fln[20] = 's'; fln[21] = '/'; n = 22;};
              count = 0;
              while(count < name_length){ fln[n + count] = name[count];
                                          ++count;};
              n = n + name_length;
              fln[n] = 0;
               if(ind != 4) {graph(fln);}
               else  { time_graph(scr,fln);
                     mn = 9; ind1select_from_menu(mn);
                     if( ind1 != 1 ){ if( ind1 == 2) scr = scr + 1;
                                      if( ind1 == 3) scr = scr -1;
                                      if(scr < 0) scr = 0;
                                      goto move_gr;};
                     };
                goto again;};
                 ind = 1;
                 out: return ind; }
```

```
/*-------------------------------------------------------------------------*/
/* I shall make the view menu segment in the metafile */
int view_menu()
{
gm_$segment_id_t segment_id;
status_$t st;
gm_$point16_t pt1;
int seg;

/* Create view_menu segment */
      gm_$segment_create("VIEW",
                                    (short)4, segment_id, st);

/* Get the right text size and background color. */
      gm_$text_size(26.0,st);
      gm_$text_background_value((long)18,st);

/* Red box prompt */
    message("CLICK ON BLUE BOX","TO CONTINUE");

/* Make 3 blue menu box */
      menu_boxes(3);

/* Enter commands */
      gm_$text_background_value((long)18,st);
      gm_$text_value((long)20,st);

      pt1.x = 1;
      pt1.y = -10;
/* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"QUIT VIEWING",(short)12,st);

      pt1.y = pt1.y - 12;
      /* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"SCROLL RIGHT",(short)12,st);

      pt1.y = pt1.y - 12;
      /* define place for text, and write command */
      gm_$text_2d16(pt1,0.0,"SCROLL LEFT",(short)11,st);

      seg = segment_id;
      return seg;
}
```

```
/*-------------------------------------------------------------------------*/
/* I shall let you pick an element from the scheme and show you related data */
view_mode()
{
extern int view_lake(), view_hydro(), view_inflow(), view_outlet(),
           view_control(), view_thermal();
extern gm_$segment_id_t segment_id, s1, s2, s3, s4, s5, s6;

/* Declare all the variables */
    gm_$segment_id_t        sg,sg1;
    long                    n_instances;
    gm_$boundsreal_t        bounds;
    status_$t               st;
    int                     ind;

/* Go to the main window */
    please_draw_scheme();

/* Pick ! */
    pick_something(0.5,0.5);
    ind = view_lake();
    if( ind != 1 ) {

        /* Pick the right segment */
            gm_$pick_segment(gm_$clear,sg,n_instances,bounds,st);
            gm_$pick_segment(gm_$setup,sg,n_instances,bounds,st);
            gm_$pick_segment(gm_$down,sg,n_instances,bounds,st);

        /* open the segment */
            doit:
            if( sg == s2 )  ind = view_hydro();
            if( sg == s3 )  ind = view_inflow();
            if( sg == s4 )  ind = view_outlet();
            if( sg == s5 )  ind = view_control();
            if( sg == s6 )  ind = view_thermal();
            if( ind != 1 ) {
            gm_$pick_segment(gm_$next,sg,n_instances,bounds,st);
                                if (st.all == status_$ok) goto doit;}
                };
                gm_$pick_segment(gm_$clear,sg,n_instances,bounds,st);
}
```

```
/* ------------------------------------------------------------------------* /
/* Viewing outlet related data */
view_outlet()
{
extern int pickoutlet();
extern short viewport_id1;

char                    fln[80];
status_$t               st;
int                     ind,n,typ,count,scr,mn,ind1;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;
gm_$string_t            name;
short                   name_length,n_short;
gm_$point16_t           pt;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
float                   hi;

/* Initialize string */
      fln[0] = 'o'; fln[1] = 'u'; fln[2] = 't'; fln[3] = 'l'; fln[4] = 'e';
      fln[5] = 't'; fln[6] = 's'; fln[7] = '/';

/* pick outlet */
      ind = pickoutlet();

/* If picked successfully - delete it */
        if ( ind == 1 ) {gm_$pick_command(gm_$step,st);
                         gm_$inq_command_type(command_type,data_type,st);
                         if(command_type == gm_$ttag) {
                                  gm_$inq_tag(name,name_length,st); };
      } else {    ind = 0;
                  goto out;};
      gm_$segment_close(true,st);

/* Open a temporary segment for the picture */
     gm_$viewport_select(viewport_id1,st);
     gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
     bounds.xmin = -150.0;
     bounds.xmax = 900.0;
     bounds.ymin = -100.0;
     bounds.ymax = 900.0;
     pt.x = 20;
      pt.y = 840;
     hi = 50.0;
     gm_$text_size(hi,st);
```

```
/* Put all that i snecessary in it */
    gm_$text_2d16(pt,0.0,name,name_length,st); pt.y = 870;
    gm_$text_2d16(pt,0.0,"NATURAL OUTLET:",(short)15,st);

/* Display the whole thing */
    gm_$segment_close(true,st);
    gm_$display_segment_part(segment_id,bounds,st);
    gm_$segment_open(segment_id,st);
    gm_$segment_delete(st);
scr = 0;

again:
    mn = 15;
    ind = select_from_menu(mn);

move_gr:  if(ind != 1) {
                        if(ind == 2) {
                                fln[8] = 'h'; fln[9] = 'i'; fln[10] = 's'; fln[11] = 't';
                                fln[12] = 'o'; fln[13] = 'r'; fln[14] = 'i'; fln[15] = 'c';
                                fln[16] = '_'; fln[17] = 'f'; fln[18] = 'l'; fln[19] = 'o';
                                fln[20] = 'w'; fln[21] = 's'; fln[22] = '/'; n = 23;};
                        if(ind == 3) {
                                fln[8] = 's'; fln[9] = 't'; fln[10] = 'a'; fln[11] = 'g';
                                fln[12] = 'e'; fln[13] = '_'; fln[14] = 'd'; fln[15] = 'i';
                                fln[16] = 's'; fln[17] = 'c'; fln[18] = 'h'; fln[19] = 'a';
                                fln[20] = 'r'; fln[21] = 'g'; fln[22] = 'e';
                                fln[23] = '/'; n = 24;};
                        count = 0;
                        while(count < name_length){
                                fln[n + count] = name[count];
                                ++count;};

                        n = n + name_length;
                        fln[n] = 0;
                        if( ind == 3) graph(fln);
                        if( ind == 2) { time_graph(scr,fln);
                                        mn = 9;
                                        ind1 = select_from_menu(mn);
                                        if( ind1 != 1 ){
                                                if( ind1 == 2) scr = scr + 1;
                                                if( ind1 == 3) scr = scr -1;
                                                if(scr < 0) scr = 0;
                                                goto move_gr;};
                                      };
                        goto again; };
```

```
ind = 1;

out:
        return ind;

}
```

```
/* ------------------------------------------------------------------------* /
/* Viewing thermal station related data */
view_thermal()
{
extern int pickthermal();
extern short viewport_id1;

FILE                    *gf;
char                    fln[80];
status_$t               st;
int                     ind,n,typ,count,scr,mn,ind1;
gm_$segment_id_t        segment_id;
gm_$boundsreal_t        bounds;
gm_$string_t            name;
short                   name_length,n_short;
gm_$point16_t           pt;
gm_$command_type_t      command_type;
gm_$data_type_t         data_type;
float                   hi;

/* Initialize string */
      fln[0] = 't'; fln[1] = 'h'; fln[2] = 'e'; fln[3] = 'r'; fln[4] = 'm'; fln[5] = 'o';
      fln[6] = '/';

/* pick thermal for deleting */
      ind = pickthermal();

/* If picked successfully  */
      if ( ind == 1 ) {
                  gm_$pick_command(gm_$step,st);
                  gm_$inq_command_type(command_type,data_type,st);
                  if(command_type == gm_$ttag) {
                                                gm_$inq_tag(name,name_length,st);
                                                      };
                        }
               else { ind = 0;
                     goto out;};
      gm_$segment_close(true,st);

/* Open a temporary segment for the picture */
     gm_$viewport_select(viewport_id1,st);
     gm_$segment_create("TEMPORARY",(short)9,segment_id,st);
     bounds.xmin = -150.0; bounds.xmax = 900.0;
     bounds.ymin = -100.0; bounds.ymax = 900.0;
     pt.x = 20; pt.y = 840; hi = 50.0; gm_$text_size(hi,st);
```

```
/* Put all that i snecessary in it */
    gm_$text_2d16(pt,0.0,name,name_length,st);
    pt.y = 870;
    gm_$text_2d16(pt,0.0,"THERMAL STATION:",(short)16,st);

/* Display the whole thing */
    gm_$segment_close(true,st);
    gm_$display_segment_part(segment_id,bounds,st);
    gm_$segment_open(segment_id,st);
    gm_$segment_delete(st);
scr = 0;

again:
    mn = 13;
    ind = select_from_menu(mn);
    if(ind != 1) {
            fln[7] = 's';
            n = 8;
            count = 0;
            while(count < name_length){
                    fln[n + count] = name[count];
                    ++count;};
            n = n + name_length; fln[n] = 0;
            graph(fln);
            goto again; };

    ind = 1;
out:
    return ind;
}
```