THE UNIVERSITY OF MANITOBA

A PROTOTYPE DECISION SUPPORT SYSTEM FOR THE DEVELOPMENT OF STAGE-DISCHARGE RATING CURVES

by

Glen G. Douglas

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree of Master of Science

DEPARTMENT OF CIVIL ENGINEERING

Winnipeg, Manitoba

ISBN 0-315-81730-5

Name ____________________ ____________ ____________________

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

____ | 0 | 5 | 4 | 5 | U·M·I

SUBJECT TERM — SUBJECT CODE

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture	0729
Art History	0377
Cinema	0900
Dance	0378
Fine Arts	0357
Information Science	0723
Journalism	0391
Library Science	0399
Mass Communications	0708
Music	0413
Speech Communication	0459
Theater	0465

EDUCATION

General	0515
Administration	0514
Adult and Continuing	0516
Agricultural	0517
Art	0273
Bilingual and Multicultural	0282
Business	0688
Community College	0275
Curriculum and Instruction	0727
Early Childhood	0518
Elementary	0524
Finance	0277
Guidance and Counseling	0519
Health	0680
Higher	0745
History of	0520
Home Economics	0278
Industrial	0521
Language and Literature	0279
Mathematics	0280
Music	0522
Philosophy of	0998
Physical	0523
Psychology	0525
Reading	0535
Religious	0527
Sciences	0714
Secondary	0533
Social Sciences	0534
Sociology of	0340
Special	0529
Teacher Training	0530
Technology	0710
Tests and Measurements	0288
Vocational	0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language	
General	0679
Ancient	0289
Linguistics	0290
Modern	0291
Literature	
General	0401
Classical	0294
Comparative	0295
Medieval	0297
Modern	0298
African	0316
American	0591
Asian	0305
Canadian (English)	0352
Canadian (French)	0355
English	0593
Germanic	0311
Latin American	0312
Middle Eastern	0315
Romance	0313
Slavic and East European	0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy	0422
Religion	
General	0318
Biblical Studies	0321
Clergy	0319
History of	0320
Philosophy of	0322
Theology	0469

SOCIAL SCIENCES

American Studies	0323
Anthropology	
Archaeology	0324
Cultural	0326
Physical	0327
Business Administration	
General	0310
Accounting	0272
Banking	0770
Management	0454
Marketing	0338
Canadian Studies	0385
Economics	
General	0501
Agricultural	0503
Commerce-Business	0505
Finance	0508
History	0509
Labor	0510
Theory	0511
Folklore	0358
Geography	0366
Gerontology	0351
History	
General	0578
Ancient	0579
Medieval	0581
Modern	0582
Black	0328
African	0331
Asia, Australia and Oceania	0332
Canadian	0334
European	0335
Latin American	0336
Middle Eastern	0333
United States	0337
History of Science	0585
Law	0398
Political Science	
General	0615
International Law and Relations	0616
Public Administration	0617
Recreation	0814
Social Work	0452
Sociology	
General	0626
Criminology and Penology	0627
Demography	0938
Ethnic and Racial Studies	0631
Individual and Family Studies	0628
Industrial and Labor Relations	0629
Public and Social Welfare	0630
Social Structure and Development	0700
Theory and Methods	0344
Transportation	0709
Urban and Regional Planning	0999
Women's Studies	0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture	
General	0473
Agronomy	0285
Animal Culture and Nutrition	0475
Animal Pathology	0476
Food Science and Technology	0359
Forestry and Wildlife	0478
Plant Culture	0479
Plant Pathology	0480
Plant Physiology	0817
Range Management	0777
Wood Technology	0746
Biology	
General	0306
Anatomy	0287
Biostatistics	0308
Botany	0309
Cell	0379
Ecology	0329
Entomology	0353
Genetics	0369
Limnology	0793
Microbiology	0410
Molecular	0307
Neuroscience	0317
Oceanography	0416
Physiology	0433
Radiation	0821
Veterinary Science	0778
Zoology	0472
Biophysics	
General	0786
Medical	0760

EARTH SCIENCES

Biogeochemistry	0425
Geochemistry	0996
Geodesy	0370
Geology	0372
Geophysics	0373
Hydrology	0388
Mineralogy	0411
Paleobotany	0345
Paleoecology	0426
Paleontology	0418
Paleozoology	0985
Palynology	0427
Physical Geography	0368
Physical Oceanography	0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences	0768
Health Sciences	
General	0566
Audiology	0300
Chemotherapy	0992
Dentistry	0567
Education	0350
Hospital Management	0769
Human Development	0758
Immunology	0982
Medicine and Surgery	0564
Mental Health	0347
Nursing	0569
Nutrition	0570
Obstetrics and Gynecology	0380
Occupational Health and Therapy	0354
Ophthalmology	0381
Pathology	0571
Pharmacology	0419
Pharmacy	0572
Physical Therapy	0382
Public Health	0573
Radiology	0574
Recreation	0575
Speech Pathology	0460
Toxicology	0383
Home Economics	0386

PHYSICAL SCIENCES

Pure Sciences

Chemistry	
General	0485
Agricultural	0749
Analytical	0486
Biochemistry	0487
Inorganic	0488
Nuclear	0738
Organic	0490
Pharmaceutical	0491
Physical	0494
Polymer	0495
Radiation	0754
Mathematics	0405
Physics	
General	0605
Acoustics	0986
Astronomy and Astrophysics	0606
Atmospheric Science	0608
Atomic	0748
Electronics and Electricity	0607
Elementary Particles and High Energy	0798
Fluid and Plasma	0759
Molecular	0609
Nuclear	0610
Optics	0752
Radiation	0756
Solid State	0611
Statistics	0463

Applied Sciences

Applied Mechanics	0346
Computer Science	0984
Engineering	
General	0537
Aerospace	0538
Agricultural	0539
Automotive	0540
Biomedical	0541
Chemical	0542
Civil	0543
Electronics and Electrical	0544
Heat and Thermodynamics	0348
Hydraulic	0545
Industrial	0546
Marine	0547
Materials Science	0794
Mechanical	0548
Metallurgy	0743
Mining	0551
Nuclear	0552
Packaging	0549
Petroleum	0765
Sanitary and Municipal	0554
System Science	0790
Geotechnology	0428
Operations Research	0796
Plastics Technology	0795
Textile Technology	0994

PSYCHOLOGY

General	0621
Behavioral	0384
Clinical	0622
Developmental	0620
Experimental	0623
Industrial	0624
Personality	0625
Physiological	0989
Psychobiology	0349
Psychometrics	0632
Social	0451

A PROTOTYPE DECISION SUPPORT SYSTEM FOR THE DEVELOPMENT OF STAGE-DISCHARGE RATING CURVES

BY

GLEN G. DOUGLAS

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

MASTER OF SCIENCE

© 1993

ABSTRACT

This work presents the development of a prototype computer-based decision support system for creating and maintaining stage-discharge rating curves. Stage-discharge analysis deals with the formulation of relationships relating stage and discharge, for a given flowing body of water. The intent of the prototype system is to experiment with various computer technologies and explore the problem domain of stage-discharge analysis, prior to the development of a fully operational system, for the Department of Environment in Canada.

The difficulties associated with creating relationships between stage and discharge in natural streams is presented, with respect to section and channel controls, and physical channel characteristics. Computer decision support technology and mathematical modelling is introduced and applied to the problem of stage-discharge analysis. Graphical user interface concepts are built into the window environment of the prototype, to facilitate simple, yet efficient, user input and enhance user comprehension of system information.

The prototype design was restricted to application in stable channels only, to allow design emphasis to be placed on developing a logical framework, without expending limited development time on details. The framework structure allows for easy expansion to include elaboration into non-stable channel analysis and more sophisticated and complex modelling techniques. Recommendations and are made for further developments and additions to the prototype, as well as for a fully operational decision support system.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to several individuals, for their various contributions throughout my studies. First, my thanks to regional offices of the Department of Environment (DOE) for their financial contribution to this project and their commitment to its continuation in the future. I am very grateful for the technical guidance and support provided by Klaus Wiebe and Dr. Henry Hudson, of the Ottawa and Winnipeg Branches of the DOE, respectively.

Next, my sincere thanks to my advisor, Dr. Slobodan Simonovic, for his advice, guidance and encouragement throughout my studies. I feel very fortunate to have been invited, by Dr. Simonovic, to participate in this project, as well as in the early stages of this promising venture, undertaken by the DOE and the University of Manitoba.

I would also like to thank my colleagues and friends in the FIDS lab (Facility for Intelligent Decision Support in Water Resources Engineering), for their camaraderie and encouragement during our many long hours of research and study.

To my parents, I am truly grateful for their years of continuous support and confidence, which without, I could not have achieved my accomplishments.

Finally, to my new wife Jennifer, I thank you for the strength and inspiration you have given me, over our long distance separations, during my studies. I will always be indebted to you for your patience and understanding, which have been invaluable to the completion of my work, and more importantly, the success of our relationship. Thank you.

TABLE OF CONTENTS

	Page
ABSTRACT	i
ACKNOWLEDGEMENTS	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	viii
LIST OF ABBREVIATIONS	x
1. INTRODUCTION	1
1.1 Problem Statement	2
1.2 Scope of Research	7
2. LITERATURE REVIEW	10
2.1 Introduction	10
2.1.1 Procedural Standards	10
2.1.2 Recent Work	13
2.1.3 Computer Software	15
2.2 Decision Support Systems in Water Resources	17

TABLE OF CONTENTS

3.	**STAGE-DISCHARGE ANALYSIS**	**19**
3.1	Introduction	19
3.1.1	Hydrometric Organizations in Canada	19
3.2	Current Practice in Canada	20
3.2.1	Concepts	20
3.2.2	Collecting Stage-Discharge Data	21
3.2.3	Establishing a Rating Curve	22
3.3	Stage-Discharge Relationship Controls	24
3.3.1	Types of Controls	24
3.3.2	Effects of Physical Characteristics on Controls	27
3.4	Stable and Unstable Channels	32
3.4.1	Stable Channels	35
3.4.2	Unstable Channels	37
4.	**COMPUTERIZED DECISION SUPPORT TECHNOLOGY**	**39**
4.1	The Evolution of Decision Support Systems	39
4.1.1	Early Information Systems	39
4.1.2	The Essence of Decision Support Systems	40
4.1.3	DSS Architectures and Components	41

TABLE OF CONTENTS

4.2 Intelligent DSS and Expert Systems Technology 43

4.2.1 A Brief Background of Expert Systems 43

4.2.2 The Architecture of Expert Systems 46

4.3 A Proposed Computer System 48

5. A PROTOTYPE DECISION SUPPORT SYSTEM FOR STAGE-DISCHARGE ANALYSIS 51

5.1 DSS and the Stage-Discharge Analysis Process 51

5.2 Design Framework 53

5.2.1 Analysis Concept 53

5.2.2 Operational User Modes 58

5.2.3 Modelling and Evaluation 61

5.3 Technology 65

5.3.1 Operating Platform 66

5.3.2 System Control 69

5.3.3 Database Management 72

5.3.4 User Interface 73

5.3.5 Modelling 74

TABLE OF CONTENTS

5.4 Operational Structure 76

5.4.1 Main Menu 77

5.4.2 Curve Development 78

5.4.3 Curve Use 87

5.4.4 Curve Modification 89

6. CASE STUDY - APPLICATION OF SDDSS TO EDWARDS CREEK 94

6.1 The Study Station 94

6.2 Development of a Rating Curve 97

6.2.1 Selecting Measurements 97

6.2.2 Modelling Selected Data 101

6.2.3 Evaluating The Models 104

6.2.4 Analyzing Outliers 107

6.2.5 Storing A Curve 111

6.3 Using a Rating Curve 111

6.4 Examination of Results 116

7. CONCLUSIONS AND FUTURE RESEARCH 121

7.1 Evaluation of SDDSS Prototype 121

7.2 Future Research Work 122

TABLE OF CONTENTS

REFERENCES **127**

APPENDIX A - Unix Scripts **134**

LRM (Linear Regression Model) 135

GC (Generate Curve) 136

XGR (Xgraph Run) 137

APPENDIX B - Fortran Programs **138**

LR2 (Linear Regression 2) 139

GCRV (Generate Curve) 143

APPENDIX C - Nexpert Object Knowledge Bases **145**

A0_main.tkb (Definitions) 146

A1_main.tkb (System Operations) 160

A2_main.tkb (Main Menu & Curve Use Modules) 165

A3_main.tkb (Curve Development Module) 179

A4_main.tkb (Outlier Analysis Module) 232

A5_main.tkb (Curve Modification Module) 255

LIST OF FIGURES

Figure 1. Specific Gauge Plots for Edwards Creek.

Figure 2. Structure of IDSS.

Figure 3. SDDSS system architecture.

Figure 4. Hierarchical Data Selection.

Figure 5. SDDSS Workbench.

Figure 6. Nexpert Object organization of SDDSS.

Figure 7. SDDSS Curve Development Process.

Figure 8. SDDSS Curve Use process.

Figure 9. Sample flow table.

Figure 10. SDDSS Curve Modification process.

Figure 11. Example station list.

Figure 12. (a) Map of Edwards Creek area, (b) Sketch of Edwards Creek gauging site.

Figure 13. All measurements selected (phase 1) for Edwards Creek.

Figure 14. Specific measurements selected (phase 2), May to October observations.

Figure 15. Summary of results from Phase 2 selection procedure.

Figure 16. Summary of SDDSS results for 3 models.

Figure 17. Model ML2, shown with May to October measurements.

Figure 18. Session Control Window displaying options for model evaluation.

Figure 19. Dialogue window displaying outlier information.

Figure 20. Example questions used for analysis of outlier deviation.

Figure 21. Summary of outlier analysis, showing causes of deviations.

Figure 22. Final stored curve, with outliers omitted from modelling.

Figure 23. Sample water level record.

Figure 24. List of curves created for Edwards Creek.

Figure 25. Example of computed flow table.

Figure 26. Comparison of SDDSS and hand rendered (WSC-90) curves.

Figure 27. Comparison of curves, low flow regime.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CD	Curve Development
CM	Curve Modification
CU	Curve Use
DOE	Department of Environment (Canada)
DSS	Decision Support System
EC	Environment Canada
EES	Engineering Expert System
ES	Expert System
GUI	Graphical User Interface
IDSS	Intelligent Decision Support System
ISO	International Organization of Standards
IWD	Inland Waters Directorate
KB	Knowledge-base
KBES	Knowledge-base Expert System
LRM	Linear Regression Model
NN	Neural Network
NOO	Number of Outliers
OA	Outlier Analysis
PDP	Parallel Distributed Processing

RDBMS	Relational Database Management System
S-D	Stage-Discharge
SCM	System Control Manager
SDA	Stage-Discharge Analysis
SDDSS	Stage-Discharge Decision Support System
SEE	Standard Error of Estimate
SFMAS	Streamflow Measurement Advisory System
SQL	Structured Query Language
USGS	United States Geological Survey
WRB	Water Resources Branch
WSC	Water Survey of Canada

A PROTOTYPE DECISION SUPPORT SYSTEM FOR THE DEVELOPMENT OF STAGE-DISCHARGE RATING CURVES

1. Introduction

Rivers and streams are, and always has been, one of mankind's most valuable and essential natural resources. The first attempts at understanding the nature of open channel flow originated in ancient Egypt, Mesopotamia and India for the purpose of irrigation. However, it was not until the seventeenth and eighteenth centuries that fundamental principles of hydrodynamics were greatly developed by Sir Isaac Newton, Daniel Bernoulli, and Leonhard Euler (Robertson and Crowe, 1985).

Over the course of history, streamflow itself has had many uses. For centuries, civilizations all around the world have depended on streamflow to provide continuous supplies of water for daily use and irrigation of their crops, as well as a means for transportation. Today, these basic uses still continue to be primary necessities of our everyday life. Large scale commercial and industrial processing has also grown to depend on streamflow, for dilution of waste materials and cooling of thermal power stations, to name but a few uses. Finally, countries throughout the world have been using the massive potential energy of streamflow for the production of hydroelectric power.

For these and other reasons, streamflow can be seen as an essential element in civilian and commercial developments in Canada, as well as around the world. Therefore, the collection and processing of streamflow data for the production of streamflow records is of paramount importance in meeting the needs of the many potential users.

1.1 Problem Statement

The collection of hydrometric data in Canadian rivers, streams and lakes originated more than 80 years ago. Today, over 3400 gauging stations (Environment Canada, 1992) are in operation throughout the 10 provinces and 2 northern territories. Typical types of data collected on a regular basis by water survey technicians includes water surface elevations, streamflows and sediment concentrations. Some stations are also equipped to collect atmospheric and water quality data.

Since it is impractical to use continuous discharge measurement methods, due to the high cost and limited use of the equipment, discharge data is generally compiled indirectly. Corresponding water level and discharge measurements are used to establish graphical relationships known as *Stage-Discharge Curves*, or *S-D Curves*. (These are also commonly referred to as *Stage-Discharge ratings, rating curves*, *relationships* or *relations*). Rating curves are then used in conjunction with continuous water level records to generate desired discharge records.

The preparation, analysis and representation of S-D relationships can be a very complex and involved process. To begin, the process of collecting and compiling data into a usable form requires a great amount of time, effort and expertise of water survey technicians. Knowledge of section and channel controls influencing S-D relationships, is also an essential part of the data collection and analysis process. Further, establishing relationships requires experience and intuition from the technician. Data analysis and curve fitting involves repeated subjective decision-making based on human judgement, experience and rules-of-thumb. Finally, the nature of changing stream conditions requires relationships to be continuously updated. Consequently, the processes involved in establishing rating curves must be repeated frequently, to ensure the relationship is accurately represented.

Thus, it becomes apparent that the development of rating curves is a problem of "semi-structured" nature; that is a procedure that consists of a number of sequential tasks, conditional on subjective input, based on a technicians' experience, knowledge and intuition. In response to this problem, it has been suggested that new computer technologies and techniques may be used to simulate (or model) the problem in one single framework. Understanding these processes provides the basis for developing a computer system to create, update and employ S-D relationships, for the generation of streamflow records.

The ***main objective*** of this study, therefore, was to develop a computer decision support system for the development, maintenance and use of S-D relationships. The system would explore preliminary design considerations and concepts to be used in the development of a "full blown" operational system. It is the intention that the development of this system is to serve as one small component of a much larger modular computer support system, for surface water quantity data management use by the *Department of Environment* (DOE), which has been proposed by Simonovic (1989).

Emphasis on the design of the computer system is directed at creating a "logical framework", incorporating all essential process elements. A modular architecture is intended to allow for the future additions and modifications. Technical decision support is to be provided using *Knowledge Base* (KB) and *Expert System* (ES) technology, capturing the expertise and knowledge of practitioners. Strategies and practices currently used by the DOE in Canada are to be studied and used as a conceptual model for the system structure.

In brief, a *decision support system* (DSS) is a computer program used to assist decision-makers in the analysis of problems and alternate solutions. Problem types have been categorized by Simonovic (1992) according to the structure of the elements, (or lack there of) which define the problem environment and solution space. A *well-structured* problem consists of elements which can be clearly identified and quantified in order to reach a solution. These types of problems are generally solvable using mathematical and/or statistical models. *Unstructured* problems are those which contain no quantifiable elements, thus making modelling practices non-applicable. Human judgement and experience is required to assess the relevant information, in order to reach a solution. A *semi-structured* problem is a relative term, bridging the gap between the two previous classifications. These problems contain some structure of pattern, but still rely on human input to assist in the solution process, (Simonovic, 1992).

Although decision support systems are used to assist in the solution of all three types of problems, they are particularly well-suited for application to problems requiring some form of human judgement. They typically employ various computer modelling routines and incorporate human interaction to include intuition and judgement in the solution process.

Decision support systems are generally understood to consist of a collection of discrete components, each responsible for a specific task in the problem-solving process.

These components typically include database management, mathematical (statistical) modelling, computer graphics and knowledge representation. Typically, these components are controlled by one central unit, which directs and monitors their execution and oversees the processing of the system.

The design of the system presented in this work builds on this concept of DSS. The structure of the DSS is segmented into discrete components, each addressing a specific task in the overall S-D analysis process.

Processing control and knowledge representation of the DSS is handled using an Expert System development tool or "shell". These shells are specifically designed to provide developers with a useful environment for creating applications to perform real-life problem-solving and reasoning tasks, involving subjective human judgement and intuition. Many ES shells are equipped with graphical development environments which enable application developers to formulate applications in a fast and efficient manner, without having to learn new programming languages. Some software shells are also equipped with tools to provide the developer with the opportunity of building *graphical user interfaces* (GUI) in front of their applications.

Storage and retrieval of data is provided by a *Relational Database Management System* (RDBMS). Relational databases are widely accepted because of their ability to efficiently handle information. Many RDBMSs incorporate the industry standard *Structured Query Language* (SQL) for ease of data manipulation between the system and database.

Computer graphics form an essential role in any DSS. Their main purpose is to improve the level of comprehensive communication between the application and the user. The use of windowing environments can provide simultaneous displays of information, while maintaining a high level of organization. Applications can be made user-friendly by incorporating menu options and buttons for user input, and by presenting results in multi-colour graphic display formats. One goal of this research was therefore to explore the possible applications of computer graphics in the development of the prototype.

Modelling S-D relationships within the DSS was accomplished using statistical modelling techniques, such as regression analysis, programmed as Fortran routines. This language was selected because of its proven ability to provide fast computations and formatted output.

Finally, the issue of communication among the system components and user-interface was of great concern. For this reason, a workstation platform was selected for development of the system. Benefits of workstations over conventional desktop personal computers include greater processing speeds, advanced networking capability and better multi-task processing for window environments.

The concept of the "engineering expert systems approach" (Simonovic and Savic, 1989) was combined with the classical ES approach in the development of the system. The *engineering expert systems* (EES) approach involves training engineering experts of a specific domain in the use of ES techniques and tools. It has proven to be an efficient method for the development of decision support systems in water resources engineering (Simonovic and Savic, 1989). In contrast, the classical ES approach relies on transferring unique knowledge from an expert to a knowledge engineer; of which the latter is generally uneducated in the expert's domain. During the course of this research, emphasis was directed at understanding the concepts of both S-D analysis and ES technology.

1.2 Scope of Research

The scope of this research has dealt with the development of a ***logical framework of processes*** to analyze and represent S-D relations within a computer system. Due to the complex nature of alluvial channels, the system prototype developed in this work is limited to analysis of stable channels.

A *stable* channel is generally characterized by a relatively constant S-D relationship, which experiences infrequent and temporary changes. They generally have bed forms which do not change dramatically in short periods, but rather experience changes in other channel characteristics affecting channel roughness and/or geometry. Such characteristics make up the *hydraulic control* of a channel, which if altered, will be manifested in the stage-discharge relationship. Stable channels are therefore more likely to experience changes in their stage-discharge relationships due to changes in these other characteristics rather than due to erosion and deposition of their be material. It should be noted that the term "stable" is a relative one since all channels experience some degree of changes in their hydraulic control over long periods, be it associated with bed form, geometry or roughness.

In contrast, *non-stable* channels are those subject to continuous changes in their hydraulic control attributes, thus causing the stage-discharge relations to shift over short (as well as long) periods. They are alluvial channels which are continuously forming in the bed material which they transport. Changes in other channel characteristics are influential as in stable channels, however they are difficult to detect amongst the dramatic changes that occur in the stream due to movement and deposition of bed material.

Recent advancements and innovations in computer tools and technology are demonstrated for their usefulness and application to computer decision support. A brief outline of the thesis organization follows.

A review of literature and research related to the topic of stage-discharge analysis is presented in Chapter 2. Operational standards developed by various organizations over the years are discussed. The review also makes note of some recent applications of decision support systems in water resources engineering, in order to demonstrate the usefulness they have achieved over the past 10 years.

Chapter 3 presents the concepts of *Stage-Discharge Analysis*. Current practices and procedures used throughout Canada by the Department of the Environment, are discussed. The concepts of station controls are examined and their relative effects on S-D relationships in stable and unstable channels are discussed.

Chapter 4 deals with the concepts of decision support systems. A brief history of DSS and its evolution from early information systems is discussed. DSS architectures and components are covered, as well as an introduction to Expert Systems technology.

Chapter 5 presents the prototype DSS developed in this research work. The first sections are devoted to the application of DSS technology to Stage-Discharge Analysis and the subsequent development of the computer system presented in this work. The design framework is presented followed by a discussion of the operation of the system components.

Chapter 6 presents a case study involving the development of a stage-discharge relationship based on data obtained from Edwards Creek in southern Manitoba. A narrative is included which describes user input required during the curve development operation of the system. A curve produced using the DSS is compared to a rating curve prepared manually by the regional office of the Department of Environment in Winnipeg, Manitoba. The comparison is made with respect to the general curve shapes, examining the overall accuracy of relationship representation.

To conclude the research, Chapter 7 presents a summation of the research work accomplished in this study. Additional research and experimentation is expected to continue on the prototype in the near future. Therefore, further research is recommended and various additions are encouraged for later development.

2. LITERATURE REVIEW

2.1 Introduction

Stage-discharge relationships are used worldwide to provide a basis from which streamflow records are estimated. They are functional relationships developed between water level (stage) and the streamflow (discharge). These relationships are formulated empirically from data collected in a process termed "stream gauging", which consists of measuring discharge for various stages in a stream. Once formulated, stage-discharge relationships are used to convert continuous water level records into discharge estimates.

The science of stream gauging has evolved over many years. Widely accepted manuals of today, describing stream gauging and computational procedures, are the result of the collective experiences and innovations gained by practitioners over the years (Rantz and others, 1982a). Following is a synopsis of some of the most widely accepted material dealing with gauging procedures and the derivation of S-D relationships. Some recent research work is also included, along with an introduction to two computer software products developed for the analysis of hydrometric records.

2.1.1 Procedural Standards

Probably the most widely accepted methods for measuring and compiling streamflow data are standardized by the *International Organization for Standardization (ISO).* National institutions from around the world, concerned with the development of standards, are brought together to form ISO technical committees. These committees work on developing internationally accepted standards in widely diverse technological fields, such as science and engineering.

ISO Handbook 16 deals with the "Measurement Of Liquid Flow In Open Channels" (ISO, 1983). It is a collection of 29 International Standards prepared by the ISO/TC 113 committee in the 1970s and early 1980s. Individual reports, contained in this handbook, deal with flow measurement techniques and practices using various methods, equipment and structures under varying flow conditions.

In particular, ISO 1100/1 entitled "Liquid Flow Measurement In Open Channels - Part 1: Establishment and operation of a gauging station" (ISO, 1981) outlines the considerations and factors involved in the selection of a gauging site and its operation. The standards are applicable to gauging stage and/or discharge of open water in rivers, lakes, reservoirs or manmade channels. Unfortunately, gauging sites seldom fulfil the criteria laid out by ISO. For this reason, the analysis of stage-discharge relations is not always straight forward.

Standard methods for determining stage-discharge relationships are specified in "Part 2: Determination Of The Stage-Discharge Relation" (ISO, 1982). The report addresses key issues such as gauge station calibration, relations for stable and unstable channels, hysteresis and unsteady flow conditions, discharge estimation from curves, testing and uncertainty of relations and extrapolation for estimation. Computational procedures are also included, with examples.

Another widely accepted reference on stream gauging in the US and Canada is entitled "Measurement And Computation Of Streamflow" (Rantz and others, 1982a and

1982b), which is produced by the United States Department of the Interior. The two volume report was first published in 1980 as an up-to-date standardized manual of stream-gauging procedures. It was a long awaited replacement for the USGS Water Supply Paper 888 "Stream-Gauging Procedures" published in 1943.

Volume 1, "Measurement Of Stage And Discharge" (Rantz and others, 1982a) deals with the selection of stream-gauging sites and state-of-the-art procedures for collecting stage and discharge data. Discussions of instrumentation and measurements are directed at the field technician commonly performing such duties. Volume 2, "Computation of Discharge" (Rantz and others, 1982b) deals with the development of discharge ratings based on simple S-D relations and/or other hydraulic parameters, facilities and conditions. Computation of daily-discharge records based on derived ratings is also examined. Finally, the document discusses the presentation and publication of stream-gauging data.

In Canada, the Inlands Water Directorate (IWD), of the federal Department of Environment (DOE) has contributed to and adopted many of the standards outlined by ISO and the USGS Water Supply Papers. These standards make up the basis for the methods and procedures used by the federal and provincial Water Resources Branches, responsible for the monitoring of gauging stations throughout the provinces and northern territories of Canada. Over the years, the DOE has published numerous internal reports, related to particular practices used in Canada for collecting and compiling S-D data. A career development program has been implemented within the DOE for training its hydrometric staff. An extensive five volume series documents the entire training curriculum, to be instructed over a two to three year period. The fundamental purpose of the documents is to train technicians in the practices and procedures used within the DOE, and maintain a high level of standard and consistency throughout the organization.

The documents deal with basic concepts of stream gauging such as types of gauging stations, site selection and benchmark datums. Measurement of stage and discharge using various devices, instrumentation and methods is also discussed. Technicians are trained in the art of establishing stage-discharge relations and the computation of discharge records from continuous water level recordings. Fundamental hydrometric and hydraulic concepts are also covered in the program.

The basic understanding gained from review of this documentation, as well as the numerous publications of ISO and USGS revealed one underlying principle for this thesis work. The science of stream gauging and the analysis of stage-discharge relationships is nothing short of a complex system of activities, requiring technicians to build a strong knowledge of concepts and theory, and subsequently, years of field practice to develop their experience.

2.1.2 Recent Work

As old as the study of S-D relationships is, there continues to be interest in the understanding of relevant variables and parameters. Considerable hydraulic literature has been published over the years, outlining simple stage-discharge relationships in common use for open water, as well as more sophisticated relations for varying or specific conditions. In addition, recent technology has led to the development of new data collection equipment and methods since the earlier years of hydrometric data collection.

Following is a sample of two recent additions to the understanding of stage-discharge analysis.

A report by Chester (1986) introduces three practical techniques developed and implemented by the Water Authority of Western Australia "which significantly rationalize the practices for the development of discharge rating curves while still retaining the essence of traditional practice". Each technique provides an increased level of quality to the analysis and development of discharge ratings. In increasing order, the techniques include the use of a so-called natural/power$^{2/5}$ scale to linearize stage-discharge plots for simplifying the drawing of rating curves, a conveyance control parameter for extending rating curves beyond observed levels and the use of computerized hydraulic modelling programs for defining stage-discharge relations.

Comparisons of seven methods for extending rating curves are presented by Smith (1987). A case study involved the Illecillewaet River in British Columbia. Methods include extension by Log Transformation, dm-Q, Area/Velocity, A/dm, Stevens method, Slope-Conveyance and Stage-Discharge Equation. He concluded that "rating curve extension is as much an art as a science". In conclusion, the study recommends the use of the stage-discharge equation in performing rating curve extensions, where it is applicable. This decision is based on its ease of application and true representation with specific knowledge of the stream channel and cross sections and hydrographer interpretation. Further comparisons of the methods was recommended using other streams with available data.

Although the topic of curve extension is beyond the scope of this thesis, both publications demonstrate that there is an important need for increased understanding of stage-discharge relations. There is a great interest and need for techniques which improve the accuracy and quality of stage-discharge relations, throughout development and during use. The application of such techniques to a computer system, has been a main focus of this thesis work.

2.1.3 Computer Software

This section introduces two computer software packages that are operationally used for managing hydrologic and hydrometric data. Although these systems perform some similar functions of the system developed and presented in this research work, they were not specifically designed as decision support systems, but rather customized time-data management tools. Their inclusion here is to simply introduce the reader to some software products developed which share some related topics of stage-discharge analysis.

HYDSYS/TS was developed in Australia by HYDSYS Pty Limited. It is based on the ideas of EDTRACE, a system developed in the early 1960s by the Commonwealth Scientific and Industrial Research Organization (CSIRO). "HYDSYS is a system for the storage, editing, retrieval and analysis of time-series data and related information" (HYDSYS Pty Ltd, 1991), such as streamflow and meteorological parameters. Operating platforms are limited to PC (DOS) systems and a UNIX V version for use on Sun Microsystems workstations is also available, either as stand-alone or networked configurations. Time and non-time dependent data may be stored and analyzed, with time precision of one second. The system employs a relational database, a graphical mouse-based editor, comprehensive menu and full-screen interactive systems, to list but a few. Numerous graphical and tabular outputs are available and may be directed to a variety of supported devices. In addition, the software accepts direct input of raw data from a number of data loggers and satellite and radio telemetry systems. The New South Wales Water Resources Department, the Sydney Water Board and the Water Authority of Western Australia all contributed ideas and funds to develop areas of their particular interest.

Another computer system, TIDEDA, has been developed for the processing of time dependent data (Rodgers and Thompson, 1991). The New Zealand Ministry of Works and Development created the program with years of assistance from several individuals since the beginning in 1981. Data dealt with by the system includes hydrometric and climactic data such as stream flows, water levels, rainfall, air temperature and wind speed. Data associated with electrical power generation is also managed. Features of the system include data management, graphical and tabular display of data, user-specified data analysis and a flexible operating environment. The system also interfaces a number of data entry devices and is operational on IBM compatible personal computers or DEC-Vax computers.

Review of both systems has proved beneficial to the planning of the system developed in this thesis. Some important features of these systems which have been considered in the development of this work include the ability to model stage-discharge relations with a selection of mathematical functions, multi-domain plots used to display rating curve related information to the user, and the presentation of graphical user interface.

2.2 Decision Support Systems in Water Resources

The development and use of decision support systems (DSS) in water resources planning, engineering and management has been growing extensively over the past ten years. Recent attention has been attributed to their ability to assist engineers and managers with decision-making problems, which are of a less than structured nature. They allow decision-makers to thoroughly examine a problem and analyze alternate solutions. DSSs have evolved into systems with a high degree of user-friendliness, due to advancements in computer graphics tools.

The number of examples of DSS employing Expert Systems (ES) technology has been increasing over the latter half of the past decade. The evolution of early ES applications (pre-1980) has dramatically changed the look and feel throughout the 1980s, mainly due to the advancements of computer hardware and software tools. Numerous computer software companies continue to develop software products for prototyping and delivering DSS for countless real-life applications. With improved computing power and graphical interfaces, today's computers are now providing excellent environments for the implementation of operational DSSs. Today, useful expert systems are being developed by scientific, engineering and technical personnel without specific training in ES technology. Considering the advancements in software technology and the increasing speed of new generation computers, developers can build DSS in substantially less time than before.

Within the discipline of water resources engineering numerous successful applications of decision support systems have been documented. Following is a sample of several applications which have been documented recently. Soncini-Sessa and others (1991) developed a prototype DSS for the management of a single-reservoir, multipurpose water resource system. A computer-based DSS prototype for the management of ground-water resources was developed by Kao and Liebman (1991). Hypertejo (Câmara and others, 1990) is a DSS developed to facilitate water quality management in the Tejo estuary in Western Europe. The establishment of better policies for land-use and land-management controls is dealt with in a DSS project planned by Davis and others (1991).

The existence of the numerous, successful examples of decision support systems in the domain of water resources engineering, is evidence that the there is growing need and interest for applying this technology to various problems. Likewise, this review has discovered that there has been very little, if no, work directed at developing a DSS for addressing the problems associated with developing stage-discharge rating curves. Thus, due to the success of other DSS examples in the field, and the need for improved ability to develop accurate rating curves, the work conducted in this thesis is highly justified.

3. STAGE-DISCHARGE ANALYSIS

3.1 Introduction

In this chapter, the reader is introduced to the complex process of stage-discharge analysis. A brief overview of organizations and practices used in Canada are presented in this first section. Next, a review of stage-discharge analysis procedures, used within the Department of Environment (DOE) is presented with respect to concept, data collection and analysis. The final two sections deal with the complex nature of S-D relationships and the hydraulic controls which govern them. An examination of the relative physical characteristics of controls is discussed with emphasis on their affects to S-D relations.

3.1.1 Hydrometric Organizations in Canada

The Inland Waters Directorate (IWD) of the DOE is the primary water agency in Canada, responsible for all freshwater resources in Canada. Through a series of federal and provincial agreements, they coordinate and centralize the collection of water quantity and quality information, across Canada. The IWD also develops and maintains operational standards and procedures for the collection, processing and distribution of resulting hydrometric data.

Over 7700 stations, including 3484 active and 4286 discontinued (Environment Canada, 1992), are operated nationwide, under this joint federal-provincial mandate. Additional provincial stations are also run by organizations such as forestry and environmental departments of the provincial governments and miscellaneous networks of stations for research and other purposes.

3.2 Current Practice in Canada

The process of *Stage-Discharge Analysis* (SDA) refers to the activities associated with developing S-D relationships. These include the collection of streamflow and related hydrometric data, the processing and analysis of data, and the plotting and subsequent curve fitting of data to form graphical relationships. SDA also requires the periodic monitoring of relationships and identification of changes over time. This section introduces the current practices and procedures typically used within Canada by the DOE, for establishing S-D relationships.

3.2.1 Concepts

In essence, a *stage-discharge relationship* is a mathematical equation or graphical curve which relates stage and discharge in a flowing body of water, such as a river or channel. (The terms *stage-discharge relation, rating curve* and *curve* are synonymous and will be used interchangeably throughout this thesis). Generally, these relationships are based on corresponding measurements of stage and discharge at a fixed cross-section in a channel; although other parameters such as water slope are sometimes incorporated into the relationship. Plotted on a linear coordinate graph, with stage as the ordinate and discharge as the abscissa, these relationships classically exhibit a concave-down shape or "curve". These empirical relationships are typically used to estimate discharge in open channels from corresponding continuous water level recordings made at established stream gauging stations.

3.2.2 Collecting Stage-Discharge Data

Determining a rating curve for a stream begins with the selection of an appropriate site for a gauging station. Once the site is established, the common practice is to make periodic measurements of the water elevation (stage) and corresponding stream discharge. In general, a technician visits the gauging site once a month; although this schedule may be more frequent during particular events such as flooding conditions and river ice freeze and break-up at a given station.

One ongoing goal of the collection process is to obtain a good coverage of S-D measurements over the entire range of expected stages in the stream. This task in itself can pose a problem since many stations are remote and cannot be reached on short notice. In this case, technicians often try to predict when floods are expected, in order to time their arrival at the station during the peak of the flood, thus ensuring a high stage (and flow) measurement.

Once at the site the technician records the stage in the river from an installed water level recorder (Terzi, 1981) or crest stage plate. The discharge is generally computed using the velocity-area method; calculated as the product of the water velocity and stream cross-sectional area (Rantz and others, 1982a). A *flow meter* (also known as a *current meter*) is used for making velocity measurements across the channel at a given cross-section. In general, the discharge is basically computed as the product of the average velocity and the cross-sectional area of the channel.

In order to make an accurate discharge computation, the stream is divided into vertical strips of equal widths or equal discharge. The area of each strip is estimated by depth sounding and velocity is measured at one or more depths with a current meter. The discharge is then computed by summing the velocity-area products for each vertical. It should be noted that this is the typical method for measuring discharge in a channel, although many other methods using other types of equipment, exist. For further discussion of these methods, the reader is directed to ISO Handbook 16 (ISO, 1983).

Once the discharge is computed for the station, the technician makes one final reading of the gauging stage. This is done to document the change in stage occurring during the time of stream gauging. A comparison of the two stage measurements may indicate a rapidly rising or falling stage condition. This entire procedure is repeated over a range of stages in the stream, usually several times a year.

3.2.3 Establishing a Rating Curve

When a substantial number of measurements have been recorded, the stage-discharge measurements or "points" are plotted on linear graph paper, with stage on the ordinate. Data is often plotted in differing scales to reflect the various ranges of stage in a stream. This is to provide the degree of accuracy necessary for later computation of a rating table in all flow regimes.

A smooth curve is then drawn through the points using "french curves". A graphical method is used to determine the *zero-flow* stage, and complete the "best-fit" curve. Curve extensions beyond the highest available stage are often performed for new stations with very few measurements. A logarithmic graphical method is sometimes used to perform such extensions.

The entire process of establishing a S-D relation can take hours to complete. Once the curve is completed, its shape is transformed into a *rating table*, relating stage specified in equal increments to corresponding discharge values. A digitizing pad is used to transfer corresponding S-D points along the curve into a computer interpolation program, which then interpolates between the points to generate a table of S-D values representing the curve. Continuous water level recordings may then be applied to the rating table to generate average daily discharges for the station.

It should also be noted that fitting the so-called "best-fit" curve to a set of S-D measurements is purely subjective on the part of the technician. The technician must rely on their judgement, experience and intuition when fitting the curve, as well as consider the conditions and reliability associated with each individual measurement. However, the latter aspect is difficult to quantify objectively.

This procedure is only one small segment of activities performed by the regional branches of the DOE. New stage-discharge measurements are continuously recorded and used to verify curve stability and to make corrections, if necessary. The process is very time consuming and requires a great deal of technician hours. In addition the procedure is substantially complex. As will be shown later in this chapter, stage-discharge measurements rarely form a distinct curve since streams are often under the influence of imperfect conditions. This leads to measurement points which deviate or "shift" significantly from the general curve shape. Therefore the technician requires an understanding of the hydraulic and hydrologic parameters controlling stage-discharge relationships.

3.3 Stage-Discharge Relationship Controls

This section deals with the concept of "station controls" and specifically, their effect on stage-discharge relations. Various types of station controls and their classifications will be discussed, followed by an examination of physical characteristics and their specific influence on controls.

3.3.1 Types of Controls

A *station control*, or simply *control*, has been defined as a physical element or combination of elements that controls the relationship existing between stage and discharge (Rantz and others, 1982a). This is generally a section or reach of the channel downstream of the gauging site, which is either a natural part of the channel or an artificial man-made structure. In order for a control to serve a gauging site satisfactorily, it should exhibit permanence (stability) and sensitivity.

Permanence refers to ability of a control to retain all its original effective physical characteristics which ultimately affect the S-D relation. If a control is not stable (ie: permanent), neither will the relation be stable, and frequent discharge measurements will be required to trace the changes in the relation. A control is said to be *sensitive*, if small changes is discharge are reflected by relatively large changes in stage. This feature is particularly important at low flow conditions, since one would like to be able to notice a change in discharge without having to measure stage to a fraction of an inch. To facilitate sensitivity, the width of flow at the control be should be significantly constricted at low flows. Usually a *notched control* such as a V-shape or parabolic shape will ensure that the width of flow decreases as the flow decreases.

Control types are often defined using different classifications (Rantz and others, 1982a). The major classification labels controls as either section or channel controls. The term *section control* refers to a single cross-sectional geometry located a short distance downstream of the gauging site, which tends to constrict the channel. Constrictions may be a rise in the streambed or a locally reduced effective width, created by natural or man-made means. Alternately, the term *section control* may also be used for downward breaks in the bed slope at a particular cross section, such as the head of a cascade or brink of a falls.

In contrast, a *channel control* refers to a much longer reach downstream of a gauging station where the geometry and roughness characteristics of the channel become the controlling elements of the S-D relation. Increases in channel discharge will cause an increase in the amount of reach length which is effective as a control. Likewise, flatter stream gradients tend to have longer reaches of channel control.

A second classification, already alluded to above, distinguishes controls as either natural or artificial. This classification is generally used in combination with other classification terms to describe the origin of the control. Examples of *natural controls* include physical features such as rock ledges, outcrops or falls in the channel or distinct streambed features such as riffles. *Artificial or man-made controls* take many forms and are generally installed to provide, among other things, a stable and sensitive feature within the channel, for the purpose of establishing a S-D relation. Typical structures include weirs, flumes and overflow dams. Bridge piers which generally reduce the natural cross section of a channel are often referred to as a man-made control, be it intentional or not.

Finally, a third classification is used for delineating the effect of the control throughout the experienced stage range for a gauging station. Controls which are not limited by stage in their ability to control a S-D relation are referred to as *complete controls*. Such conditions exist for section controls that exhibit significant height which is not drowned out during high flows. Likewise, channel controls may be complete if no section control exists to constrict the channel in low flows.

The last condition brings up an interesting point. It is not common that natural features provide a control which governs the S-D relation throughout the entire range of flows experienced in a channel. This leads to the term *combined controls*, which are made up of several different controls, each governing a particular range of stage (and discharge) in the stream. A common example of a combined controls was alluded to at the end of the previous paragraph, where a section control exists and controls the relation at low stages but is drowned out at high flows. A channel control may then control the relation. In this case, the two combined controls are also referred to independently as *partial controls*, since they do not control the entire S-D relation. With respect to the relation itself, it should be noted that in a combined control, there is a transition period that exists where both partial controls act together.

3.3.2 Effects of Physical Characteristics on Controls

As can be seen in the previous section, classification terms are usually mixed and matched to describe the make-up of a station control. Knowledge of these terms is thus essential in S-D analysis when trying to understand the various effects physical characteristics have on a S-D relation.

The largest problem that exists in S-D analysis is the phenomenon known as a *shifting control.* This concept was discussed briefly in the previous section as stability and permanence. Essentially, a shifting control results from gradual or abrupt changes in the physical features that form the control. The main effect of these changes is that of a change in stage for a given discharge, or conversely, a change in discharge with no change in stage. It should be noted that, a relation may not necessarily change if the characteristics of the control are collectively compensating with respect to their effect on the stage. The following sections discuss common physical features and characteristics of open channels and their controls, and the individual influence of each on channel discharge, relative to a fixed stage level.

Backwater

Backwater is a common phenomenon resulting from a reduced flow velocity downstream of a gauging site. Be it apparent to the eye or not, backwater exhibits a reduced surface water slope in the direction of the streamflow, in comparison to the surface slope when no backwater exists. Graphically, this translates to a shift to the left of the normal flow curve.

Reduction in flow velocity may be attributed to conditions such as constricting narrow reaches of the channel or reaches affected by seasonal conditions such as ice, vegetal growth, beaver activity and moving sediment deposits. Artificial structures such as dams, weirs and locks can also induce backwater conditions, at certain times or stages. Streams may also experience backwater affects due to increased volume and stage levels in downstream tributaries. Finally, reaches under the influence of tides will, more than likely, experience backwater.

Depending on the cause of the backwater, varying slopes may or may not exist for the same stage. Under the former circumstances, discharge is said to be a function of both stage and water surface slope. In this case, it is common to develop a relation for stage-fall discharge. This is determined empirically by observing the "fall" of the water surface in addition to typical stage and discharge parameters. *Fall* is measured by observing the difference between the gauging station stage and a stage measured at an auxiliary gauging site downstream.

Channel Modification

Probably the most obvious feature of a station control is its cross sectional and longitudinal geometry. Alterations to either of these features above, below or at the control usually has an adverse affect on the normal S-D relation at a gauging site. Depending on the type of modification, the effect may be either a *left* or *right shift* of the curve; that is, the discharge corresponding to a given stage may either decrease or increase, respectively. Channel modifications can result from natural or human activity and may be as unsuspected as mowing or brush cutting in the high-flow area of the channel. Some examples of natural channel modifications include normal scour and fill of the channel, sloughing or recession of banks and meander cut-offs. Man-made modifications include channel realignment, dredging, installed rip-rap and dam emplacements. In addition channel constrictions such as beaver activity, fallen trees and debris caught on bridge abutments are considered channel modifications. In the case of these last examples, the resulting increase in stage is often visibly distinguishable on analog and digital recorders installed at gauging sites. Non-stable streams (see section 3.4) are typically the result of continuous, natural channel modifications.

Rapidly Varying Stage

It has been shown conclusively from actual measurements that discharges for a given stage will be greater during rising stage than during falling stage (Rantz and others, 1982b). This fact is derived from the fact that rising and falling stages have differing water surface slopes. At the onset of a flood, the slope is much steeper on the rising limb of the flood wave in comparison to the normal flow conditions. This slope causes a shift of the rating curve to the right of the curve derived from normal flow conditions. The reverse effect occurs on the falling limb but to a lesser degree of magnitude. The falling stage condition is similar to that of a backwater condition as described above; shifting the rating curve left of the normal flow curve.

Vegetal and Aquatic Growth

Stage-discharge relations are often affected by the growth of aquatic vegetation on natural controls and channel banks. Growths of these types directly alter a control by reducing channel capacity and by altering the roughness coefficient of the channel. During low flow conditions, aquatic growth on a section control causes a decrease in discharge for a given stage. During higher flows, a reduction in the effective area of the cross section is experienced in the presence of vegetal growth along the banks of a channel control.

In general, the cyclic effect of aquatic growth is gradual in the spring and remains fairly constant throughout the effective period before diminishing rapidly at the onset of the first subfreezing temperatures. The magnitude of aquatic growth in a particular channel may differ from one year to the next, depending on how deviant the temperatures are from the normals for the particular region at a specific time. It should also be noted that the distinct effects of vegetal growth (ie: backwater) may be reduced or removed entirely, due to temporary high velocities and/or flows moving through the controlling reach. Such is the case, when vegetation exhibiting high roughness characteristics is "flattened" or washed away by fast moving water. Flattened vegetation may reduce channel roughness, ultimately resulting in a right shift of the curve, (i.e. the opposite affect of backwater).

Ice Formations

The basic effect of all ice formations results in backwater at the station control and is manifested by a left shift of the rating curve. Three basic formations of ice are frazil, anchor and surface ice.

Frazil ice is formed at the surface of turbulent water as fine, elongated needles, thin sheets and/or cubical crystals. Masses of floating slush form from frazil ice which has moved to slower moving water, where it has the opportunity to consolidate. Neither floating slush or frazil ice have a direct affect on the stage-discharge relation; however the use of current meters under these ice conditions is likely to induce significant measurement error.

Anchor ice refers to spongy ice or slush which adheres to the rocks of a streambed and continues to grow upward. Ice growth of this nature tends to change the geometry of the streambed or control where it forms, thus resulting in an increased stage for a given discharge in comparison to conditions without the formation.

Surface ice is that ice which forms a complete cover of the stream, replacing the water-air interface of open channel flow. The basic effect on the stage-discharge relation is again an increase in stage for given discharge. Rantz and others (1982b) explain this effect is a combination of three conditions, resulting from the ice cover.

First, the resistance to flow in the channel is increased due to additional area of friction in contact with flowing water. This resistance to flow may decline over winter however, as flowing water smooths out the originally rough texture of the under side of the ice. Second, in particularly thick ice covers, the cross sectional area available for flow may be reduced to varying degrees. In small channels, this effect may vary over winter if water levels drop, leaving the ice sheet to hang from the banks, not touching the water surface. Finally, the increased wetted perimeter of the section reduces the hydraulic radius of the section.

A final note should be included about ice in channels. Quite often, ice becomes jammed up on a section control or downstream of a gauging site, causing the typical backwater effect in the channel. The affect of ice jams, (as well as that of any other type of debris becoming caught in the section) downstream of the station will result in backwater, producing a left shift of the stage-discharge curve.

3.4 Stable and Unstable Channels

The relationship between stage and discharge in any given channel is likely to have some degree of change at one time or another. Changes in a relationship, or *shifts* as they are more commonly referred, may be temporary or permanent. This section addresses the analysis of S-D relations which experience various degrees from shifting.

The previous section discussed general phenomenon which influence hydraulic controls, governing a stage-discharge relation. Channels are basically distinguished as being either *stable* or *unstable* by the pattern or trend of the variability in their relationship between stage-discharge. These trends of variability, or lack thereof, may be examined by use of a specific gauge plot. A *specific gauge plot* of a station, is a plot of gauge over time, for a constant discharge. In general, three plots are presented for three different discharges, representing various flow regimes in the channel. Constant discharge values are usually selected for the plots corresponding to the 2, 5 and 10 year return floods.

Figure 1 shows three specific gauge plots for Edwards Creek in Manitoba. The three plots represent the gauge (stage) in Edwards Creek from 1980 to 1990, for discharge measurements of 0.40, 0.8 and 2.0 m^3/s respectively. The gauge is plotted along the ordinate in metres and time along the abscissa in years. Trends exhibited in specific gauge plots are used to indicate whether the stage is changing over time or remaining essentially, for a constant discharge. A constantly level gauge over time tends to indicate a stable channel. That is to say that the relationship between the stage and discharge is not changing over time, for the specific discharge indicated.

In Figure 1, each of the three specific gauge plots displays a corresponding gauge which is essentially constant, over the 10 year period. Individually, each level trend indicates that the stream is stable at a specific discharge level. Collectively, the three level trends indicate that the Edwards Creek is essentially stable, since the gauge does not change over time, irrespective of the discharge level. This conclusion could be further verified by generating specific gauge plots for different flow values. The three chosen in Figure 1 represent the 2, 5 and 10 year return floods in Edwards Creek.

Quite often specific gauge plots are not level and may exhibit a varying character over time, which may be increasing or decreasing in a constant or step-wise pattern. In other cases, the plot may fluctuate sporadically over time, with no apparent pattern. Regardless of the pattern, these manifestations of trends which are not level indicate the relationship is changing, or "unstable", with time. Any of these types of non-level trends will tend to indicate unstable channel conditions.

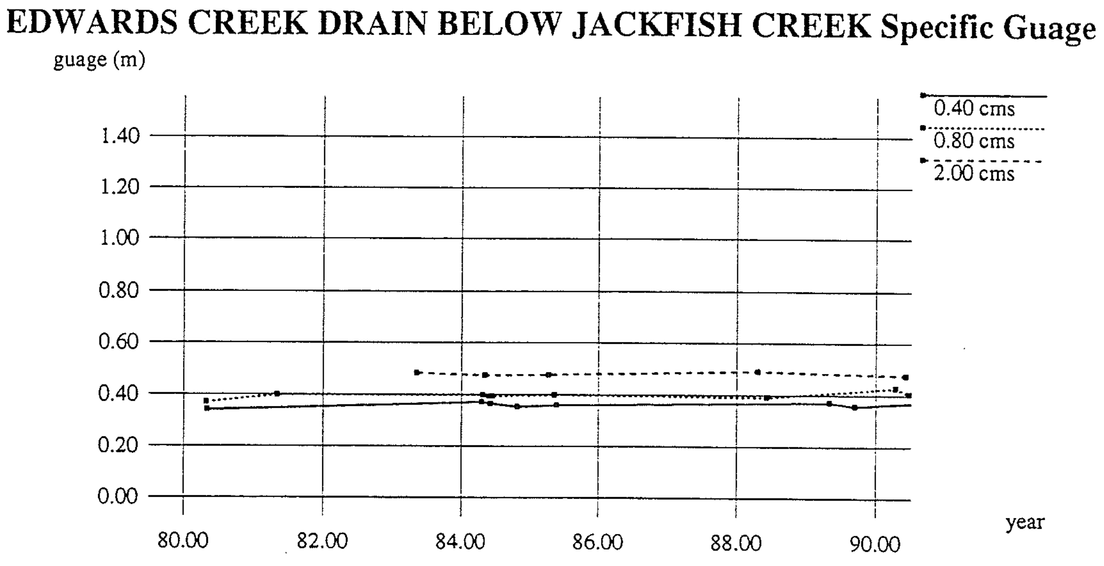


Figure 1. Specific Gauge Plots for Edwards Creek.

3.4.1 Stable Channels

As was discussed in Chapter 1, the term "stable" is a relative one, when describing the relation between stage and discharge in a stream. A *stable channel* will generally exhibit little or no *continuous variability* (see section 3.4.2) in its S-D relation over time. Such an "ideal" relation is often exhibited in channels cut through bed rock and man-made canals lined of concrete. However, stable channels are not limited to such specific conditions. Rantz and others (1982b) point out that virtually all natural channel are subject to occasional changes due to various factors. The key with stable channels is that the general geometry and form of their cross-section is relatively constant from year to year.

The previous section showed that relationship variability is directly related to the station control and physical characteristics associated with the control(s). Stable channels exhibit temporary changes, generally on an annual basis, which in turn affect the relation. They are distinguished from unstable channels in that they do not progressively change over time, but rather experience similar shifts from year to year. The next section, dealing with unstable channels, will discuss conditions which continue to evolve over time, thus causing a continuous change in the relationship between stage and discharge.

In the analysis of stable channels, changes are considered to be somewhat repetitive from year to year. This means that certain shifts are expected at particular times of the year. However, magnitudes of backwater effects, for example, will translate to differing magnitudes of a (left) shift in a relation. Primary sources of control shifts in stable channels include ice; backwater created by beaver activity, weed growth and downstream tributaries (to name but a few); and rapidly rising and falling stage conditions. In addition, channel modifications such as meander cut-offs, either upstream or downstream of the gauging site may be seasonal and somewhat repetitive.

A final factor must also be considered when performing analysis of S-D relations. Winds experienced at gauging sites can significantly affect the accuracy of discharge measurements, resulting in a perceived shift when plotted. This may be exhibited in a number of different ways. Strong cross winds may obscure the angle of the current in the channel. Winds blowing against the streamflow tend to elevate the water level in a stream, while those blowing in the direction of the streamflow may effectively reduce friction and impart energy in the stream. Stream gauging may also be subjected to several sources of significant error due to wind action. Waves resulting from winds may conceal the true water level during depth sounding and in shallow streams, the velocity at the 0.2-depth may be influenced, thus distorting the velocity distribution in the measured vertical.

Whatever the source or sources of the shift are, they must be identified during analysis to assist in the development of the rating curve. For example, in the case of temporary backwater, (say from beaver activity), it is common for a water level records to be adjusted according to the amount of shift associated with the backwater. Once adjusted, the water level record can then be applied to the normal relation, as if no backwater condition existed. However, depending on the source of the backwater, surface slopes may vary thus requiring different shifts adjustments to be made based on the type of the backwater. For these and other reasons, it is of the utmost importance that the observation of relevant conditions should always be noted when collecting hydrometric data and used to understand deviations from the expected relation.

Although some deviation is always expected, recurring deviations are often expected in stable channels. These predictable shifts are often established as separate S-D relations and used in place of singular flow relations, (which is the typical practice in the DOE). A typical example of this is the condition of rapidly varying stage. S-D relations often include separate curves for the conditions of rapidly rising and falling stage, known as a *hysteresis loop*. One might expect that flood waves of different magnitudes will produce differing deviations of the loop from the steady flow condition curve. However, this is only apparent in the case of rapid change in discharge and in channels with a relatively flat stream slope. Thus, with the exception of fast moving floods, S-D relations should exhibit one fairly distinct loop. Obviously, the more measurements recorded under flooding conditions will make this curve more apparent to the technician.

3.4.2 Unstable Channels

In contrast to stable channels, *unstable channels* are termed as such because they experience continuous changes in their stage-discharge relations over time. This is most typical in alluvial channels which, by definition, develop in the bed material which they transport. Analysis is more complex, since bed formations in alluvial channels are a function of the flow and the bed material carried by the channel (Rantz and others, 1982b). This means that not only is the flow resisted by the roughness elements, but the roughness elements are in turn formed by the flow.

Analysis of *unstable channels* involves a much more complex process than of the process used in the analysis of stable channels. The very nature of alluvial channels is such that there is a continually changing control with time (Rantz and others, 1982). This is due to scouring and deposition activity of bed material, braiding and meandering patterns of the channel, and changes in the bed formation. Resistance to flow caused by changes in bed formation, (and therefore roughness) influences the form of the stage-discharge and depth-discharge relation in alluvial streams. Bed formations (and the associated roughness) are dependant on parameters such as the properties of the bed material, including size, shape and density; the magnitude of shear stress applied by the water on the bed; seepage forces; sediment load; and water temperature.

Stage-discharge relations are relatively stable at the high flow condition, if upper regime bed formations are in effect. The lower end of the relation may very well shift with time due to bed formation influences in the lower regime. It should also be noted that, in addition to these continuous changes, all of the same physical characteristics of section and channel controls listed in section 3.3.2 are applicable to unstable channels as well, complicating matters further.

4. COMPUTERIZED DECISION SUPPORT TECHNOLOGY

This chapter deals with the concept of computerized decision support systems (DSS). The first section follows the developmental history of DSS technology from early days of simple management information systems to present day conceptual structures and components. The next section introduces the concept of developing *intelligent* decision support systems with the use of *Expert System (ES)* technology. The final section follows, introducing the concept of a modular computer decision support system, for implementation in the Department of Environment. The design of one module of this system comprises the major work of this thesis.

4.1 The Evolution of Decision Support Systems

The intent of this first section is to familiarize the reader with the DSS technology; its fundamental purpose, history, components, and application.

4.1.1 Early Information Systems

Access to information has always been an essential part to the operation and management of any company or organization. "During these fast-changing times", Thierauf (1988) regards information as the "sixth resource" of a typical organization; after people, machines, money, materials and management. Information regarding a company's past performance, current conditions and expected trends in the environment provide a basis on which managers make planning, control and operational decisions.

Early information systems were aimed at producing historical reports which evaluated past performance. Sometimes referred to as *Electronic Data Processing* (EDP), these systems focused on data, storage and processing at the operational level (Simonovic, 1992). Later, EDP principles were integrated with business functions to produce intelligent data-retrieval systems. These *Management Information Systems* (MIS) were developed to assist managers with short-term planning aspects of an organization. Emphasis was placed on fast retrieval and processing of timely information, to be relayed to the controlling operations of the environment. Systems of this nature were limited to operating environments definable in well-structured frameworks.

4.1.2 The Essence of Decision Support Systems

Decision Support Systems (DSS) were developed to operate in environments with less than structured frameworks. They directly addressed the problems of the decision process requiring human expertise, judgement and intuition. Depending on the degree of the problem structure, DSS can incorporate computerized decision-making based on quantitative modelling techniques.

Simonovic (1992) describes the following characteristics, necessary for a DSS:

- assist managers in the decision making process for unstructured and semi-structured problems;
- support and enhance managerial judgement;
- improve the effectiveness of decision making;

- combine the use of models or analytical techniques with data access functions;
- exhibit flexibility and adaptability to changes in the decision process; and
- focus on features which make them easy to be used interactively.

Thierauf (1988) cites DSSs as having an added dimension in comparison to earlier information systems. As the name suggests, decision support systems emphasize decision-making, user support and integration of computer and human abilities in a single framework. He stresses that DSS must exhibit the following characteristics:

- focus on problem finding and problem solving;
- use of an interactive processing mode; and
- a comprehensive systems approach.

This added dimension has allowed DSS to break into the diverse field of engineering, allowing the introduction of subjective judgements, trial-and-error procedures, and intuitive approaches to achieve problem solutions (Simonovic, 1992).

4.1.3 DSS Architectures and Components

Simonovic (1992) outlines three design approaches used to assist the DSS designer with conceptualizing, evaluating and characterizing a proposed system architecture. The *functional approach* incorporates a language system for problem formulation, a problem processing system for elaboration of the problem and a knowledge system for information storage. So-called *tool-based architectures* conceptualize DSS as an integration of individual technologies or components. These components include database management, modelling and dialogue modules. Finally, a *combined architecture* is based on the two previous concepts but incorporates a system manager to handle the overall control of the specific components.

Traditionally, DSSs have also been described as modular systems consisting of three components. Not surprisingly, these are similar to those of the tool-based and combined architectures listed above. The remainder of this section provides a brief description of the three typical components of a DSS, outlining their respective responsibilities and make-up. Later, in Chapter 5, a prototype DSS will be introduced, which also applies this concept to the problem of Stage-Discharge Analysis.

Most DSS application problems deal with tremendous amounts of data, which must be processed in order to render decisions. The *database management component* of a DSS is responsible for the storage and maintenance of factual data. It generally consists of a management system, data dictionary (containing descriptions of the types of data stored in the system), a querying facility (for isolating and retrieving specific data), and a facility for accessing external sources of data as well as connecting the DSS with other systems (Câmara and others, 1990).

Next, the *modelling component* is composed of various mathematical models available in the system, used for assisting in the problem-solving process. It is responsible for operation and execution of mathematical models and provides adequate linkage with the other system components to facilitate the transfer of commands, data and results.

Finally, the *dialogue component* provides a means by which the user may communicate with the system. All system input and output is handled through the dialogue component. Depending on the specific architecture of the DSS, this component may be as simple as a command language used to instruct the system to perform certain tasks, or a complex input/output (I/O) device connected to a central control unit, which handles the assignment of tasks. In this latter case, the central control unit, or *System Control Manager* (SCM), acts as the main link between all other system components and provides the main problem-solving mechanism.

4.2 Intelligent DSS and Expert Systems Technology

The rapid progression of DSSs and their underlying technologies has led to the development of more sophisticated and intelligent computer support systems, incorporating newer and more advanced concepts in the realm of *Artificial Intelligence* (AI). Thierauf (1988) noted that future advancements in the area of AI will allow the individual to use new knowledge in order to be even more efficient as a decision-maker. Thus a fourth component, "knowledge" is acknowledged in *Intelligent Decision Support Systems*, (IDSS). The following section describes the concept of Expert System technology, and its contribution to the development of IDDSs.

4.2.1 A Brief Background of Expert Systems

Recently, the terms *Expert System* (ES) and *Knowledge-Based Expert Systems* (KBES) have been receiving "buzz-word" status among professionals since the mid-eighties. The concept of expert systems stems back over two decades prior to the great interest recognized in the mid-eighties. They originated as early attempts at developing computer programs to simulate and reproduce intelligent problem-solving behaviour. Maher and Allen (1987) explain that this is attributed to the high profile received by several relatively successful expert systems and the potential for development in diverse applications. Combined with the rapid acceptance and advancement of the personal computer throughout the 1980s and development of innovative software, expert system technology has moved from the once dominated university development environment to the business community.

Gaschnig and others'(1981) definition of expert systems as "*interactive computer programs incorporating judgement, experience, rules of thumb, intuition, and other expertise to provide knowledgeable advice about a variety of tasks*" was cited by Maher and Allen (1987) and is widely accepted. However, Maher and Allen (1987) criticize that this definition fails to delineate the differences between expert systems and conventional computer programs, as do most definitions, in their opinion. A true comparison reveals that expert systems focus on the symbolic processing of knowledge using heuristics, rather than sequential algorithmic processing of numerical data. This discussion will not indulge in any further comparisons, although the reader is referred to Maher and Allen (1987). Suffice is to say that ES technology provides an environment for capturing and representing human knowledge, experience and expertise in logical computer frameworks.

Some of the earliest success stories in ES technology include a medical diagnosis system for identifying infectious diseases known as MYCIN and PROSPECTOR, a geological information interpreter (Maher and Allen, 1987). The success of these systems is attributed to their ability to solve problems in their respective domains at the level of an expert and communicate efficiently with users.

Several important applications using ES and *Knowledge-Base* (KB) technology (see section 4.2.2), in the domain of operational hydrology, were reviewed by Simonovic and Savic (1989). FLOOD ADVISOR is a computer-based consultant which assists hydrologists with the selection of a suitable model for flood estimation under five general situations. Model selection is made on the basis of the type and quantity of data available. The system incorporates the human expertise and judgement of hydrologists in its knowledge-base. A second ES application in operational hydrology, HYDRO, deals with the task of describing the physical characteristics of a watershed. The system was developed to aid in establishing numerical parameter values, for subsequent use in a watershed modelling program. The program simulates the physical processes by which precipitation is distributed through a watershed.

Another expert system called the Stream Flow Management Advisory System (SFMAS) was developed for aiding managers in the selection of a suitable method for measuring flow in open channels (Douglas, 1990a; Douglas 1990b; Simonovic, 1990; and Simonovic, 1991). SFMAS is based on recommendations published by the International Organization of Standards (ISO). Physical channel characteristics, flow conditions and available equipment are used in the selection of the most suited method.

ES technology is related to DSS by the simple fact that they are both concerned with assisting in problem-solving tasks. ES provide enhanced problem-solving capability beyond DSSs because they contain qualitative knowledge of a particular domain. Knowledge-bases store knowledge within the ES, primarily as symbolic elements rather than numerical. Problem-solving tasks using Ess are therefore handled in a qualitative sense, using human expertise and judgement represented in the KB. Thus, DSSs can benefit greatly from ES technology.

With respect to the domain of stage-discharge analysis, many applications for ES technology exist. Specific knowledge regarding section and channel controls can be incorporated into a DSS. Various characteristics and properties, and their effects on S-D relations, may also be stored as knowledge within an ES. The DSS would be capable of making inferences on this knowledge, in order to suggest the relative influence on rating curves. Regression analysis modelling generally requires subjective decision making, based on human expertise. Such expertise may also be incorporated into a KB, in order to enhance the modelling procedures of a DSS.

4.2.2 The Architecture of Expert Systems

Expert systems typically consist of 3 main components; a *knowledge base*, *inference engine* and *user-interface*. Within these components, developers can express expert knowledge of both a qualitative and a quantitative nature.

The *knowledge base* (KB), which is analogous to a data base of an information system, stores information about a specific problem domain. Information may be obtained by interviewing experts of the specific problem domain or by including heuristics from recorded case studies (Davis and others, 1991). Once obtained, information is stored in discrete elements known as *rules*. These logical structures are based on the same principles as the familiar "if-then" programming structure, often used in many programming languages.

Each rule stores one elementary piece of information or knowledge, within an "if" and a "then" clause. The *If-clause*, *premise* or *condition* of the rule (as it is also referred) defines the conditions of certain problem parameters, that must exist to infer some result or piece of knowledge that is stored in the *Then-clause*. The conditions of the *If-clause* form a logical expression that evaluates to either a true or false value. If the conditions evaluate to true, the rule is said to be "fired" and the conclusions, listed in the *Then clause*, are established.

The *inference engine* of the ES provides the methodology for retrieving information from the knowledge base. KBs are sometimes envisioned as massive solution spaces for a particular problem domain, made up of qualitative elements. In this sense, ES and linear programming models, (and other optimization models), are very similar, in that a solution space is defined by problem constraints and the objective is to search the space for the best (although sometimes, compromising) solution. The inference engine of the ES provides a search mechanism for defining the path to an appropriate solution in the KB, as the Simplex Method does in Linear Programming. So-called *backward chaining* and/or *forward chaining* strategies may be used together or independently to link facts stored in the knowledge base rules to attain the problem solution.

4.3 A Proposed Computer System

The previous two sections have served to introduce the concepts of decision support systems and expert systems technology. These concepts have been proposed for application in the development of a computer system for use in the Department of Environment.

A large, modular system was conceived in a joint venture by Environment Canada and the University of Manitoba and was introduced in the late 1980s by Simonovic (1989) as an *intelligent decision support system* (IDSS) for the management of surface water quantity data. Formulation of the system structure incorporates;

1) existing databases and practices used within the Department of Environment;
2) an IDSS for surface water quantity data acquisition; and
3) a set of mathematical models and tools.

Figure 2, shows a conceptualized diagram of the overall IDSS structure, and the relations among the various elements and operations (Simonovic, 1989). Selection, support, and integration capabilities were to be provided within the IDSS framework, through the implementation ES technology. This application resulted from interest generated by a study which explored the opportunities of *Artificial Intelligence* (AI) technologies in government organizations (Simonovic, 1989). The study cited ES tools (and techniques) as the most promising AI concept for improving productivity and efficiency in the Canadian government.

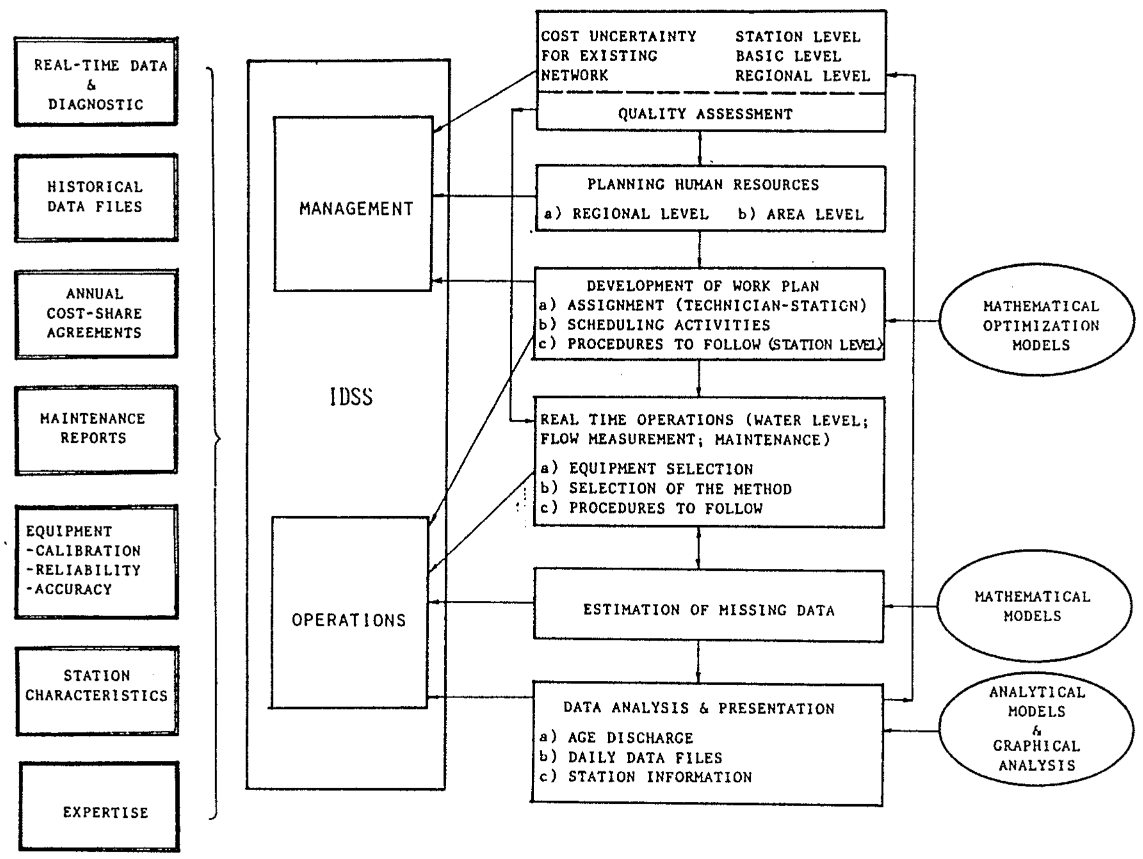

REAL-TIME DATA & DIAGNOSTIC
HISTORICAL DATA FILES
ANNUAL COST-SHARE AGREEMENTS
MAINTENANCE REPORTS
EQUIPMENT -CALIBRATION -RELIABILITY -ACCURACY
STATION CHARACTERISTICS
EXPERTISE
IDSS
MANAGEMENT
OPERATIONS
COST UNCERTAINTY FOR EXISTING NETWORK
STATION LEVEL BASIC LEVEL REGIONAL LEVEL
QUALITY ASSESSMENT
PLANNING HUMAN RESOURCES
a) REGIONAL LEVEL b) AREA LEVEL
DEVELOPMENT OF WORK PLAN
a) ASSIGNMENT (TECHNICIAN-STATION)
b) SCHEDULING ACTIVITIES
c) PROCEDURES TO FOLLOW (STATION LEVEL)
REAL TIME OPERATIONS (WATER LEVEL; FLOW MEASUREMENT; MAINTENANCE)
a) EQUIPMENT SELECTION
b) SELECTION OF THE METHOD
c) PROCEDURES TO FOLLOW
ESTIMATION OF MISSING DATA
DATA ANALYSIS & PRESENTATION
a) AGE DISCHARGE
b) DAILY DATA FILES
c) STATION INFORMATION
MATHEMATICAL OPTIMIZATION MODELS
MATHEMATICAL MODELS
ANALYTICAL MODELS & GRAPHICAL ANALYSIS

Figure 2. Structure of IDSS, (from: Simonovic, 1989)

Development of the SFMAS discussed in section 4.2.1, was performed at the University of Manitoba and served as a pilot project, intended for integration with the modular IDSS. The work of this thesis has developed a second module for inclusion in the overall IDSS. This module, known as the ***Stage-Discharge Decision Support System* (SDDSS),** provides an alternative to the present, time consuming method of manual plotting and curve fitting for production of stage-discharge rating curves, discussed in

Chapter 3. Using statistical analysis techniques, the system considerably reduces the subjectivity of the "manual" curve-fitting procedure. This is necessary to provide a basis for consistent and standard work among technicians, as well as to provide a means to quantify the "goodness of fit" of the curve to the data. The system also incorporates specialized expertise, based on knowledge obtained from practising technicians. Existing data bases, compiled over years of monitoring by the Department of Environment, are used by the SDDSS for the establishment of rating curves.

The intent of the system development in this thesis, was to examine the concepts of the SDA problem, and serve as a learning tool, for later development of a fully operational SDDSS. For this reason the SDDSS will also be referred to as a *Prototype* throughout the remainder of this thesis.

5. A PROTOTYPE DECISION SUPPORT SYSTEM FOR STAGE-DISCHARGE ANALYSIS

This chapter discusses the application of DSS technology and concepts to the SDA process. First, the problems of stage-discharge analysis are outlined and related to the problem-solving capability of DSS technology. Next, the concept of a prototype DSS for SDA is introduced and a description of the computer technologies integrated within the SDDSS prototype. Finally, the operational structure of the SDDSS is presented to the reader.

5.1 DSS and the Stage-Discharge Analysis Process

Chapter 3 presented the reader with a background of the S-D analysis practices currently used by the Water Survey of Canada, a branch of the DOE. It described the dynamic complexities of station controls and their changing physical characteristics. Finally, the chapter outlined the main difference between stable and unstable channels and the relevant parameters to consider with respect to the analysis of relations in each.

The main purpose of presenting the material and background in chapter 3 was to outline the complexities involved in S-D analysis. SDA is not simply a process of plotting stage and discharge measurements on a graph and fitting the most appropriate french curve to the data. Nor is it a pure statistical problem of regression analysis. SDA is an involved process, comprised of several fundamental procedures, reinforced with various levels of human judgement, intuition and decision making. For this reason, a DSS was consider for assisting in the process of SDA.

Some specific aspects of SDA have been identified, which fit neatly within the conceptual structure of the "combined" DSS architecture, (see section 4.1.3). To begin, data is the fundamental element of SDA, on which all other procedures and designs are based. For this reason, the data management component of a DSS provides a facility for efficient organization, storage and retrieval of S-D data. Statistical modelling using regression analysis provide the ability to represent rating curves with mathematical equations, replacing the need for generating hand-drawn curves and rating tables. Various mathematical modelling techniques may be stored as routines in the modelbase component of the DSS.

The interaction between the DSS and the user is extremely important issue in a DSS, for obvious reasons. Technical knowledge must be transferred to and throughout the system and results must be relayed back to the user. The dialogue component is typically responsible for user-interface and the control of information within the system. However, in the combined approach for conceptualizing a DSS, these tasks are handled by a central control unit, linking the various components of the system.

A knowledge base was included for the representation of facts and information regarding the specific problem domain. It is here that specific knowledge with respect to SDA may be integrated into the development of S-D relationships.

5.2 Design Framework

The following section introduces the main design concepts of the *Stage-Discharge Decision Support System* (SDDSS). A system architecture is presented, outlining three distinct user modes. Aspects of the modelling theory and evaluation are also discussed in the final section.

5.2.1 Analysis Concept

The basic design of the SDDSS was modelled around several main activities performed by the Water Survey of Canada in the DOE. From these activities, the system was divided into three main processes: Curve Development; Curve Use; and Curve Modification. From a programming perspective, these processes were divided into separate modules, in order to provide a design framework allowing for future additions and modifications. Within the system, each process is made up of several smaller procedures, which may be shared among other processes. Thus, the main system is a modular architecture consisting of three main user-mode components, each linked to a main menu and user-interface, as shown in Figure 3.

The main objective of SDA is the development of rating curves from S-D observations, and is modelled within the first user mode, *Curve Development.* The SDA concept designed within the system exhibits a hierarchical nature of data selection, as shown in Figure 4. Developing a rating curve logically begins with the analysis of all available data. The data is modelled using statistical techniques and represented with a mathematical equation. The model is then plotted against the data and evaluated for its

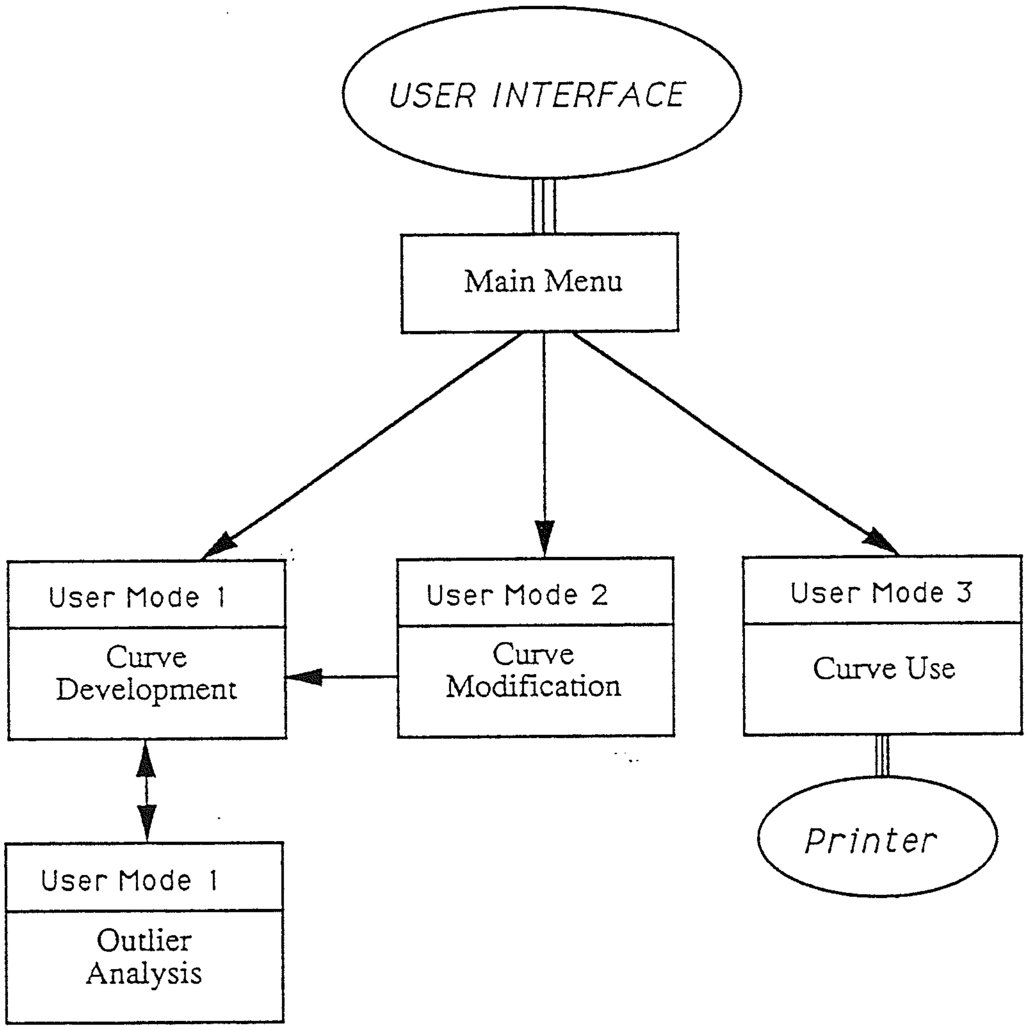


Figure 3. SDDSS Architecture.

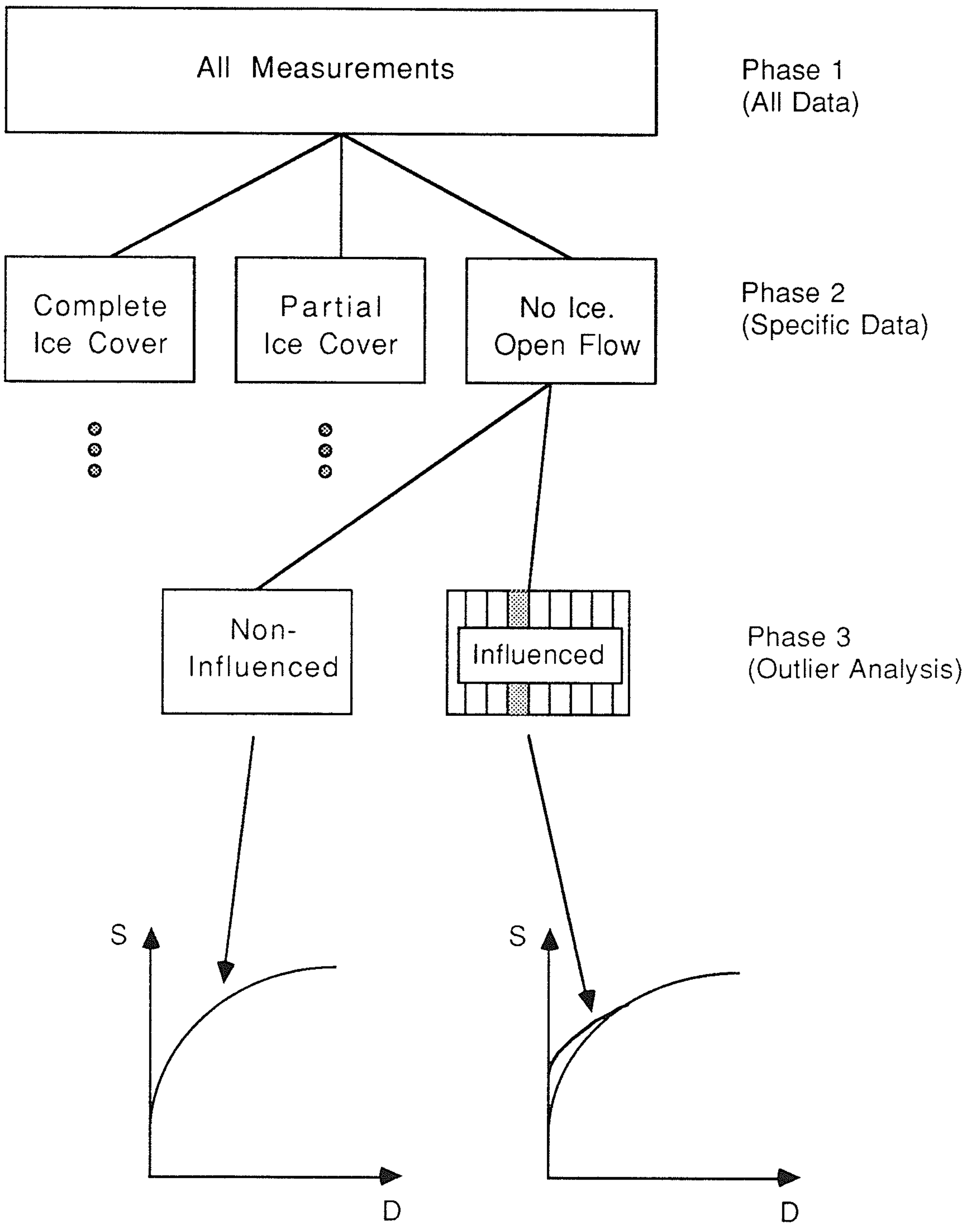


Figure 4. Hierarchical Data Selection.

effectiveness in representing the data. This makes up the *highest level of the analysis hierarchy*. Due to the complex graphical nature of most S-D relationships, it is unlikely that one curve will apply to the entire flow regime of a stream. Only in cases where a station is governed by some natural or manmade control with weir-like characteristics, is it possible that a single curve is likely to represent an entire flow regime. Where such weir-like conditions are not present, it is customary to represent different stage levels and/or different flow conditions with separate curves.

In order to apply several curves to one stream, the data must be divided into smaller groups or *data sets*, representative of particular stage levels or flow conditions. Once the data sets are established, each is modelled individually in the same manner as the previous data set containing all available measurements. This ability to segregate data provides *the second level of the analysis hierarchy* and is shown as Phase 2 in Figure 4. It is justified as the next logical step in developing a S-D relationship since measurements are collected under various conditions, which can dramatically influence the shape of the rating curve, as was discussed in Chapter 3.

Section 3.3 discussed the concept of section and channel controls and their relative influence on S-D relationships. The geometry of a particular channel, its flood plain and the nature of flooding in the stream are some of the basic factors which can determine the overall relationship between stage and discharge. When presented graphically, the relationship may display rough transitions between different flow regimes, making mathematical modelling difficult. Similarly, rising and falling stage under flood conditions generally display a hysteresis loop, which can not be represented with a simple mathematical function. Therefore, in each of these cases, it is necessary to analyze only those measurements which are observed under a particular condition or specific water level range.

Even within the boundaries of specific flow conditions and water level ranges, measurements may still be influenced by random or unexpected conditions. Measurements taken in the presence of such parameters may appear as ***outliers*** on the graphical S-D plot. An ideal plot will not contain any outliers since the data modelled should be those of one particular condition, with all other variables (parameters) held constant. The presence of such outliers in a S-D relationship is analogous to the unexplainable error associated with a laboratory experiment performed under less than perfectly controlled conditions.

In S-D analysis, conditions are impossible to control but the errors however, are quite often explainable. Therefore, segregating these explainable outliers from measurements not influenced by physical parameters is likely to produce a smoother curve of observations, which can be modelled for use under specific "pseudo controlled" conditions. This segregation of explainable outliers comprises *the third level of the analysis hierarchy*. The influenced measurements may be similarly grouped by their specific conditions and modelled for application under those more specific conditions, as shown in Figure 4.

5.2.2 Operational User Modes

The previous section introduced the concept of a hierarchical analysis scheme, making up the main SDDSS structure. A general description of each of the three user mode frameworks will now be presented, followed by a detailed discussion in the next section.

Curve Development, (CD)

The first main component of the SDDSS is the *Curve Development* (CD) module. It is responsible for the creation and storage of new curves for various data selection schemes. CD consists of four main tasks: data selection; data modelling; model evaluation and outlier analysis. The procedure captured in this module is also accessed by the curve modification module (discussed later), for use in reanalysing and updating existing curves.

Data selection involves establishing a selection scheme for retrieving measurements from the data base for modelling. Initially, the system begins with the selection of all measurements recorded for the current station, (Phase 1 of Figure 4). However, the user may skip this general selection scheme, and opt to select measurements on the basis of specific conditions present during gauging, (Phase 2 of Figure 4). Examples of these conditions include rapidly rising, falling or stable flow and various ice conditions. Once the scheme is established, the system retrieves the S-D data, writes it to an external file and displays it graphically.

The next task, *data modelling*, deals with the representation of the selected data. A regression analysis is performed on the data, establishing the mathematical equation or *model,* for the S-D relationship. In addition, the analysis outputs several statistical parameters for evaluating the model. The data modelling framework allows multiple models to be constructed for the data set, which may be compared for the best fit.

Model evaluation involves the comparison of multiple models with respect to the statistical parameters, generated by the regression analysis. One of three parameters is used for the comparison of models.

Finally, an *outlier analysis* is performed on the best fit model. The objective of this task is to establish the condition(s) which may have caused a measurement to plot as an outlier. Each outlier is isolated from the data set and information regrading the conditions under which the measurement was observed, is referenced from the database. If no information is available, the system prompts the user with a series of questions, trying to establish the cause of the deviation.

Throughout the CD process, the framework generates these tasks in sequence but allows the user to revert back to the initial task (data selection) if the system or user feels the development is not progressing favourably. Generally, viewing the graphical presentation of the data set will reveal whether or not the data can be represented with one model. In the event that a data set exhibits a large amount of scatter, the user may opt to select specific measurements from the database, relating to a specific condition. Thus, the sequence of tasks may be restarted, beginning with the establishment of a new data selection scheme.

Curve Use

The second component of the SDDSS, *Curve Use* (CU), is dedicated to the computation of average daily discharges. Computations are made by applying water level records to appropriate S-D curves created in the first system component. A table of average daily flows for each month is displayed on the terminal monitor and is also available on hard copy.

The procedure begins by having the user select which curve to be used. Next, a file name is entered in which the water level recordings are stored. The water level records may represent an entire year of daily water levels or only selected months which may be specific to a particular curve. For days with active water level fluctuations, the file format has been designed to allow for multiple water level records for one day. Once the file name is entered, an external program is then executed by the system to compute the table of average daily flows. The system displays the table to the user on the terminal monitor, offers to print a hard copy if so desired.

Curve Modification

The third component of the prototype, *Curve Modification* (CM), is concerned with updating curves as additional S-D data becomes available.

Curve modification begins with the selection of a curve and retrieving all S-D measurements matching the curve definition scope. As measurements are retrieved, old measurements (those used in the initial creation of the curve) and new measurements are sorted and stored in two external files for plotting. Upon completion of the retrieval, both sets of measurements are plotted in different colors with the curve. Upon viewing the new measurements, the user may decide to re-model a new curve based on the addition of the new measurements. In this case, the sets are combined and the system calls the curve development system.

In the case that the new measurements are scarce or deviate significantly from the curve, the user may choose to abort the curve modification procedure. Such action may be chosen in order to either create an entirely new curve or wait for additional data to be collected. The latter case may be necessary if a shift in the curve is suspected, with very little data to substantiate the assumption.

5.2.3 Modelling and Evaluation

The statistical process of curve fitting can be very complex. Therefore, the design of the SDDSS prototype has tried to organize the fitting process into a series of simple steps, in order to simplify matters in the early stage of system's overall development. The modelling process described below, is contained within the framework of the CD module.

The first step of modelling involves linearizing the S-D data by transforming one or both of the variables. In the current development state, the SDDSS uses a simple power transformation of the independent variable, stage. Transformed stages are computed by raising the measured stage values to the power of a value "T", ranging from 1.0 to 4.0. This value range is based on experimentation with various rating curves, developed for Manitoba rivers by the Department of Environment. At the present time, other transformation methods are not available in the SDDSS. However, the SDDSS framework was developed with an open architecture to the allow for additional inclusion of different transformation methods. Optional transformation methods were not included in the current system due to time constraints and due to the fact that the emphasis of this work was to provide "logical framework", demonstrating the entire stage-discharge analysis process, (as stated in Chapter 1). Future enhancements to the current transformation procedure, as well as additional methods of transformation are discussed in Chapter 7.

Next, a simple (linear) regression analysis is performed on the transformed stage and discharge data. Output from the analysis includes the *coefficient of determination* (R2 value), the *standard error of estimate* (SEE) and *the number of outliers* (NOO). The first two parameters are defined below, taken from Neter, Wasserman and Kutner (1989).

The coefficient of determination is computed from the standard statistical equation,

$$R^2=\frac{SSTO-SSE}{SSTO}=\frac{SSR}{SSTO} \qquad [1]$$

where SSE is the sum of squared variations due to model error, SSR is the sum of squared variations due to the regression and SSTO is the total squared deviation. These terms, used in equation [1], are calculated as follows,

$$SSR=\Sigma(\hat{Y}_i-\bar{Y})^2 \qquad [2]$$

$$SSE=\Sigma(Y_i-\hat{Y}_i)^2 \qquad [3]$$

$$SSTO=\Sigma(Y_i-\bar{Y}_i)^2 \qquad [4]$$

and where

$$\bar{Y}=\sum(\frac{Y_i}{n}) \qquad [5]$$

$$\hat{Y}_i=b_0+b_1X_i \qquad [6]$$

where X_i is the independent variable (transformed stage data) of the regression analysis and Y_i is the dependant variable (discharge data).

The Standard Error of Estimate is calculated as follows,

$$SEE=\sqrt{\frac{\mathrm{SSE}}{(\mathrm{n}-2)}} \qquad [7]$$

where n is the sample size of the regression analysis and SSE is defined as above.

Graphically speaking, the Number of Outliers is defined as the number of data points which lie outside allowable error limits defined for the model. In the current design of the system, the allowable error limit for any particular model is supplied by the user as a percentage. A default value of 30 percent is provided on the basis of experience with several stations. Data points are then identified as outliers if the percent error between the measured discharge and the model estimation of discharge exceeds the allowable error limit.

These three output parameters are used within the system as a means of measuring the goodness of fit of the model to the data, and will therefore be referred to as *evaluation parameters*. Each is temporarily stored in the system for evaluation later in the Curve Development process.

Within the curve development framework, the two main steps of modelling, (data transformation and regression analysis) may be repeated multiple times to allow for curve fitting using different transformation exponents. The output parameters from each regression analysis, are all stored within the system with their respective transformation exponents and a model identification label, for later comparison. Future development of the system will include an automated routine to perform this iterative process, and more elaborate methods of modelling, (see Chapter 7).

After performing one or more iterations of the modelling process, with varying transformation exponents, the next step requires a comparison of the models with respect to the three evaluation parameters. In the current version of the system, comparison of models is required to be performed by the user. Depending on the data set and the transformation exponents used, models may or may not differ significantly with respect to one or more parameters. For example, the R2 parameter may indicate an insignificant difference between two promising models whereas the NOO values for the two models may be significantly different. In this case the NOO parameter may be used as the deciding factor in selecting the "best fit" curve. Similarly, parameters may contradict each other with respect to comparison of "goodness of fit". It is the responsibility of the user to compare the relative differences in the parameter values for selection of a "best fit" curve. Therefore, the user must have a good understanding of the statistical parameters; their meanings and relative importance. A major goal of future development of the system will be directed at automating this comparison procedure, (see Chapter 7).

5.3 Technology

Although the concepts of DSS have been around for many years, only in the past decade has computer technology evolved to the stage where it could challenge the many imaginations of its users. The design of computers with greater processing speeds and memory capacity seems to be continuously accelerating. Computer programmers and application developers have more power on their desktops today, than most companies or government departments had in their offices only ten years ago. Combined with an equally fast development of specialized computer software and support devices, today's computers are providing numerous possibilities.

One might think that such advanced technology comes at a very high price. This may be true to a certain extent, in the case of new innovations. However, one dominant attitude prevails among all computer buyers; no matter how fast and powerful one machine or software is today, there will inevitably be one that is even more so tomorrow. The reality of this statement, combined with the extremely competitive market, has made computer hardware and software products evermore affordable for the majority of consumers.

The following sections will introduce several products used in the development of the SDDSS prototype. These products have been chosen on the basis of their widely accepted reputations and capability of integration with other products.

5.3.1 Operating Platform

At the end of section 4.1, the concept of DSS was introduced as an integration of three main components; a database system, a modelbase, and a system control manager. The tasks of each of these components may be performed by one or several software products and/or programming languages. Deciding on a method of integrating different software and programming languages can be a serious issue for DSS developers. Generally all software tools and languages use their own specific format for reading and writing data to and from files. Thus, the problem of making it possible to move data between several different tools or languages in one application, (known as *software integration*) becomes obvious.

In some cases, programmers may choose to avoid integration problems, and develop their application using one programming language or software tool for all components of the DSS. A disadvantage to this alternative is that it is usually a lengthy process since some tasks which may be easily performed by other specialized software tools, must be programmed from scratch in the one unique tool or language selected. This method is typically used for creating full-blown operational systems from specific detailed plans, rather than for experimenting in the initial stages of system development.

The obvious alternative to the above method requires the developer to deal with the problem of integration of several tools or languages. This usually requires the developer to program customized routines to link the various tools, so that data may be transferred and shared in the application. Many commercial software is equipped with utilities for allowing data to be transferred between other well known tools, using standardized or common data formats. Thus using such highly integratable software is highly advantageous to a developer, since the amount of customized programming required is reduced. For this reason, integrating specialized software is a suitable method for early development of DSS applications.

Due to the amount of experimentation required in the early development stages of the SDDSS, it was decided that the system should be developed by integrating several different software and languages, while handling integration problems as they arose. In order to attain this type of software integration, a workstation platform was selected for the development of the SDDSS prototype.

The development of the SDDSS was performed on a Sparc 1+ workstation, by Sun Microsystems. The machine is equipped with a 340 megabyte (MB) hard drive for data and software storage and 28 MB of random access memory (RAM) for processing. Other peripheral equipment includes a 150 MB tape drive, a Sparc postscript laser printer and an Epson LX-810 dot matrix printer. In addition, the workstation is connected to the

University of Manitoba's mainframe Unix system, by means of an Ethernet connection. The OpenWin 2 (X Windows) graphical windowing environment was used as the operational platform in the SDDSS, for bringing the various software tools together.

Workstation computing environments, such as the Sparc 1+, possess several benefits (in comparison to the common desktop personal computer or *PC*) which facilitate the integration of multiple software components. To begin with, workstations are fundamentally equipped with extremely fast and powerful microprocessors than PCs, thus providing faster and more efficient mechanisms for managing large processing tasks. They are equipped with sophisticated mechanisms for multi-task processing, enabling several programs, processes and/or software to run simultaneously. Workstation platforms, such as the Sparc 1+, are also not as confined as PCs in the amount of *random access memory* (RAM) available and are easily expandable to larger amounts. Combined with user-friendly windowing environments, developers are able to program very efficiently and produce applications which incorporate the processing of simultaneous duties.

In addition, workstations generally feature large display monitors with high resolution screens, enabling windowing environments to utilize their full potential. Applications can employ several windows of significant sizes simultaneously, allowing users to visually comprehend information without having to resize or close windows.

Finally, the networking capabilities associated with most workstations are highly sophisticated, allowing developers to create applications which stretch beyond the boundary of office walls. Organizations using workstation platforms can link machines from opposite ends of a country with relative ease and also maintain applications from any remote site.

5.3.2 System Control

Having selected the operational platform for the DSS, the next step of the prototyping process involves the selection of the various tools, which will be assigned the tasks of the DSS system. Typically, the main task of a DSS involves the overall control of the system, performed by a *system control manager* (SCM). From an operational perspective, it is the SCM which is responsible for the coordination of specific processing tasks of the various DSS components and the integration of their information. It is also responsible for communication between the system and the user, through the use of devices such as mouse, keyboard, monitors and printers.

From the design perspective, the type of DSS architecture used to conceptualize the problem domain and the various types of components employed will determine how the SCM is represented in the overall system. It may be a programming language such as C or Fortran, capable of integrating various data structures and modelling routines. In other cases, developers often employ a specialized software product or "tool", which supports integration with other products. *Expert system shells* are one type of tool which have been gaining popularity over recent years with application developers. They specialize in providing programming environments for the modelling of real-life problems.

One of the largest dilemmas of the software engineering process, and similarly in the DSS development process, is that there are always inevitable mistakes to be made during the process. Fundamental mistakes are generally the result of misunderstanding between developers and end users and by poor or incomplete specifications provided by the end user (Sommerville, 1985). It is the occurrence of fundamental mistakes which often plague and slow down the development process. Mistakes such as these, often require developers to redesign and reprogram large segments of their systems or in some cases, restart from scratch. This common dilemma has been termed the "water fall effect" of software engineering, since the process is directed by an irreversible (gravitational-like) force.

Expert system shells are particularly useful in the early stages of DSS development. They provide developers with the ability to perform rapid prototyping of an application, exploring many facets of the entire problem domain, in a fast and comprehensive manner. As a result, potential problems can be recognized faster, without the need for exhaustive hours of programming. Depending on the degree of development problems, in the event that any exist, and the amount of reprogramming required, the prototype may either evolve into a finalized operational system, or discarded and used as a model for a new system, programmed from scratch.

The ES shell, **Nexpert Object** by Neuron Data Inc. (1991), was selected as an efficient tool for the development of the SDDSS. This decision was based on its reputation, availability and its ability to deliver completed applications in different (computer) usable forms, (ie: DEC-Vax, PC, MacIntosh and Unix). Nexpert provides several distinct representation mechanisms which allow developers to capture and model non-algorithmic, decision making problems.

In Nexpert Object factual, procedural and descriptive information of an expert's domain is modelled using the two basic structures of Nexpert Object: rules and objects. ***Rules*** form one of the basic elements of a *knowledge base*, which stores the domain knowledge of specific experts. Nexpert rules are made up of two parts. The first part or *condition block*, is made up of boolean expressions or *If-clauses* which verify a particular situation. They must be satisfied in order for the inference engine to activate the second part of the rule, the *actions block*. This second part of the rule may contain one or more *Do-actions* related to the specific conditions of the situation. In addition, all Nexpert rules are assigned a *hypothesis* which is established upon the conclusion of the condition block. Hypothesis may be used in rule conditions to form logical links between rules, thus enhancing the knowledge framework.

Descriptive knowledge is stored in the ***objects*** of a Nexpert knowledge base. The use of objects within Nexpert is analogous to that of variables in an algorithm, programmed in a language such as Fortran. Objects may contain any number of *property slots*, which describe the characteristics of a particular object. Nexpert's *classes* are used to group specific objects which share common characteristics or properties. Classes may be used to transfer knowledge among objects by means of the Nexpert *inheritance* mechanism.

The *inference engine* of Nexpert Object is the main element responsible for the processing of rules. Information of a problem domain is stored in the knowledge base as rules, with no particular solution path. It is the responsibility of the inference engine to determine the strategy and perform a logical search of the problem space, in order to reach a solution.

Creating knowledge-based applications in Nexpert Object may be accomplished by means of various graphical editing windows. Alternately, developers may program knowledge-base structures in Nexpert's proprietary language code. An *Application Programming Interface* allows developers to incorporate external routines (programmed in C, Fortran, Pascal, etc.) within their applications, or alternately, to embed Nexpert Object knowledge-base frameworks directly into their own object-oriented (C++) application code.

The design of the SDDSS employed Nexpert Object as the system control manager of the system framework. Thus, Nexpert provides the integration between the various elements shown in the system architecture in Figure 3. Each of the three user modes (see Section 5.2.2) is contained within an individual knowledge-base. The overall control of these knowledge-bases and external program calls are managed by an additional knowledge-base and the Nexpert Object inference engine.

5.3.3 Database Management

Developing a DSS for SDA requires the management of considerable amounts of streamflow data. The Oracle Relational Database Management System (RDBMS) by Oracle Corporation was selected to provide the database management component in the SDDSS prototype. This utility was selected on the basis of its widely accepted reputation as an industry leader in the domain of database systems. In addition, testing of the SDDSS during development required hydrometric data which was available and provided in the Oracle format from the Department of Environment.

One final reason for selecting Oracle was because it is directly supported by the Nexpert Object development tool. Data may be easily transferred between Oracle tables and Nexpert Object property slots, via the *Nexpert-Oracle Bridge*, provided by Neuron Data. Nexpert supports the use of SQL commands (discussed below) directly in its rule structures, to perform necessary read, write and miscellaneous maintenance duties in Oracle tables. The bridge thus provides an essential integration mechanism within the SDDSS.

Relational systems, such as Oracle, are made up of one main organizational structure; *the table*. Data records are stored in individual *rows* of a table, and organized by specific record information in table *columns*. Oracle uses single relational commands to retrieve, update and delete records in tables. These commands make up the ANSI standardized *structured query language* (SQL, pronounced "sequel") used in all relation database management systems. Using SQL, users specify *what* actions are to be performed on data tables, rather than *how* the actions should be performed.

5.3.4 User Interface

Another essential characteristic of a DSS is its ability to communicate with its end user(s). This characteristic alone may determine the ultimate success or failure of a DSS, since a system which is difficult to use, most likely won't be. Therefore, the user-interface component of a DSS must present information which is easy and simple to comprehend. *Graphical user interfaces* (GUI) incorporate computer graphics and colors to display information to users in logical and visually pleasing formats. GUIs also emphasize the simplification of user input by means of menus with selection items and buttons. Data and results are presented as pictures to improve user comprehension.

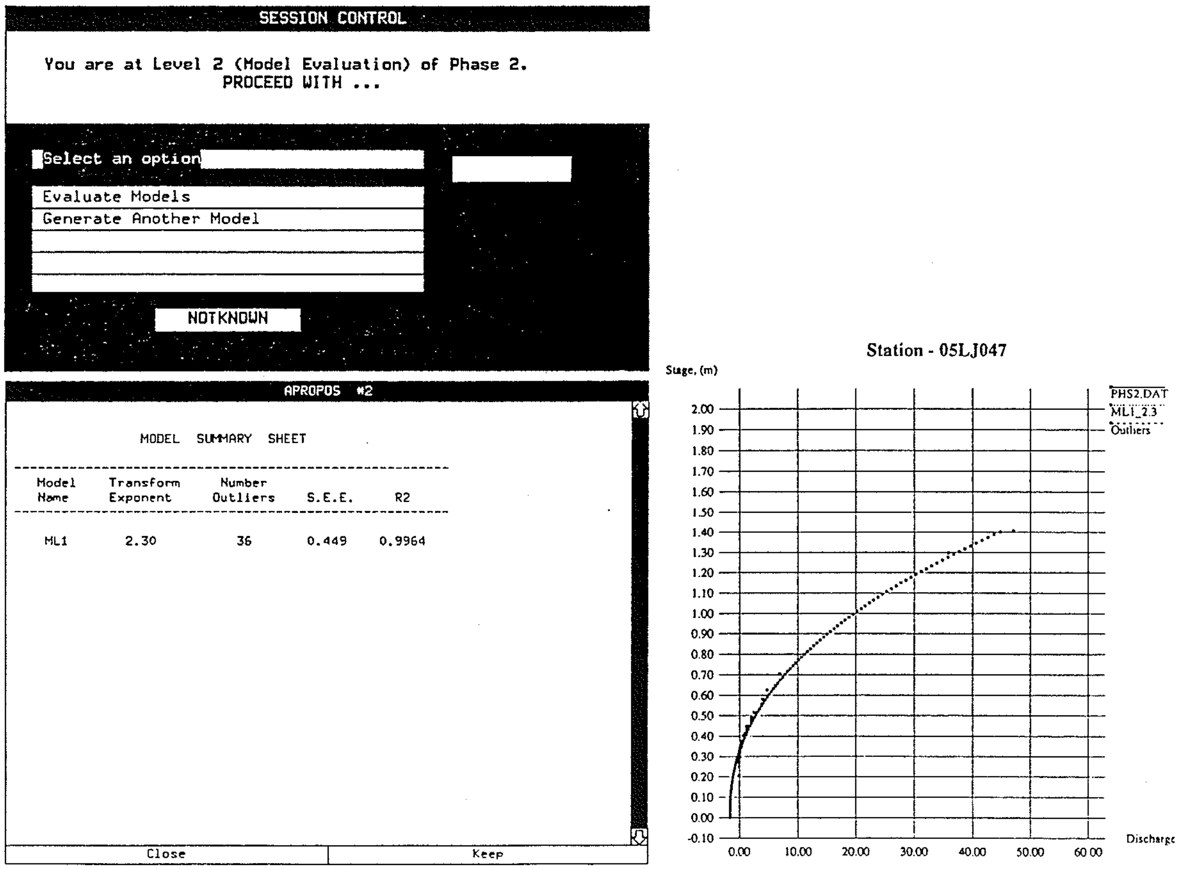
Several of these GUI concepts were incorporated in the SDDSS to demonstrate their potential application. S-D data and rating curves are displayed visually using an external graphing utility called *XGraph* (Harrison, 1986). This program was developed for use on X Windows Systems. The XGraph window is capable of presenting multiple sets of data simultaneously, each represented in a different colour.

Two additional windows are provided by Nexpert Object, for the user interface. User input is handled exclusively by the *Session Control* window, which is responsible for prompting the user for input and providing options to be selected whenever possible. Nexpert *Apropos* windows provide the second window for displaying processing messages and numerical results to the user.

The two Nexpert Object windows and Xgraph window make up the basic "*workbench*" of the SDDSS prototype, shown in Figure 5. Nexpert Object directly controls the information displayed in the two windows on the left of the workbench. It is also responsible for the arrangement of S-D data and curves, and subsequent calling of the XGraph window for presentation.

5.3.5 Modelling

The final component of the DSS, deserving particular attention is the modelling facility. As was mentioned previously, the SDDSS represents S-D relations as mathematical (regression) models. With respect to the prototype design, the modelling



SESSION CONTROL

You are at Level 2 (Model Evaluation) of Phase 2.
PROCEED WITH ...

```
                 MODEL  SUMMARY  SHEET
-------------------------------------------------------
 Model   Transform    Number
 Name    Exponent     Outliers    S.E.E.     R2
-------------------------------------------------------

  ML1      2.30          36        0.449    0.9964
```

Figure 5. SDDSS Workbench.

procedure is an external process, called from the system control component.

This modelling task is performed by a Unix script and Fortran program. Nexpert Object is responsible for executing the script with a string of necessary command parameters. A Unix script is simply a collection of Unix commands, executed sequentially. It is the responsibility of the script to prepare an input file from the parameter string, execute the regression analysis (Fortran) program, and redirect the analysis output, for input by Nexpert once the script has terminated. It should be noted that this technique of combining of Unix scripts and Fortran programs is used repeatedly throughout the DSS, for other assorted tasks such as message production and formatting data.

5.4 Operational Structure

The previous sections discussed the framework of processes designed within the system and the technologies incorporated in the SDDSS. This section will discuss the various responsibilities of Nexpert Object, Oracle and the other programs in the overall framework. Each user modes and the main menu is introduced as a separate knowledge-base controlled by the Nexpert Object system. Two additional knowledge-bases make up the master system, controlling the loading, unloading and execution of all other knowledge-bases. This control structure is shown in Figure 6.

One note should be made regarding the following discussion of the system processes. All references made to processes performed by knowledge-bases are controlled by the Nexpert Object tool.

5.4.1 Main Menu

The main menu knowledge base is automatically loaded when the master system is started. This KB has two main purposes. First, it is responsible for setting the current station for study. Second, it marks the starting point of the three user modes. It also contains the framework for the *Curve Use* module, which is discussed later in this chapter.

When started by the master KB, the first task of the main menu KB is to establish the current study station. The KB performs a sequential query of the Oracle table containing the list of available stations in the system. As each station is read its identification number and name are printed to a Unix file. After all stations are printed, the file of station names is displayed. The user is then prompted to input the identification number of the station to select as the current study station. A verification of the input is performed by the KB and if accepted, the current study station is appropriately set.

Setting the current station involves establishing the Oracle table names holding all data corresponding to the selected station. Each individual station has a group of tables dedicated to it uniquely. Tables include information such as S-D observations, curve definitions and observation attributes and usage. One main station list table contains the available station name, identification numbers and the names of the information tables associated with each station. When setting the current study station, the KB retrieves the list of table names into specific system variables. These variables provide the table names to be queried by the system control manager, when information is required to be retrieved.

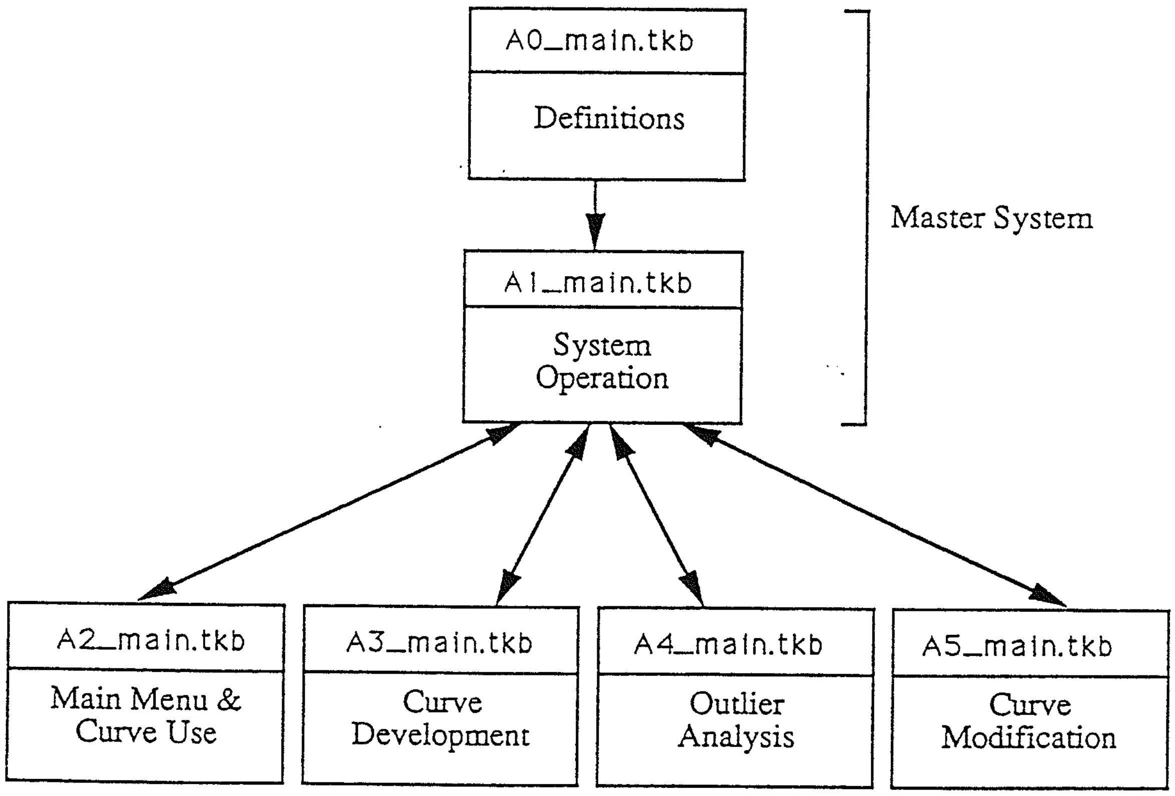


Figure 6. Nexpert Object organization of SDDSS.

5.4.2 Curve Development

The *Curve Development* (CD) module is comprised of two individual KBs, one for the modelling process and the other for outlier analysis. Each is loaded and executed independently from the main system KB. Upon selecting Curve Development from the main menu, the master system unloads the main menu KB and loads the modelling KB. Figure 7 shows a diagram of the entire CD process.

Modelling

Once loaded, the modelling KB begins with establishing the data selection phase. The data selection phase refers to one of three data query schemes, indicating the level of the CD process. Phase 1 and 2 deal with the selection of all data and specific data from the database, respectively. Data transferred from the outlier analysis back to curve development is referred to as the third selection phase. CD always begins with either phase 1 or phase 2 data selection, depending on the user's preference.

Establishing the phase number informs the modelling KB of what type of SQL query string to construct. Selecting phase 1 instructs the system to simply select all available S-D measurements from the database. The phase 2 option instructs the system to construct a more complex query statement to select specific measurements from the database, (eg. open water or complete ice cover measurements). In the latter case, the system prompts the user for details regarding the type of query that is desired. Further explanation of the types of Phase 2 queries, is discussed in detail in section 6.2.1.

In either case, once the selection phase and query string are determined Nexpert Object begins a sequential retrieval of data from the Oracle measurement tables. Only measurements matching the conditions of the query string are retrieved by Nexpert. Each S-D measurement record is retrieved into a temporary variable in Nexpert and processed before the next measurement is retrieved. Processing tasks include checking for maximum and minimum stage and discharge values and writing the stage and discharge values to a Unix file for plotting and modelling. After all measurements are retrieved, a summary sheet is displayed and the system calls the plotting utility (XGraph) to display

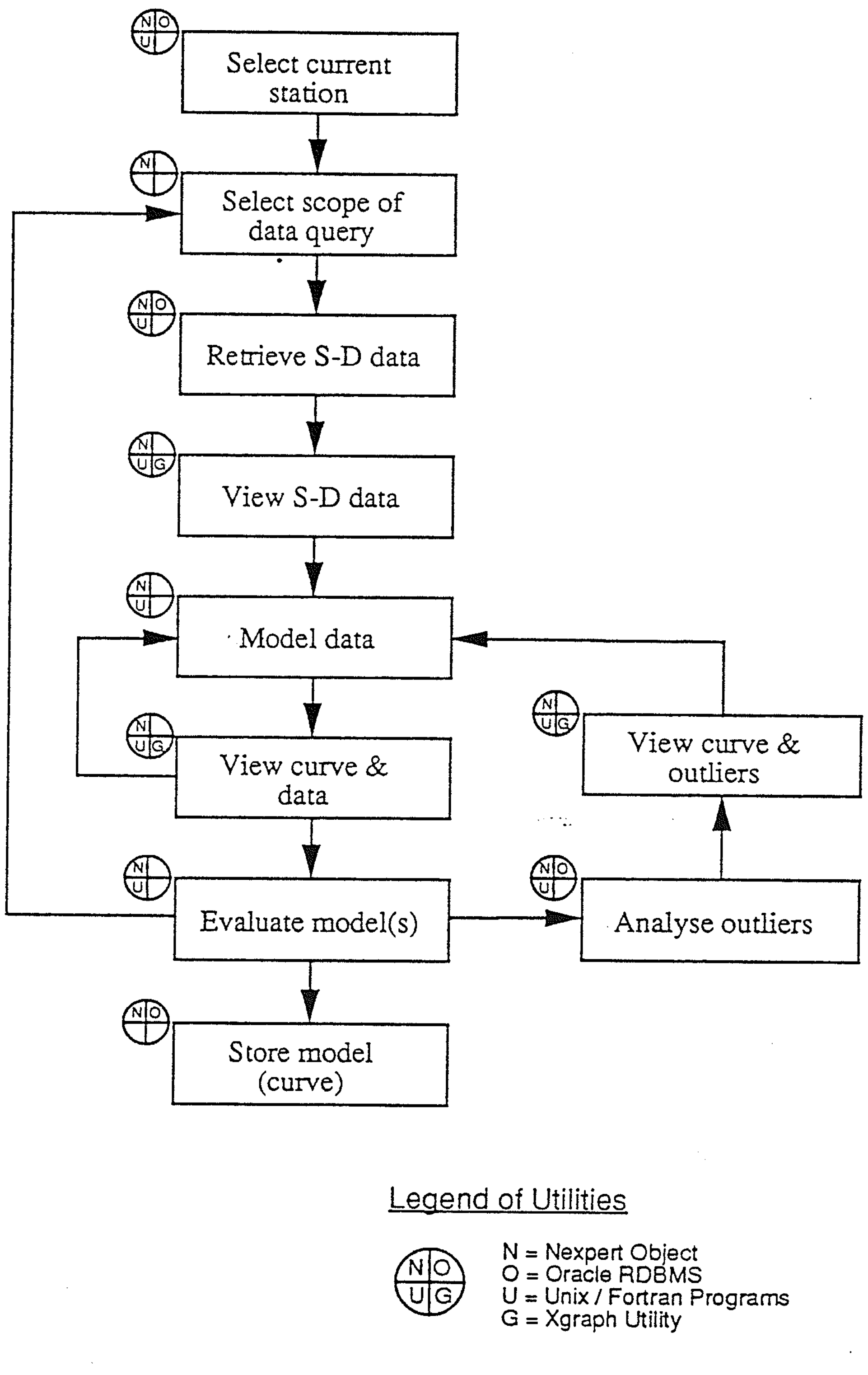


Figure 7. SDDSS Curve Development Process.

the S-D measurements selected. Data selection is thus complete and the user may continue to the modelling step or choose to re-select a different data set.

A brief introduction at the end of section 5.2 discussed the modelling process involving a series of three simple steps. In brief, in the first step (transformation), the system prompts the user for a real number and then checks that it is within the allowable limits (1.0 - 4.0). Next, the KB activates an external Unix script (LRM) to execute the linear regression analysis program. A string of input parameters is included when LRM is called from Nexpert.

LRM is responsible for three distinct tasks. Firstly, it prepares the input file for the fortran program, LR2, which performs the simple linear regression analysis. The input file is constructed from several input parameters transferred to LRM when it is called from Nexpert. The second task is to call LR2, to perform the regression analysis. Finally, LRM copies the output files of LR2 to filenames listed in the LRM parameter string.

The Fortran program LR2 has additional tasks, aside from computing the model parameters. Several evaluation criteria are also computed, including the number of outliers in the data set. An additional task is to create an "outlier file" of S-D measurements which do not fall within the allowable error limits of the curve. These limits are user specified, with a default value of 30 percent, as described in section 5.2.3. The allowable limit in percent is input to LR2 in its input file prepared by LRM. As outlying S-D points are identified, they are counted and copied to the outlier file for plotting later.

After the execution of LRM is completed, Nexpert retrieves the output from LRM and stores it temporarily as a separate model. The outlier file name is additionally stored with the model. The system then calls a second Unix script called GC, to generate a *curve file*, to be used in presenting the curve graphically. A curve file is simply an ASCII text file of X-Y coordinates, representing the curve equation. This is required since XGraph cannot plot mathematical equations directly. Nexpert prepares a string of parameters for GC including the transformation exponent, the model parameters and the maximum stage of the data set used in generating the model.

Similar to LRM, GC prepares an input file and calls a FORTRAN program (GCRV) to generate a file of X-Y coordinates based on the model equation. Once GCRV generates the curve file, GC writes it to an output filename (specified in the GC parameter string), and returns control to Nexpert. Finally, Nexpert executes XGR, a third Unix script, to plot the data, outliers, and the curve. In order to do this, the KB first merges the data set file, the curve file, and the outlier file into one file, with titles for each data set. Like LRM and GC, XGR receives an input parameter string which it interprets. This string contains the merged file name, the extreme axis limits and a graph title. XGR transfers these parameters to Unix system variables to be used as options in the XGraph command string. When executed, XGraph temporarily suspends input to Nexpert. The XGraph window must be closed in order for Nexpert to regain control.

As was mentioned in section 5.2, the process of data modelling and plotting may be repeated multiple times, in order to generate several models for comparison. Therefore, after each model is stored in Nexpert, the KB prompts the user to either continue with:

1) evaluating the models or;

2) generating another model.

Option two obviously restarts the modelling process described above. The first option exits the modelling process and starts the evaluation process.

The single objective of the evaluation process is to establish which of the multiple models (assuming more than one was generated) is the "best fit" curve. In order to compare the models, three evaluation criteria are presented to the user. R2, SEE and NOO values are displayed for each separate model after each call of the LRM script. A similar combination Unix script/Fortran program routine performs the formatting of the regression results in the model summary table. The system will allow the user to specify by which criteria the models should be judged. As discussed in section 5.2.3, the current system requires the user to compare the evaluation parameters and selecting one that will be used as the selection criteria. Once determined, Nexpert compares the curves with respect to the selected criteria and displays the name of the selected "best fit" curve at the bottom of the model summary table.

The selection of the "best fit" model marks the end of the modelling process. At this point in the system framework, the user may select one of three possible actions:

1) re-define the data set;

2) store the "best fit" model or;

3) perform an outlier analysis.

In the case of the first option, the system resets the CD module variables and automatically restarts in the phase 2 data selection scheme. Option 2 causes the system to store the model as a curve in the Oracle RDBMS using a Nexpert Object "atomic" write statement. The information stored includes the model parameters, the query definition, and limitations of the model's use. Control is then returned to the main system KB. Option 3 sets the outlier analysis as the next process to be initiated by the main system. The modelling KB is then exited and control returns to the main system KB. Once there, the main system KB loads and starts the Outlier Analysis knowledge base.

Outlier Analysis

To begin the *Outlier Analysis* (OA), the SCM performs the identical sequential query used in either the phase 1 or 2 data selection schemes of the modelling process. This time as each measurement is retrieved from the Oracle table, the measurement is analyzed with respect to the "best fit" model determined previously. To begin, the measurement is classified as either an "accepted" measurement, that is within allowable error limits, or an outlier. In the former case, the stage-discharge measurement pair is written to a data file of "accepted" measurements. If it is recognized as an outlier from the "best curve", the system tries to establish the cause of the deviation.

A window displays the outlying measurement values for stage and discharge, the date of the observation, and other related information. The OA system then checks the measurement information regarding conditions of the observation. If none exist in the database with measurement records, the user is questioned for the information. The goal of the questions is to classify the outlier in one of eight possible categories, relating to a specific cause of deviation. Possible causes of deviation and their respective codes include:

1) beaver activity (BV);

2) meander cut-offs in the channel (CO);

3) debris in the channel (DB);

4) rapidly falling stage (RF);

5) rapidly rising stage (RR);

6) weed growth in the channel (WE);

7) strong wind during measurement (WI);

8) flow from nearby tributaries (TR); and

9) unexplainable error (UX).

Once classified, the measurement is stored in a corresponding data file for plotting later. Outliers with multiple causes of deviation are coded as MU and stored in a separate file. It should be noted that other sources of deviation exist, however they were not included in the current system for simplicity. The system is in no way limited to the addition of other sources. Future work should consider grouping influences of deviation into categories such as backwater related, (eg. beaver activity, debris, tributaries, ice jamming, etc.), slope change related, (eg. meander cut-offs and channel modification) and friction change related, (eg. weed growth, and wind), for further analysis.

In addition, as each measurement is analyzed, (and categorized if it is an outlier) its measurement identification number is stored in a separate Oracle table with its attribute(s). The attributes are used to indicate the cause of the deviation, using the codes listed above. A curve identification number is also stored with the measurements, to distinguish which curve the measurement was used in developing. These records are stored for use in future developments of the overall system.

Upon completing the sequential retrieval of all the measurements, the KB presents a summary of the outliers identified during the previous analysis. The system also displays all measurements and the curve using XGraph. Assorted outlier categories are shown in different colours for identification, along with the accepted measurements and the curve created in the modelling process. The XGR Unix script is used for the plotting of the various S-D measurements. Nexpert merges the various outlier and accepted measures files with the curve file into one file for XGraph to read as input.

After viewing the plot, the system allows the user to either abort the curve development process entirely or continue with remodelling of only the accepted measurements. In either case, the Outlier Analysis KB is unloaded and control returns to the master system KB. If the remodelling option is chosen, the OA system sets the selection scheme to Phase 3, and instructs the master system to re-call the modelling KB. Once loaded, the modelling KB executes the modelling process loop as described earlier. The data set used contains only accepted measurements defined in the outlier analysis.

A final note should be made regarding the execution of the modelling process using Phase 3 data. Once the model evaluation has been completed and a new "best fit" curve is chosen, the system does not permit another outlier analysis. This restriction was included to avoid a second or several iterations of modelling and stratification of outliers.

5.4.3 Curve Use

The sole purpose of the *Curve Use* (CU) user mode is to compute average daily flows from a rating curve, and report them in tabular form. In the current design, the simple framework of this module was easily incorporated within the main menu KB. Later development of the system will see this mode as a separate KB as its options become more diverse.

Computing average daily flows consists of three simple steps. First, the curve to be used in the computation of discharges is selected. Next, a file name is entered by the user, containing a series of water level records. Finally, the discharge table is computed and displayed to the user. A hard copy of the table is also available upon request by the user.

To begin the process (Figure 8), the KB forms a summary of curves currently stored for the previously selected study station. The system uses a sequential retrieval to compile the curve listing. As each curve is retrieved from Oracle, its definition and attributes are written to an external Unix file. The previous integration method of combining the use of a Unix script and Fortran program is adopted here to perform this task. Once the list is compiled it is displayed to the user and the system prompts the user to select the curve to be used. Within the listing, each curve is catalogued by its curve identification number. When the user inputs a curve id number, the system verifies it and then retrieves all relevant curve information into a group of "current curve" variables.

Next, the system continues the process by prompting the user for the file name containing the water level records. The file format was designed as a preliminary prototype, proposing a possible structure to be used in the system. Each day in the records is included on one line in the file, with a maximum of 10 levels allowed per day.

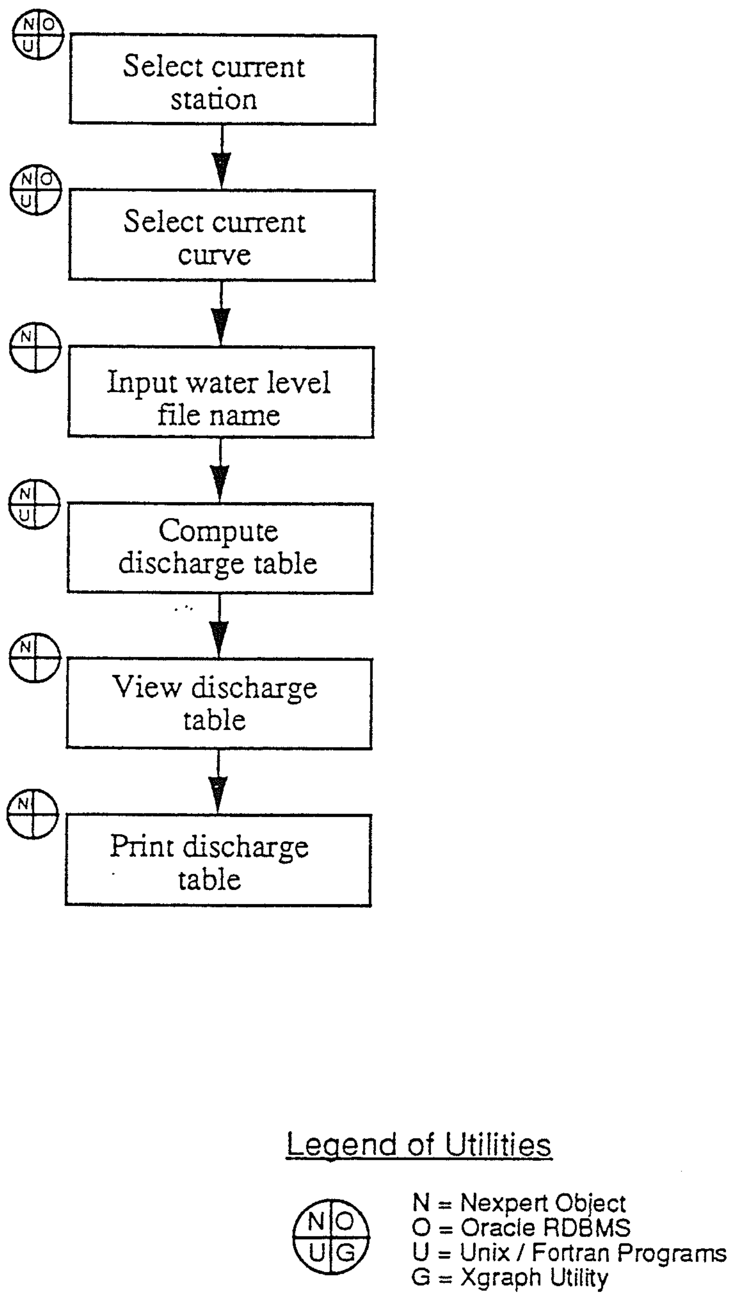


Figure 8. SDDSS Curve Use process.

Using multiple records in one day allows for the representation of rapidly fluctuating water levels. Months of records are separated by a single blank line.

Computation of daily flows and the formatted flow table is performed using a Fortran program. Again, a Unix script is called from the KB with a string of parameters containing the model parameters and the input file name of the water level records. The script interprets the input string and formats an input file for the fortran program. It then executes the fortran program which compiles the formatted table of flows. When completed, control returns to the KB, which in turn displays the table to the user in a viewing window. A hard copy is also made available to the user, after the display window is closed. Printing the table is performed using the Unix "lpr" command, directly from the KB. Output is sent to the local default printer. An example of printed table is shown in Figure 9.

5.4.4 Curve Modification

The *Curve Modification* (CM) module of the SDDSS is concerned with updating existing curves with additional S-D observations as they become available. The processing framework is shown in Figure 10.

Processing begins in the main menu KB, with the selection of a curve from the curve listing. The procedure for compiling and displaying the curve list, as well as the selection process is identical to that used in the previous section. Once the curve is selected it is set as the current curve and control is returned to the master KB.

Next, the master KB loads the Curve Modification KB. The latter is responsible for

```
26-JUN-92

                                         DAILY DISCHARGE IN CUBIC METERS PER SECON

DAY      JAN       FEB       MAR       APR       MAY       JUN       JUL       AU
--------------------------------------------------------------------------------

 1                                              31.512    20.663
 2                                              38.813    18.745
 3                                              62.776    17.145
 4                                              67.309    16.535
 5                                              64.332    14.452

 6                                              58.373    12.575
 7                                              55.878    10.118
 8                                              57.129    10.848
 9                                              51.229     8.418
10                                              46.407     6.092

11                                              48.218     5.718
12                                              52.722     4.010
13                                              55.393     2.639
14                                              50.681     1.995
15                                              45.743     2.068

16                                              40.518     2.552
17                                              37.244     2.729
18                                              33.794     2.501
19                                              33.918     2.056
20                                              30.634     1.653

21                                              34.025     1.605
22                                              38.307     2.356
23                                              42.464     2.505
24                                              37.816     2.941
25                                              34.889     1.889

26                                              34.283     1.366
27                                              29.662     2.104
28                                              26.700     2.134
29                                              25.465     3.106
30                                              24.379     4.145

31                                              22.806
--------------------------------------------------------------------------------

MAX                                             67.309    20.663
MIN                                             22.806     1.366
MEAN                                            42.368     6.255

MAXIMUM Daily Flow =   67.309 cu.m/s
MINIMUM Daily Flow =    1.366 cu.m/s
--------------------------------------------------------------------------------
```

Figure 9. Sample flow table.

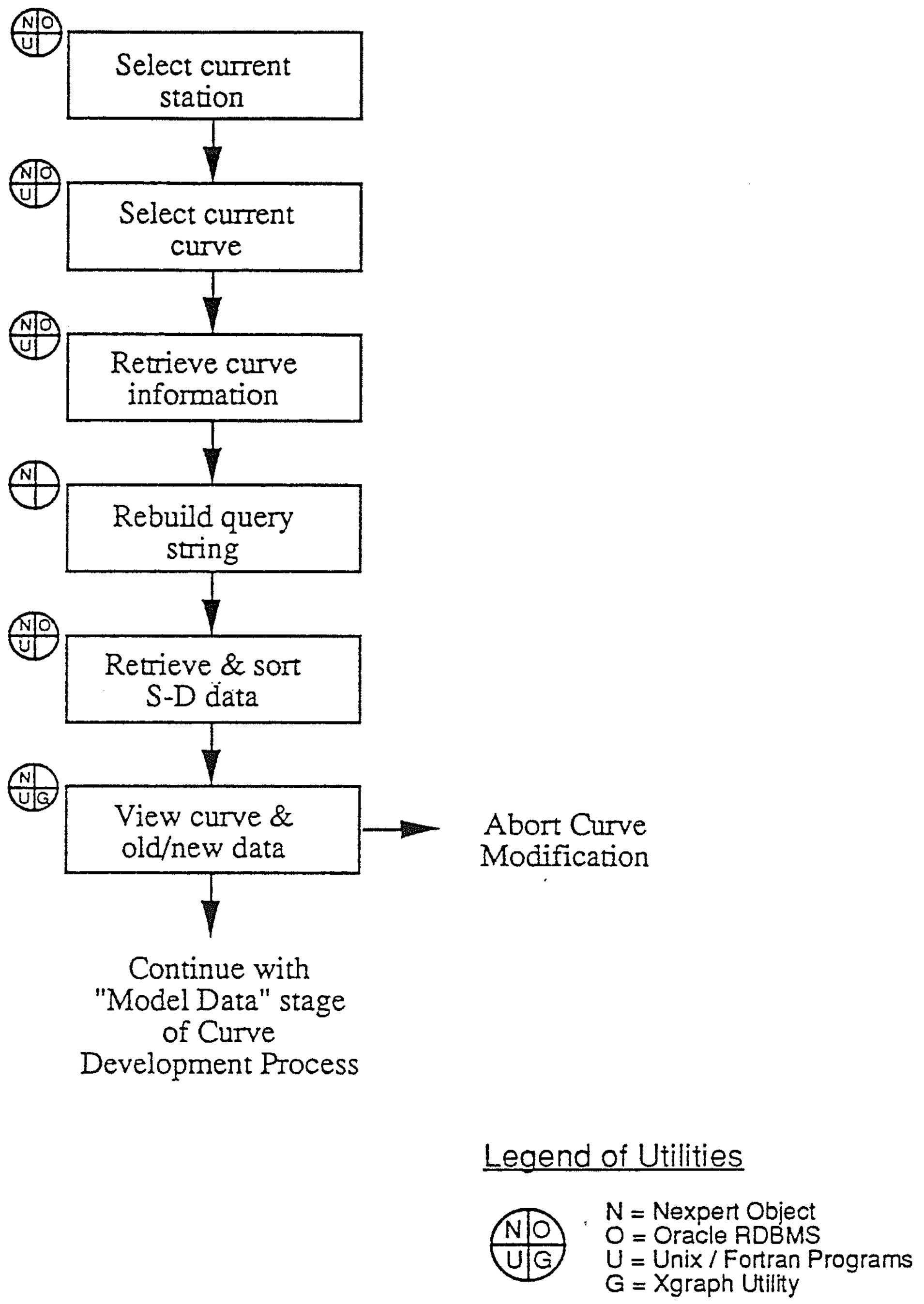


Figure 10. **SDDSS Curve Modification process.**

the retrieval of S-D measurements for updating the current curve. A query string is reconstructed from information retrieved with the curve. The reconstructed query is exactly the same as that used in selecting measurements, which in turn were used in defining the current curve.

During the retrieval process, the system classifies measurement as either old or new. Old measurements are defined as those that were used in the development of the current curve at some earlier date. To distinguish the transition between old and new measurements, the last old measurement date is stored with the curve information, at the time of curve development. Any measurements recorded after this date are classified as new measurements.

Classification is performed during a sequential retrieval of measurements from the appropriate Oracle table for the current study station. As each measurement is classified, it is written to one of two Unix files of old and new measurements. In addition, both old and new measurements are stored in a third file, used later in the modelling phase or the Curve Development module.

After the retrieval has been completed, a curve file is generated using GCRV (section 5.3.2) for graphical representation of the current curve. This file is merged with the two Unix measurement files for use as XGraph input. Finally, the system displays the curve with the old and new measurements shown in different colours.

At this point in the CM framework, the user may continue with the updating process or abort the system, without changing the current curve. The latter option simply sends control back to the master KB which ends the session.

Continuation of the process, involves implementing the Curve Development KB using the combined set of old and new measurements as the model data. The CM KB sets the CD phase number to 4 prior to exiting to the master KB. Setting phase 4 as the selection scheme indicates to the CD module that the data to be modelled has been transferred from the CM module. The modelling process within the curve development framework follows the same procedures as the phases 1 and 2. The only difference is that phase 4 indicates to the system that an existing curve is being modified rather than created from scratch. This condition is only relevant in the case where the user decides to store a curve modelled on the combined data set. In this case, the system performs a SQL "delete" command to remove the curve from the Oracle tables. The new model is then stored using the normal CD procedure.

6. CASE STUDY

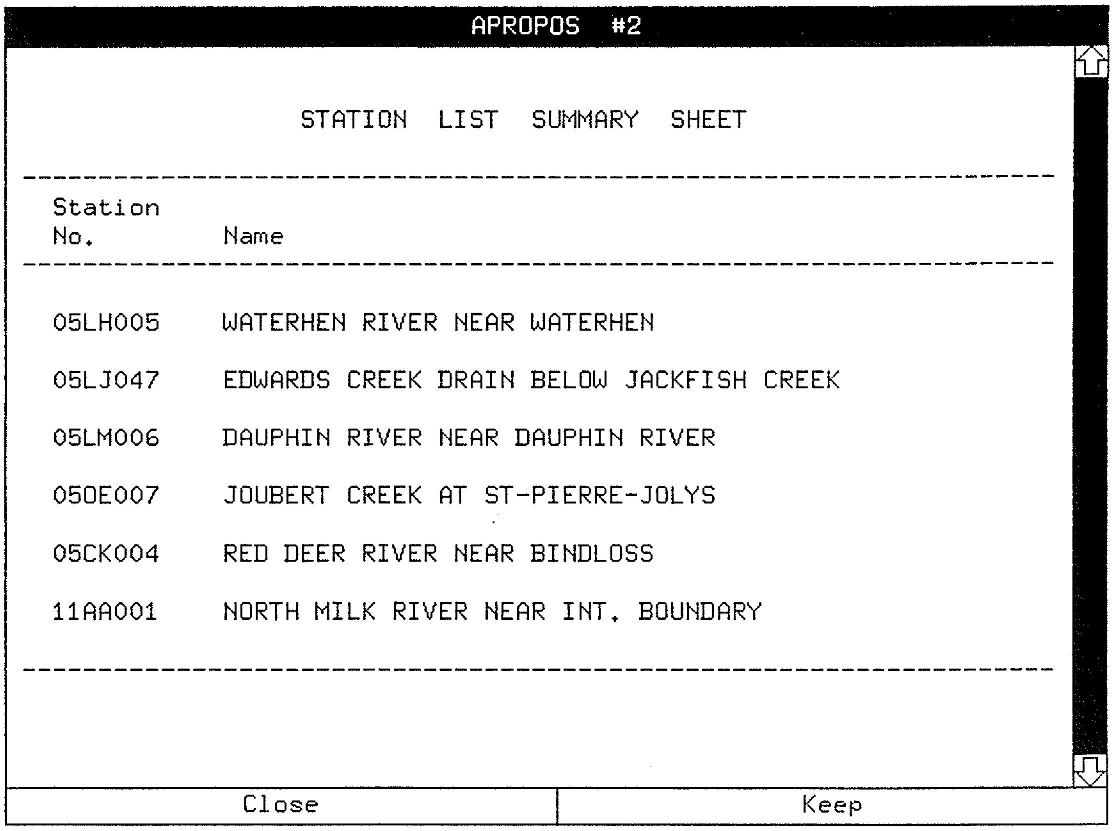
APPLICATION OF SDDSS TO EDWARDS CREEK

The purpose of this chapter is to demonstrate the Curve Development and Curve Use modes of the system. Data collected from Edwards Creek, located near Dauphin, Manitoba, was selected for the demonstration of the system components.

6.1 The Study Station

Upon entering the SDDSS, the user is immediately prompted to select a current station from a list of possible stations. For this study, data collected at three stations was obtained from the Winnipeg Water Resources Branch of the DOE. A list of these stations and their respective identification numbers are shown in Figure 11.

The gauging station at Edwards Creek Drain, below Jackfish Creek (No. 05LJ047) was established in the early 1980s and began producing streamflow data in 1981. Figure 12 shows a sketch of the site, including equipment and benchmarks. It was originally controlled naturally and later in 1983, a steel sheet pile weir was installed, 50 meters downstream of the gauge. The station is equipped with a cableway for high flow conditions, otherwise wading is the typical method for stream gauging. The relation for the station is categorized as stable, based on the specific gauge plots shown in Figure 1, (see section 3.4). Approximately 12 measurements are recorded each year; the majority of which are taken during open flow conditions. Water level records are often adjusted to compensate for frequent shifts caused by ice and aquatic vegetation. As of 1991, seven curves had been constructed to account for permanent curve shifts experienced, since the establishment of the station, almost 10 years prior.



```
                STATION  LIST  SUMMARY  SHEET
-------------------------------------------------------------------
 Station
 No.         Name
-------------------------------------------------------------------

 05LH005     WATERHEN RIVER NEAR WATERHEN

 05LJ047     EDWARDS CREEK DRAIN BELOW JACKFISH CREEK

 05LM006     DAUPHIN RIVER NEAR DAUPHIN RIVER

 05OE007     JOUBERT CREEK AT ST-PIERRE-JOLYS

 05CK004     RED DEER RIVER NEAR BINDLOSS

 11AA001     NORTH MILK RIVER NEAR INT. BOUNDARY

-------------------------------------------------------------------
```

Figure 11. Example station list.

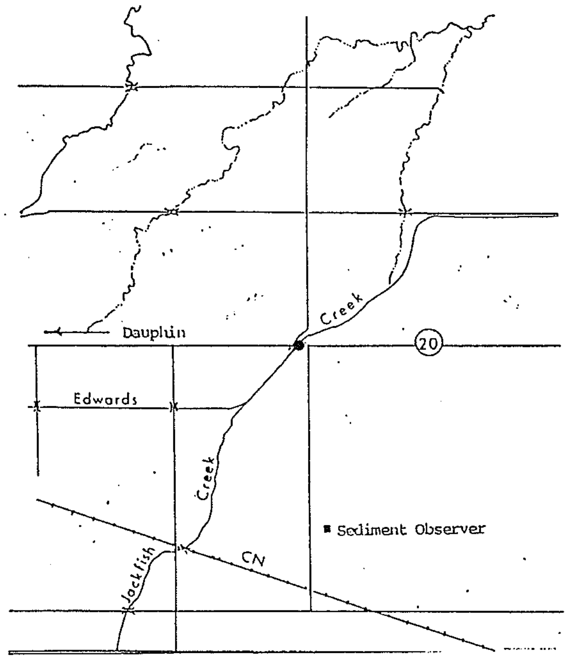


Figure 12 (a). Map of Edwards Creek area.

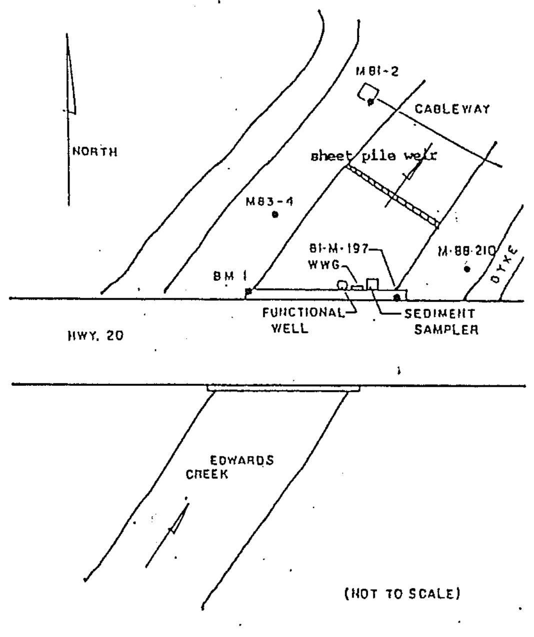


Figure 12 (b). Sketch of Edwards Creek gauging site.

6.2 Development of a Rating Curve

Selecting the *Make New Curve* option from the main menu starts the curve development system. An additional KB is loaded and the system presents two options. Curve development commences with the selection of stage-discharge measurements.

6.2.1 Selecting Measurements

Initially, two options are presented to the user regarding S-D measurement to be selected. The system allows the user to select either all measurements (Phase 1) from the database or select specific measurements according to one of several predefined schemes, (Phase 2). It is recommended that the user begin with the *All Measurements* option, and view the entire data set. Later, after the measurements are viewed, the user is given the opportunity to switch to the *Specific Measurements* option. Figure 13 is a plot of all measurements recorded for Edwards Creek, obtained using the former option.

Upon viewing this plot, it becomes apparent that there is considerable scatter. It is likely to be associated with backwater conditions created by rapidly rising and falling stages, and/or ice conditions. Terminating the display continues the session with the next step.

At the current state in the development process the user is prompted to either continue with the modelling stage or return to the selection stage to *Reselect Specific Measurements*. Modelling such scattered data is not likely to produce a meaningful representation of the true stage-discharge relationship. Therefore, the latter option was chosen in the case study to reduce the scope of the data selection process.

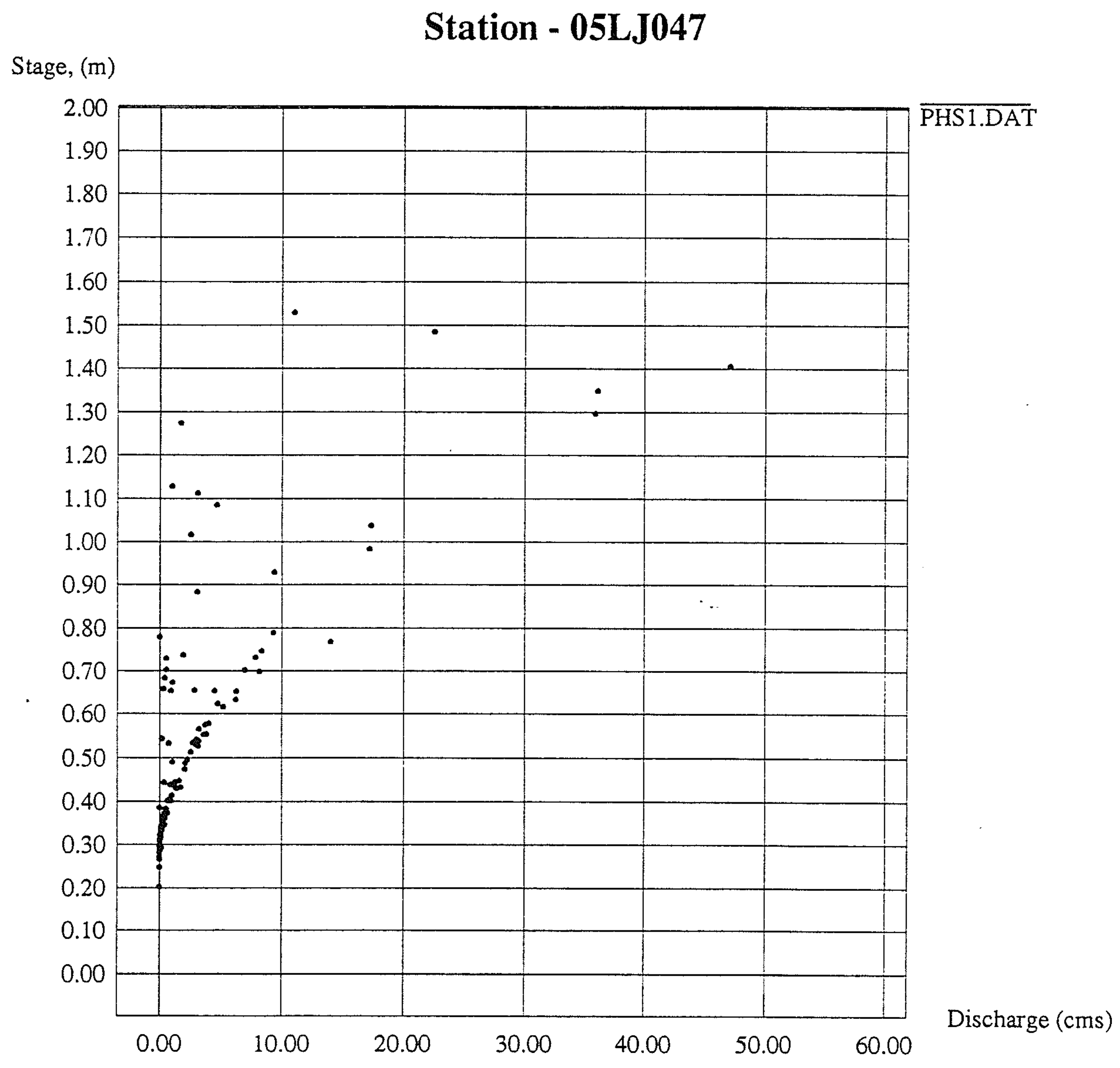


Figure 13. **All measurements (Phase 1) selected for Edwards Creek.**

Several selection schemes are currently supported in the SDDSS for selecting specific measurements (Phase 2). Each scheme is intended to focus on specific measurement conditions. The main schemes include:

1) ice conditions;

2) open flow conditions;

3) rapidly changing stage conditions and;

4) time range.

Unfortunately, the *remark* data and comments obtained from the DOE for this study were incomplete with respect to information regarding the first three selection schemes. The time range scheme uses the date of each measurement as delimiting information, which was included in the data obtained. Therefore, this fourth option was used for selecting specific measurements. It should be noted that the future success of the SDDSS will require a more sophisticated coding system for recording specific site conditions by technicians. Such a restructuring of a coding system for organizations like the Department of Environment, may cause significant implications within the organization, but none the less, will be essential to allow the SDDSS to perform the complex analysis procedures, necessary for accurate rating curves.

The *Time Range* option requires the user to specify two dates (month and day) in the calender year representing start and stop dates for the range. All measurements recorded between the two days (in any year) are selected for the current data set. Figure 14 shows the reduced data set, based on measurements recorded between the first day in May and the last day in August. This time range was selected as such to simulate the selection of

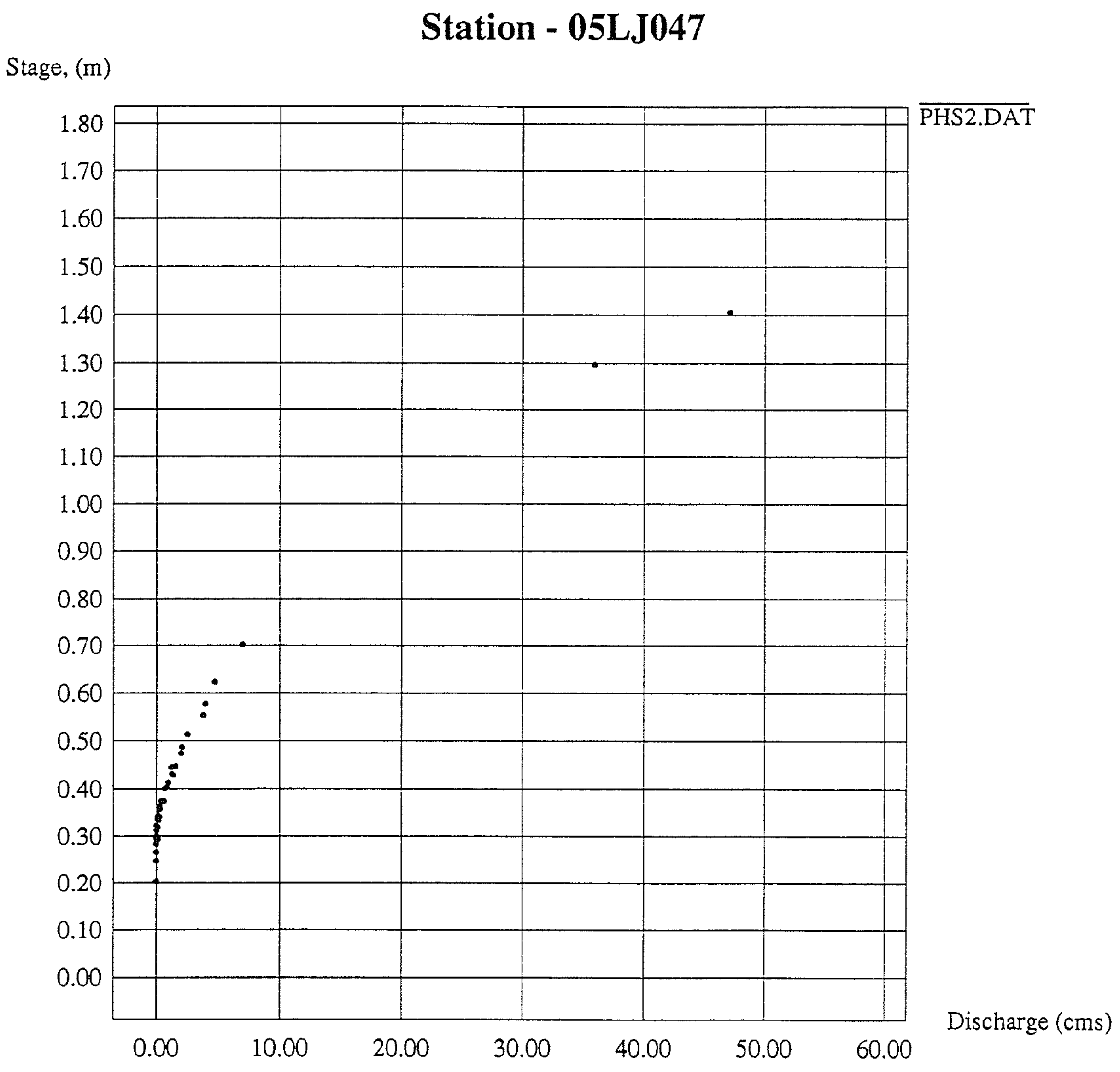


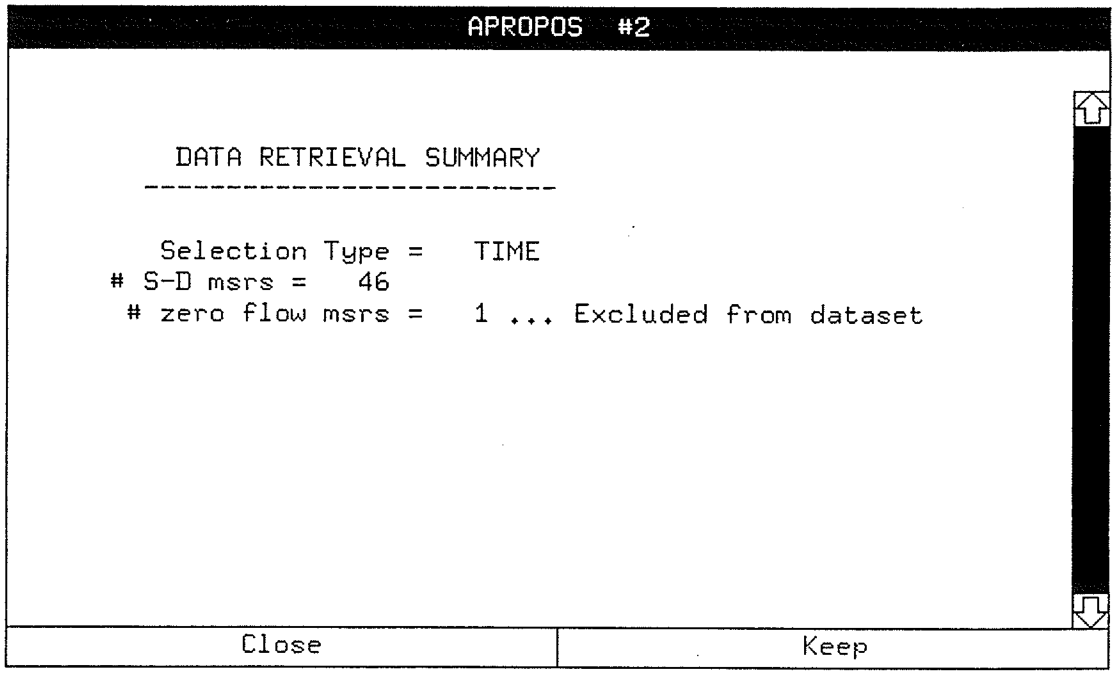
Figure 14. Specific measurements selected, May to August observations.

measurements recorded during open flow conditions. In comparison to Figure 13, the reduced data set displays a more definite shape. A summary of results from the data selection procedure is shown in Figure 15. The reduced data set shown in Figure 14 was accepted as the data set to be used in the next modelling stage.

It should be noted that the *Open Flow Conditions* option is intended to provide this type of selection scheme. In fact, a more accurate selection of data would be expected, since the *Time Range* scheme is based on dates and is therefore likely to erroneously exclude some open flow measurements recorded outside the time range or include ice conditions recorded within the time range. On the other hand, the *Open Flow Conditions* scheme bases its selection criteria on information codes stored with each measurement. As was mentioned previously, this coded information was not available at the time of this case study.

6.2.2 Modelling Selected Data

Continuing with the modelling procedure, the next information required from the user is an integer value for the allowable error limit. This value integer defines the tolerance, in discharge percent error, allowed before a measurement is considered an *outlier*. A default value of 30 percent is provided, as described in section 5.2.3. After supplying a value for the allowable error limit, the user must enter a real number between 1.0 and 4.0 (known as the *transformation exponent*), to be used in transforming the data set for the regression analysis. Once entered the system relays a modelling message to the user while it performs a simple regression analysis on the transformed data. When completed,

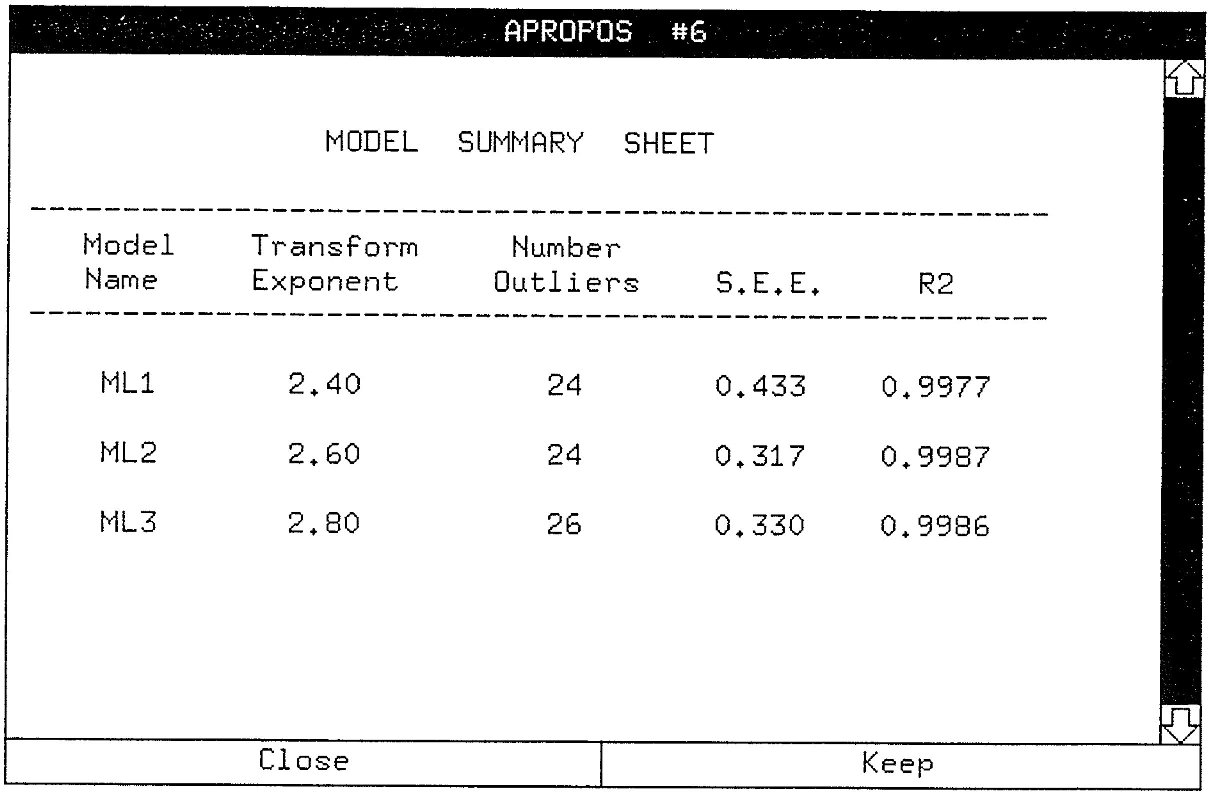


```
     DATA RETRIEVAL SUMMARY
    ------------------------

     Selection Type =    TIME
  # S-D msrs =    46
   # zero flow msrs =    1 ... Excluded from dataset
```

Figure 15. Summary of results from Phase 2 selection procedure.

a summary of the analysis results is displayed in the bottom left window. In addition, the data is re-displayed using XGraph along with the modelled curve. After viewing the curve produced by the system, the user may choose to repeat the transformation process to generate a slightly different curve.

Figure 16 shows a regression analysis summary for three models generated using the procedure described above. The outlier column of the summary reflects the number of measurements which did not fall within the allowable error limit, previously entered by the user at the beginning of the modelling stage. Outliers are defined by the absolute percent error between the true discharge measurement and the model discharge response.



MODEL SUMMARY SHEET

Model Name	Transform Exponent	Number Outliers	S.E.E.	R2
ML1	2.40	24	0.433	0.9977
ML2	2.60	24	0.317	0.9987
ML3	2.80	26	0.330	0.9986

Figure 16. Summary of SDDSS results for 3 models.

A plot of the second curve in the summary listing (ML2), is shown in Figure 17. It should be noted that the computer monitor uses different colors to distinguish the curve and data. Red points are used for discharge measurements that fall within the allowable error limits of the modelled discharge. Such points are referred to as "accepted" measurements. Measurements falling outside the allowable error limits (outliers) are displayed as blue points. Green points are used to represent the fitted curve.

Once the user is satisfied with the models generated, (having attempted several transformations using a range of exponents), control may be directed to the model evaluation stage in the process.

6.2.3 Evaluating The Models

To recap, at this stage in the process the system has generated three models for the current data set, selected on the basis of measurements recorded between May 1 and October 31. The model summary sheet displayed the regression analysis results for each model fit to the current data set. At the present stage, the system attempts to choose one of the models as the "best-fit", according to one of the three analysis parameters listed in the summary sheet. The current state of the SDDSS prototype prompts the user to specify by which analysis parameter the models should be judged. These options are displayed to the user in the session control window, shown in Figure 18 . Once chosen, the selected model is shown below the summary table and the system continues to the next task. For the case study, the ML2 model listed in Figure 16 was chosen as the "best-fit" model according to the *Standard Error of Estimate* criteria.

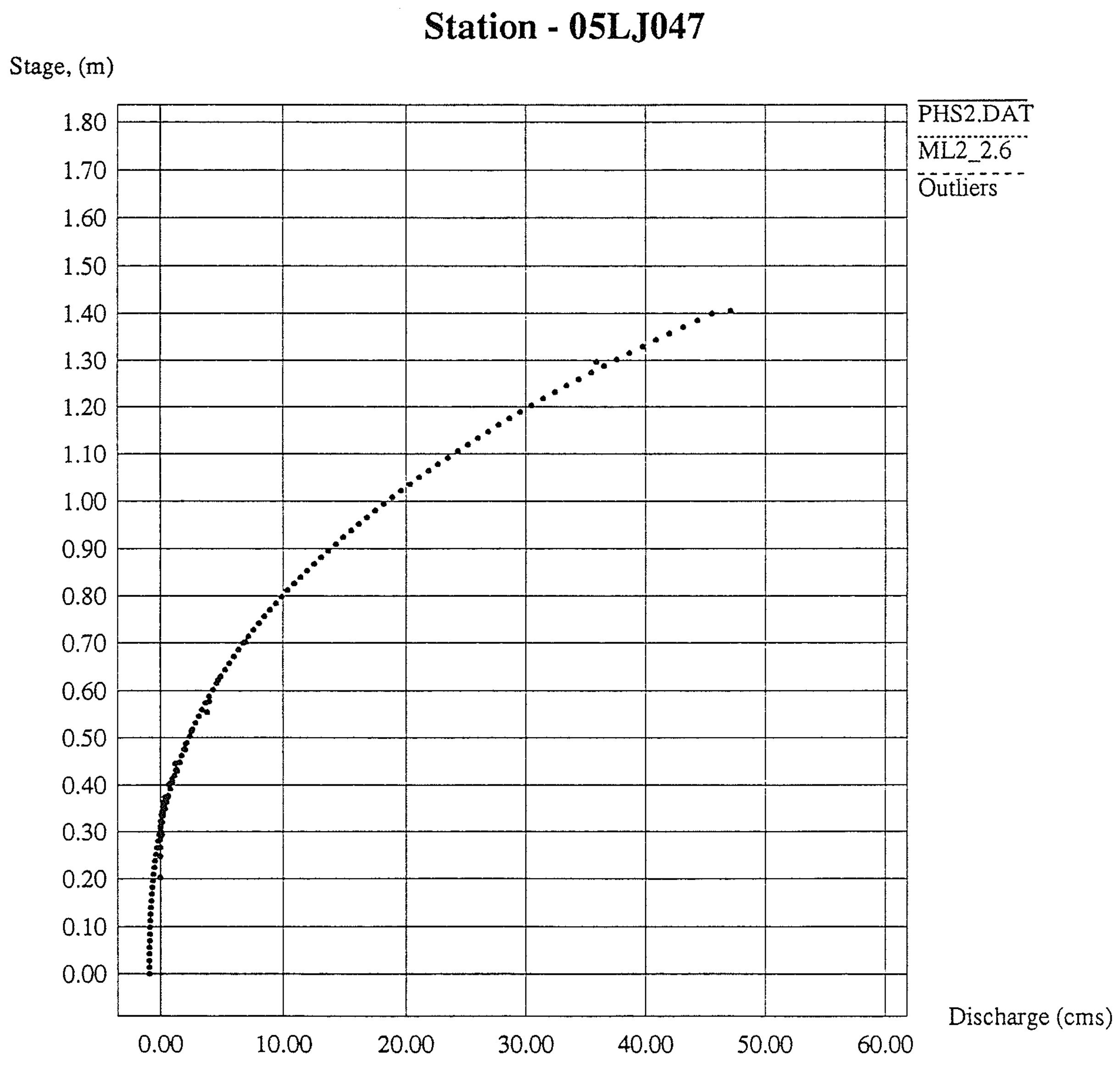
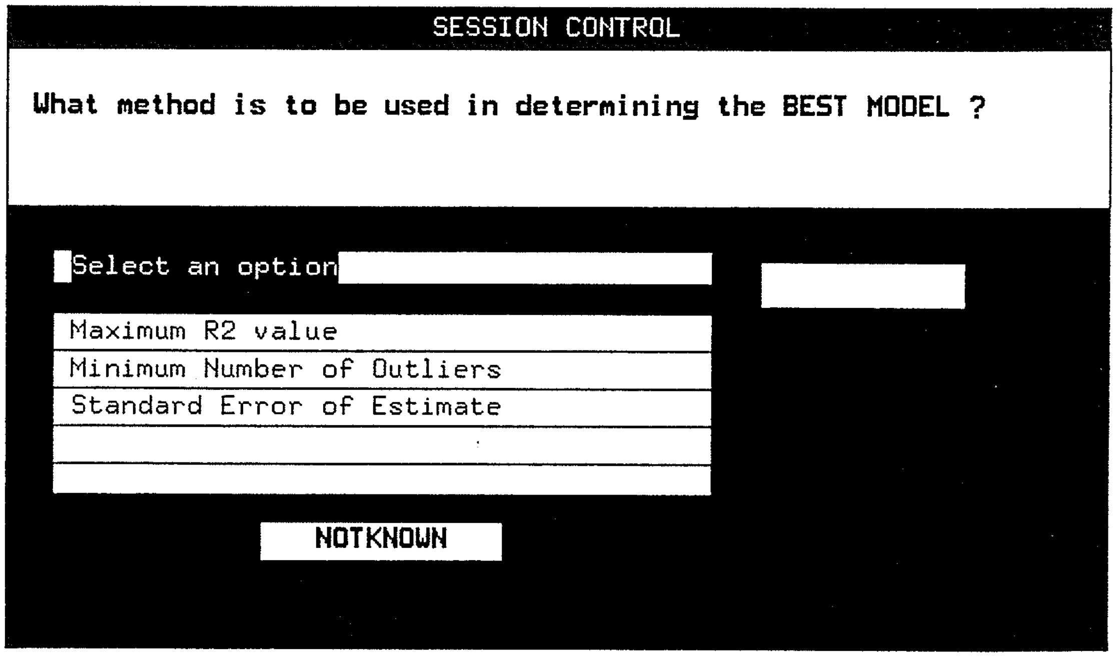


Figure 17. Model ML2, shown with May to October measurements.


SESSION CONTROL

What method is to be used in determining the BEST MODEL ?

Select an option

Maximum R2 value
Minimum Number of Outliers
Standard Error of Estimate

Figure 18. Session Control Window displaying options for model evaluation.

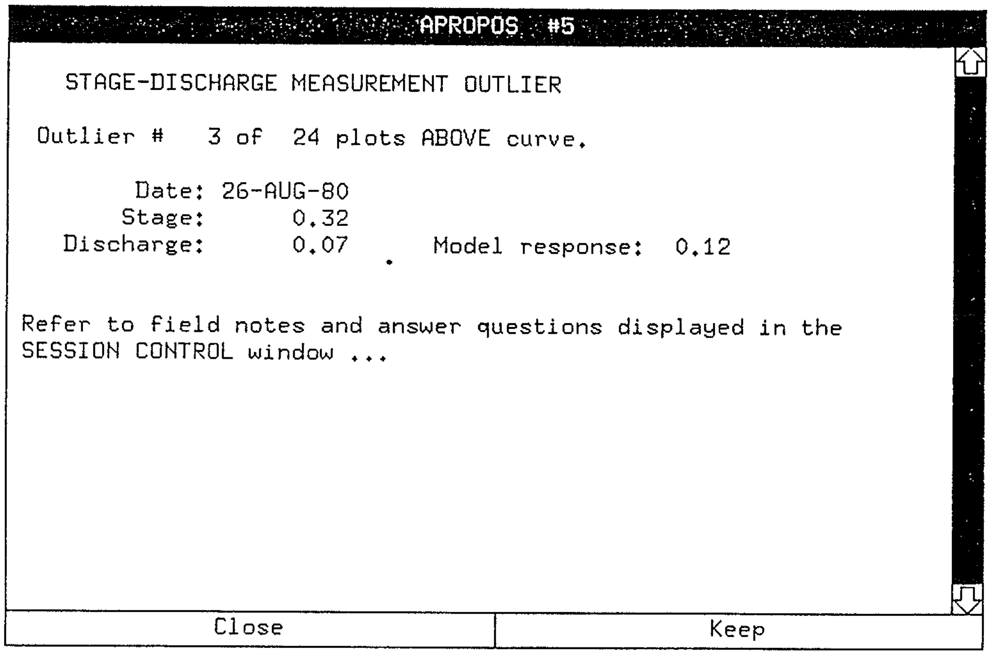
Next, three options are presented to the user after the selection of the "best fit" model. Option 1, *Identify Outliers*, begins an analysis of the outliers relative to the "best fit" model. This procedure is discussed later in this section. If the user is not satisfied with the results of the modelling procedure, the second option may be chosen to *Select a Different Dataset*. This option returns the user to the data selection stage, to specify a different selection scheme. All models generated and the current data set are cleared from the system, in this case. The third option, *Store Model*, may be selected if the user is satisfied with the best fit model. This option simply instructs the system to store the model in the data base. The SDDSS is then exited automatically.

To demonstrate the outlier analysis procedure, the first option was selected in the case study session.

6.2.4 Analyzing Outliers

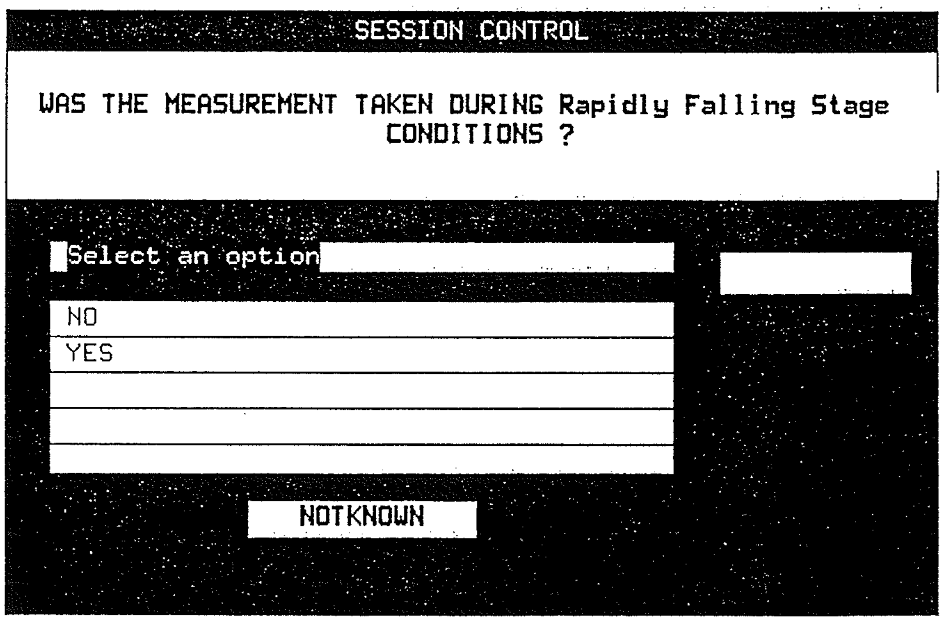
The outlier analysis is a somewhat automated process. As was discussed in Chapter 5, the analysis considers each outlier individually and tries to establish the cause of its deviation, from the best-fit curve selected. The system accesses measurement information for each outlier as it is encountered. In the absence of the necessary information, the system resorts to user input to establish the cause of the outlier's deviation.

In order to establish the cause of an outlier deviation, via user input, the system directs a series of questions to the user in the session control window. First, the date of the measurement and the type (either a left or right shift from the curve) of outlier encountered are displayed in the dialogue window as shown in Figure 19. Next, questions and possible responses are displayed in the session control window. An example question used in the analysis of outlier deviation is shown in Figure 20. Once all responses to the question series are completed, the system continues to flag outliers and repeat the question process until all measurements have been considered. Upon completion of the analysis, a report is displayed in the dialogue window, summarizing the classification of the outliers. A summary of the 24 outliers classified in the analysis, is shown in Figure 21.



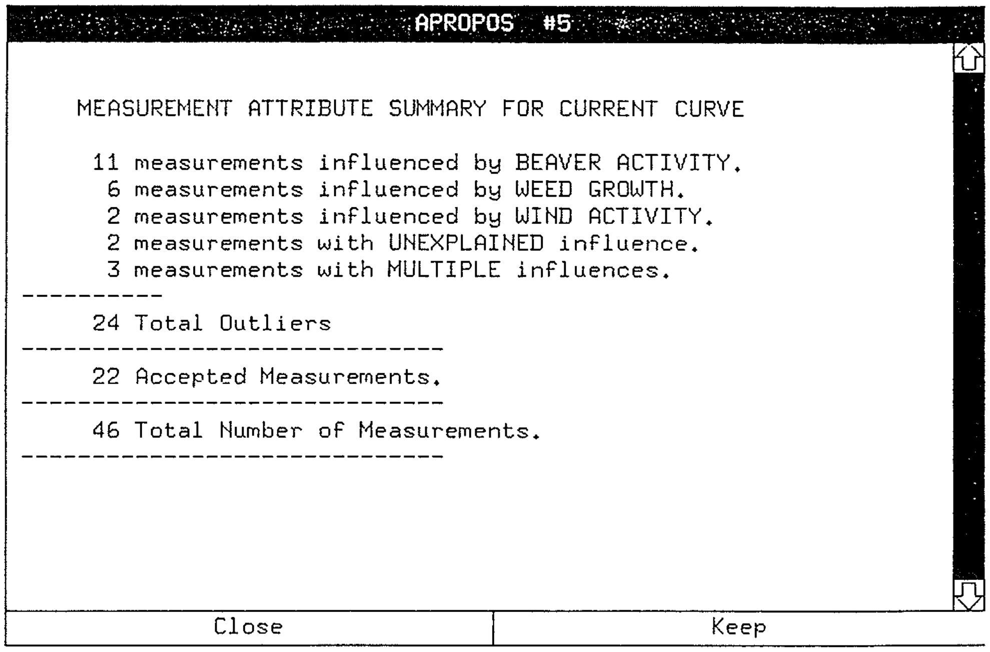
```
  STAGE-DISCHARGE MEASUREMENT OUTLIER

 Outlier #   3 of  24 plots ABOVE curve.

        Date: 26-AUG-80
       Stage:       0.32
   Discharge:       0.07      Model response:  0.12


Refer to field notes and answer questions displayed in the
SESSION CONTROL window ...
```

Figure 19. Dialogue window displaying outlier information.



WAS THE MEASUREMENT TAKEN DURING Rapidly Falling Stage CONDITIONS ?

Figure 20. Example questions used for analysis of outlier deviation.



```
    MEASUREMENT ATTRIBUTE SUMMARY FOR CURRENT CURVE

     11 measurements influenced by BEAVER ACTIVITY.
      6 measurements influenced by WEED GROWTH.
      2 measurements influenced by WIND ACTIVITY.
      2 measurements with UNEXPLAINED influence.
      3 measurements with MULTIPLE influences.
----------
     24 Total Outliers
------------------------------
     22 Accepted Measurements.
------------------------------
     46 Total Number of Measurements.
------------------------------
```

Figure 21. Summary of outlier analysis, showing causes of deviations.

The system then offers to re-plot the current data set with the outliers stratified into categories, corresponding to their cause of error. Three options are presented in the Nexpert session control window. If either option 1 or 2 is selected, the XGraph window is displayed. Measurements are shown in various colours corresponding to their classification. The legend in the top right corner of the XGraph window explains the colour coding. Selecting option 1 causes the current curve to be displayed with the measurements. Option 3 disregards plotting and continues to the next menu. A figure depicting this stratification has not been included here, as a color rendering was not available at the time of this publication's printing.

Three final options are displayed after the outlier plot has been closed. The main intent of the system at this point is to allow a new model to be fit to the accepted measurements only, (Option 2). The other options allow the user to abort all processing (Options 1) or review the stratified data plot (Option 3).

Continuing with the remodelling option repeats the modelling procedure discussed earlier in section 6.2.2. It should be noted however that the data set to be modelled here only contains those measurements which were not considered outliers to the previous "best-fit" curve. The previous curve is thrown away at this point and the accepted measures are remodelled. Again, several models may be generated for the accepted data set.

In this case, there are 24 outliers and 22 accepted measurements. This large proportion of outliers is due to the fact that the smooth parabolic curve cannot accommodate the sharp "bend" in the data, which occurs in the low flow regime. This indicates the need for representation of the stage-discharge relationship with multiple curve segments. From a statistical standpoint, there is little reason to continue with modelling, since more data would be removed than would remain. However, in order to exemplify the outlier analysis system, the modelling of accepted measurements will be continued. It is acknowledged that this method is tending to fit the data to the curve rather than the curve to the data. Reference to this problem will be made in section 6.4.

Continuing with the case study analysis, the outlier analysis identified 24 deviant measurements. The system was then instructed to return to the curve development procedure and remodel only the accepted measurements.

The remodelling procedure behaves exactly the same as that of phase 1 and 2 schemes. However, upon completing the modelling and the evaluation of the "best fit" curve, the system does not permit another outlier analysis. Instead, the user is only allowed to either abort the system or store the final curve.

6.2.5 Storing a Curve

Figure 22 shows the final curve produced from the remodelling procedure performed on the accepted measurements separated in the outlier analysis. This curve is based on the same data set used prior with the exclusion of the 24 outliers identified. The model was stored in the data base for future use in computing average daily discharges. It should be noted that the time range selection scheme was stored with the curve. Therefore, the curve should only be used to apply to water level records obtained for the specified period between May and October.

6.3 Using a Rating Curve

The "Compute Daily Discharge" option of the SDDSS system requires three simple steps. First, an input file of water level records must be prepared for use in the computation of daily flows. The format of the "water level file" was explained briefly in Chapter 5, and is intended to represent the format of data produced by digitizing a continuous water level record. A sample file has been included in Figure 23 for reference. It is assumed here that the user has prepared such a file, prior to entering the system and is now ready to begin the computation procedure. For the case study, the

Station - 05LJ047

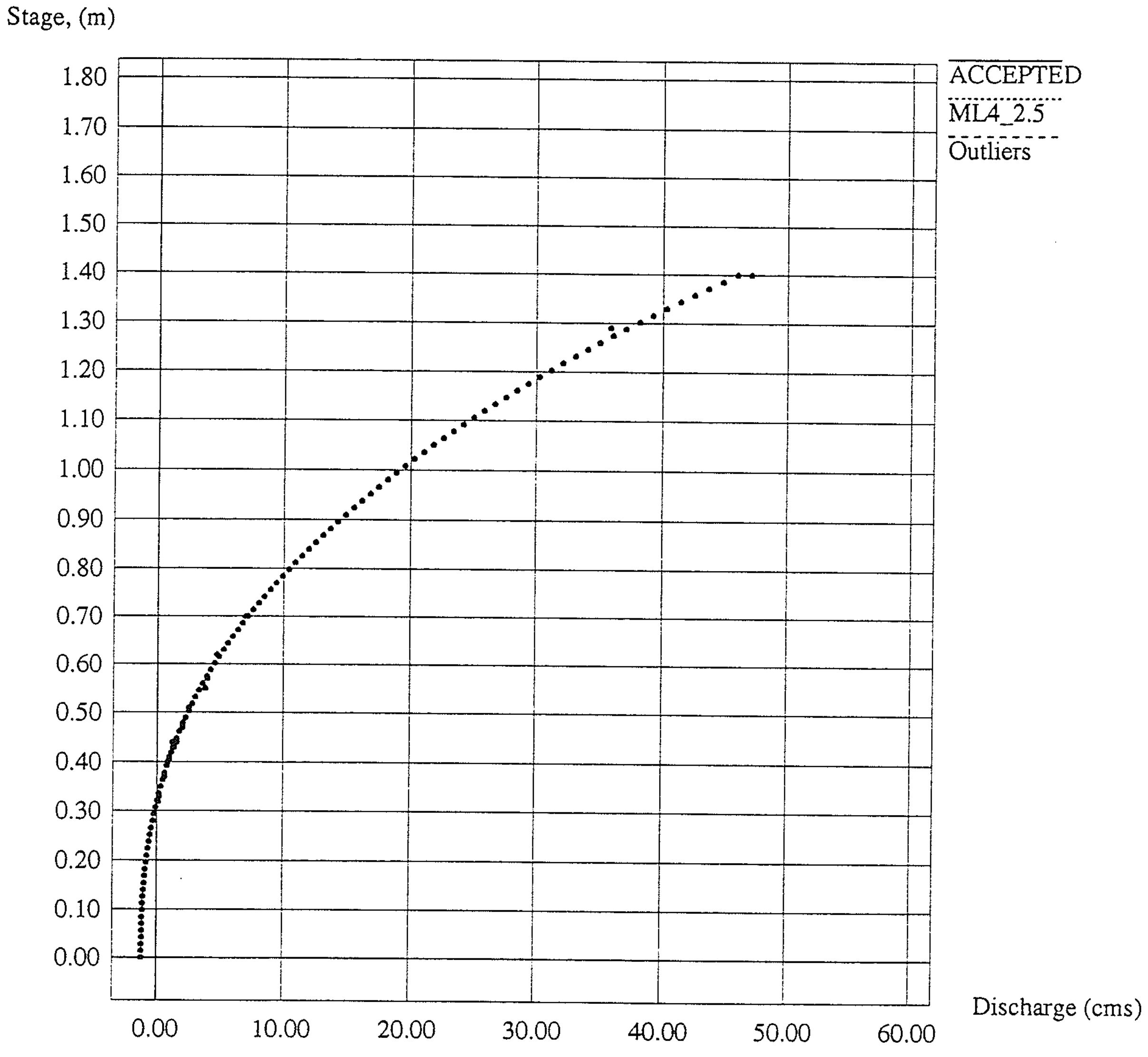


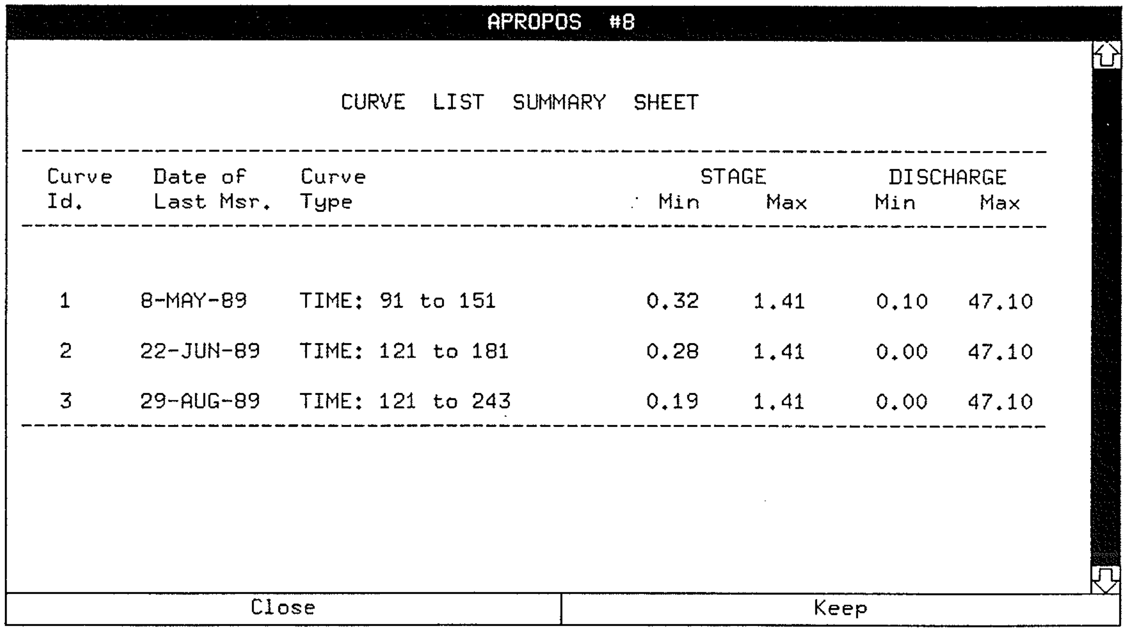
Figure 22. Final stored curve, with outliers omitted from modelling.

```
26-JUN-92 05LJ047
1992
M5
1   1.18   1.21    1.24    1.22
2   1.21   1.24    1.32    1.35    1.42    1.49    1.56
3   1.50   1.55    1.59    1.60    1.62
4   1.6    1.63
5   1.58   1.57    1.59    1.61
6   1.55   1.54    1.52    1.51
7   1.51   1.50
8   1.51   1.53    1.54    1.49
9   1.50   1.44    1.43    1.45    1.46    1.425
10  1.40   1.36    1.42    1.43
11  1.42   1.43    1.42
12  1.44   1.46    1.47    1.50    1.49    1.51    1.53
13  1.49   1.51
14  1.48   1.47    1.44    1.41
15  1.36   1.39    1.42    1.41
16  1.40   1.35    1.30    1.31    1.30    1.31
17  1.31   1.32    1.29    1.275   1.26    1.24
18  1.25   1.24
19  1.24   1.26    1.24
20  1.19   1.21    1.20
21  1.22   1.24    1.26    1.25    1.27
22  1.24   1.28    1.31    1.33    1.36
23  1.36   1.35    1.36
24  1.35   1.32    1.29    1.28    1.25
25  1.26   1.26
26  1.265  1.24    1.25
27  1.25   1.22    1.19    1.15    1.11    1.13    1.135
28  1.14   1.14
29  1.13   1.11
30  1.10   1.09    1.095   1.115   1.110
31  1.09   1.075   1.06
0
```

Figure 23. Sample water level record.

input file shown in Figure 23 contains artificial water level records for the months of May and June.

The second step involves selecting a curve to be used with the water level records. A list of curves is compiled for the current station and displayed to the user. Each curve is displayed with a unique curve id number and its relative attributes. Figure 24 displays the curve list summary for the Edwards Creek station. Model ML2 was developed previously in section 6.2 of this chapter is shown in the list as curve 3. The user is prompted to enter the "id" number of the corresponding curve to be used.

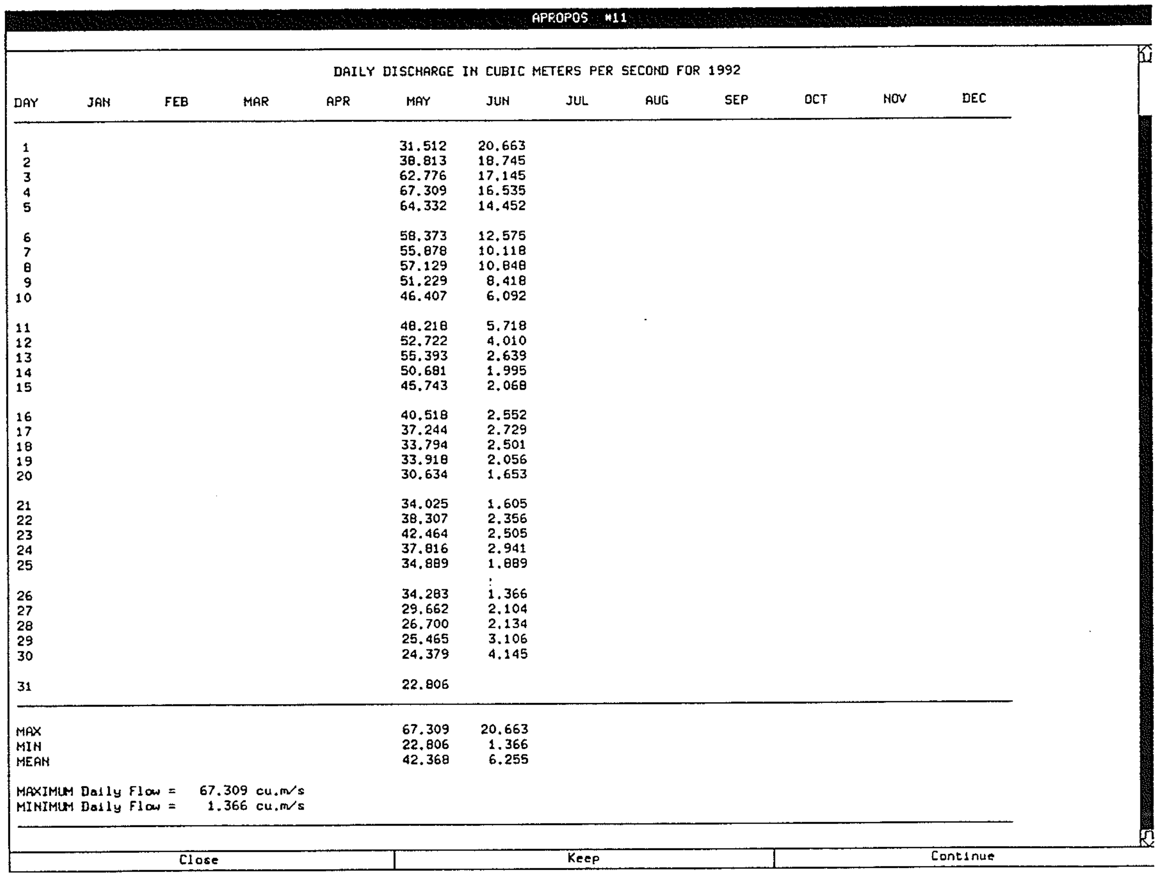


CURVE LIST SUMMARY SHEET

Curve Id.	Date of Last Msr.	Curve Type	STAGE		DISCHARGE	
			Min	Max	Min	Max
1	8-MAY-89	TIME: 91 to 151	0.32	1.41	0.10	47.10
2	22-JUN-89	TIME: 121 to 181	0.28	1.41	0.00	47.10
3	29-AUG-89	TIME: 121 to 243	0.19	1.41	0.00	47.10

Figure 24. List of curves created for Edwards Creek.

The final step requires the user to enter the file name of the water level records. A table of average daily discharges is computed and displayed to the user in a large window filling the entire working surface of the monitor. Once displayed, the user may use the scroll bar of the window to view the entire table. A portion of this table is shown in Figure 25, showing discharges computed from the water level records of May and June shown in Figure 23.



```
                          DAILY DISCHARGE IN CUBIC METERS PER SECOND FOR 1992

DAY   JAN    FEB    MAR    APR    MAY      JUN     JUL    AUG    SEP    OCT    NOV    DEC
------------------------------------------------------------------------------------------

 1                                31.512   20.663
 2                                38.813   18.745
 3                                62.776   17.145
 4                                67.309   16.535
 5                                64.332   14.452

 6                                58.373   12.575
 7                                55.878   10.118
 8                                57.129   10.848
 9                                51.229    8.418
10                                46.407    6.092

11                                48.218    5.718
12                                52.722    4.010
13                                55.393    2.639
14                                50.681    1.995
15                                45.743    2.068

16                                40.518    2.552
17                                37.244    2.729
18                                33.794    2.501
19                                33.918    2.056
20                                30.634    1.653

21                                34.025    1.605
22                                38.307    2.356
23                                42.464    2.505
24                                37.816    2.941
25                                34.889    1.889
                                            :
26                                34.283    1.366
27                                29.662    2.104
28                                26.700    2.134
29                                25.465    3.106
30                                24.379    4.145

31                                22.806
------------------------------------------------------------------------------------------

MAX                               67.309   20.663
MIN                               22.806    1.366
MEAN                              42.368    6.255

MAXIMUM Daily Flow =   67.309 cu.m/s
MINIMUM Daily Flow =    1.366 cu.m/s
------------------------------------------------------------------------------------------
```

Figure 25. Example of computed flow table.

6.4 Examination of Results

The purpose of this section is to make a critical assessment of the ability of the SDDSS to accurately model S-D relationships. The discussion will focus on comparing the relative shape of the curve produced in section 6.2 with one produced by the Water Survey of Canada (WSC) office in Winnipeg.

Figure 26 presents a visual comparison of the two rating curves. A spreadsheet tool was used to plot the curves as corresponding S-D points. Each curve was produced by the same stage values, using one centimetre increments. A rating table prepared by the Water Resources Branch in Winnipeg was used for the WSC curve. The table itself was created from a manually drawn curve, which was computer digitized into a S-D rating table. The SDDSS curve is based on the mathematical model formulated earlier in section 6.2,

$$D = -1.2 + 20.0 * S^{2.6} \qquad [8]$$

where D is the discharge in m^3/s and S is stage in meters.

Upon first glance the curves seem very closely matched. The SDDSS curve tends to over estimate flows, with the exception of the extremely low flow regime. Two reasons for the discrepancy come to mind. Firstly, the SDDSS system is not capable of producing *complex rating curves*. Such relationships are typical where river controls create distinct

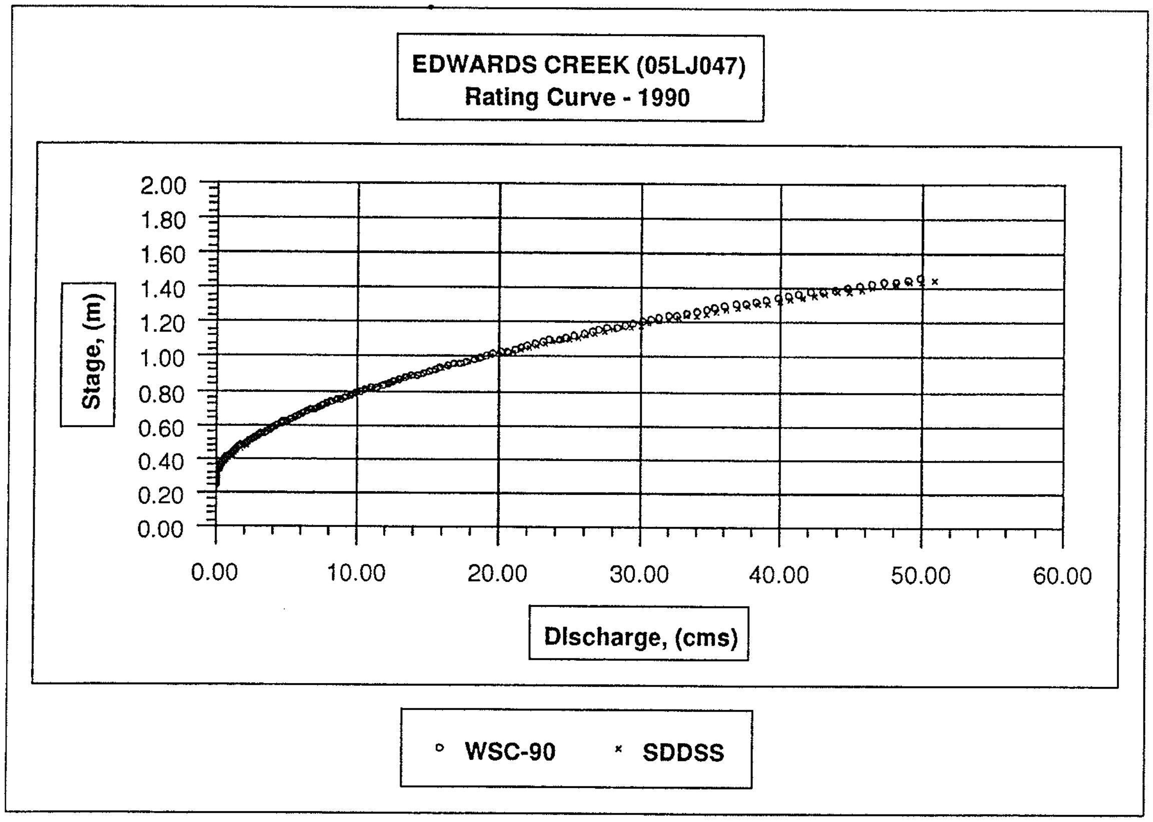


Figure 26. Comparison of SDDSS and hand rendered (WSC-90) curves.

transitions between different flow regimes. They are manifested in rating curves by rapidly changing slopes connected by a distinct bend. The manual method of using french curves to draw S-D relations can easily represent data which exhibits such relations. However, these bends are difficult to reproduce with one mathematical equation in the prototype system. This is because the current SDDSS is only capable of producing smooth parabolic curves. These smooth curves are inappropriate for modelling such abrupt transitions. As a compromise, regions of the curve will tend to over- and under-compensate the relation, in comparison to a complex curve.

Figure 27 shows a close-up comparison of the two curves for the low flow regime. A distinct transition in the slope of the WSC curve is visible at a flow of 0.10 m^3/s. The smooth parabolic curve generated by the SDDSS was not able represent this phenomena. However, if the SDDSS was applied to only measurements observed in this extremely low flow regime, (below 0.32 m of stage), a smooth parabola could be fit to this shape, providing the data itself exhibits such a shape.

The second reason for the discrepancy is that the SDDSS does not discriminate between the date of particular measurements. Regardless of the date of the observation, each measurement is given equal weight. However, it was pointed out in Chapter 3 that all rivers are subject to at least some small degree of shifting. The manual nature of the WSC procedure, allows technicians to observe visual shifts in a relationship and subsequently shift a curve up or down graphically, while retaining its basic shape of the curve. Again, such shifting procedures are generally based on the most recent observations, while those from years past are given significantly less (subjective) weight.

Therefore, from this comparison, it becomes apparent that the system requires improvement in its ability to accurately model S-D relations. The overall SDA process modelled in the DSS should facilitate the representation of complex rating curves, and

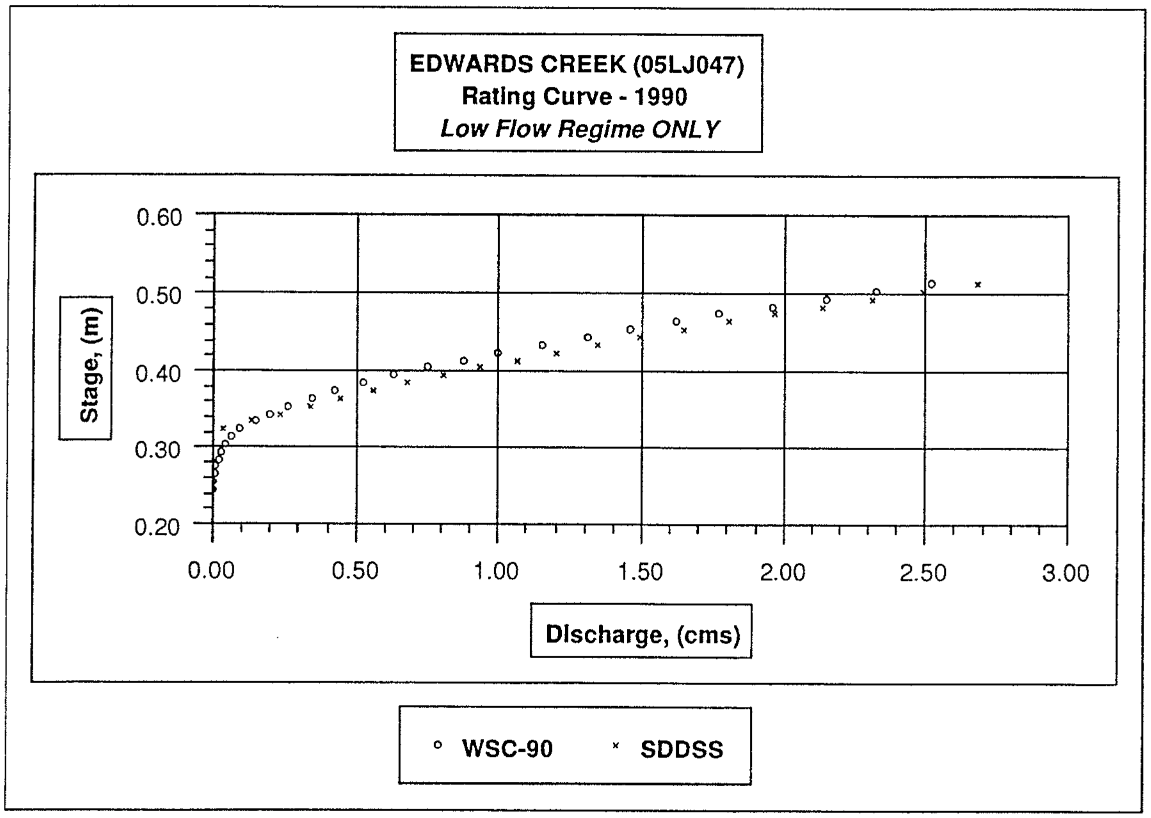


Figure 27. Comparison of curves, low flow regime.

guide the user into discerning the appropriate flow and/or stage for transitions between curves. The computer system HYDSYS, discussed in section 2.3, is capable of modelling rating curves with multiple curve segments, based on the cross-sectional geometry of the channel. Transition points are distinguished and justified by plotting the channel cross sectional shape with the stage-discharge measurements. A similar utility is thus recommended for inclusion into the SDDSS in future development. In addition, the system should be capable of modelling specific years of data, to account for recent shifts in the relationships.

7. CONCLUSIONS AND FUTURE RESEARCH

7.1 Evaluation of SDDSS Prototype

The research work presented in this thesis has introduced the development of a prototype computer-based decision support system (SDDSS) for stage-discharge analysis. In the development of this system, several related topics were studied in detail.

First, the study of stage-discharge analysis and specifically the procedures used within the Department of Environment in Canada exhibited a strong potential for the application of DSS technology. A semi-structured problem was defined and used to conceptualize the system. The prototype system was designed as a framework of procedures, encompassing the activities performed by the Water Resources Branches of the DOE, throughout Canada. It has been intended to explore and experiment with the concepts of SDA process within a computer environment and pave the way for a fully operational system. A much larger intelligent decision support system for the management of surface water quantity data has been proposed, and shall include a fully developed SDDSS, as it is developed over the next several years.

Mathematical modelling is recognized as an effective representation of S-D relationships. The SDDSS is currently a prototype system but the framework design allows for future advancements of the SDDSS to facilitate the modelling of relationships with complex curves and more sophisticated and comprehensive analysis techniques.

Nexpert Object has proven to be a successful tool for prototyping DSSs to deal with problem-solving and reasoning. Nexpert is not necessarily efficient for structured problems which follow a discrete executional path to solve a problem. Conventional programming languages such as C, are better suited for this type of programming. However, in the prototype stage of this research, Nexpert did provide a good environment for modelling the semi-structured nature of SDA. It also allowed for easy and efficient database integration with Oracle tables. Later development of a fully operational SDDSS system is recommended to be programmed in object-oriented C++ code. This will allow for integration among all elements of the DSS including knowledge base structures modelled in Nexpert.

7.2 Future Research Work

The development of the SDDSS has experienced several iterations of planning, designing and programming in order to completely define the entire problem. In light of these numerous iterations, the resulting work has managed to focus on specific details of Stage-Discharge analysis. As mentioned earlier, the current system design presents a logical framework for S-D analysis, lending itself to future additions. Further development of the SDDSS prototype is recommended, in order to further examine the SDA problem domain, before developing a fully operational system. The following sections present some plans for future work and experimentation in the SDDSS prototype.

Automated Modelling

Section 5.2 described the modelling procedure as an iterative process requiring the user to provide input; specifically transformation exponents for modelling. One future advancement in the area of modelling should include an automated process for selecting the optimal exponent for stage transformation. A routine would generate a series of models based a range of transformation exponent values, increment in small steps.

In attempt to be more intelligent, the routine would step through the range in an efficient manner, trying to converge upon the optimal value with the least number of iterations. Decision support would also be incorporated for selecting the best model based on all three evaluation parameters. The system would analyze the data, with respect to all three analysis parameters and recommend the best transformation to the user. Graphical displays of the evaluation parameters would be presented to the user, allowing them to override the system recommendation.

The intent of such a routine is to remove from the user, the burdens of arbitrarily choosing exponents and having to specify one single evaluation criteria. Other transformation functions such as logarithmic and exponential, and methods such as transformation of the dependant variable (discharge) or both, should be included in the modelling process, as well as attention to statistical analysis of regression residuals for evaluating model validity. The system should also contain specific knowledge of statistics and regression analysis which may be used in the selection process or which the user may consult in order to obtain a preferred model.

Curve Segmentation

Presently, the system does not include the ability to handle complex curves. To recap, *complex curves* refer to relationships which are can not be represented with one smooth parabolic curve. Such curves are indicative of S-D relationships which are governed by several station controls for different flow regimes. Complex curves may graphically display one or more inflection points occurring at the point in the relation where two controls are governing the relation simultaneously; ie: a *combined control.*

Representing an entire flow regime of this type with one curve is inappropriate. This is especially true if one control is shifting at a greater rate than an adjacent one, resulting in a more pronounced transition point over time. Variable shifting between two controls is likely to occur since different characteristics define different controls.

Future development of the system should incorporate the ability to deal with the issue of complex curves. This would be achieved by allowing curve data to be segmented by flow or stage ranges and modelled separately. A mechanism would also be included, to provided smooth transitions between adjacent curve segments. Decision support could be provided to assist in the selection of appropriate flow level values, for defining transitions. The plotting utility and database should be further developed to allow channel cross-sectional geometry to be plotted with the stage-discharge measurements to assist in the identification of transition in different flow regimes, as adopted in the HYDSYS computer system (HYDSYS Pty Ltd, 1991).

The above development recommendations raise a critical issue which must be addressed in the near future by organizations such as the Department of Environment.

The ability of the SDDSS to evolve into a sophisticated tool, requires a rigorous and organized program for the collection and documentation of site data to accompany each stage-discharge measurement. A coding system for documenting conditions at the site and surrounding areas, should be defined, standardized and adopted which allows for descriptions by technicians, without allowing variation between different individuals. Introducing such a program is of paramount importance to the development of the SDDSS, as well as the quality and accuracy of rating curves produced.

Decision Support for Unstable Channels

Further development of the system is expected to include special analysis techniques for developing S-D relationships in unstable channels. The basic design framework of the system will be used for curve development. Additional knowledge-bases will address the identification of unstable conditions and incorporate the use of physical and morphological parameters of the channel into the development process.

Intelligent Interface

Due to the rapid pace of the prototyping stage of the SDDSS development, only minor attention was directed at providing a sophisticated user interface. As was mentioned in section 5.3, graphical user interfaces play a key role in the ultimate success of a DSS. For this reason, further development of the SDDSS should experiment with the topic of efficient GUI. Improvements on the current prototype could include more elaborate messages and instructions for the user, better graphical presentation of relationships and data, using continuous lines (rather than discrete points) for curves, and a more extensive user input system, utilizing menus, selection items and buttons. An on-line help and instruction manual is also recommended for consideration in future work.

References

Bedient, P.B. and Huber, W.C., (1988). "Hydrology and Floodplain Analysis", Addison-Wesley Publishing Company, Reading, Massachusetts, USA.

Câmara, A.S., Cardoso da Silva, M., Rodrigues, A.C. Remédio, J.M., Castro, P.P., Soares de Oliveira, M.J. and Fernandes, T.F., (1990). "Decision Support System for Estuarine Water-Quality Management", *Journal of Water Resources Planning and Management*, Vol.116, No.3, pp.417-432.

Chester, B.L., (1986). "Stage Discharge Relationships - Overview and Theory", Water Authority of Western Australia, Australia.

Davis, J.R., Nanninga, P.M., Biggins, J. and Laut, P., (1991). "Prototype Decision Support System for Analyzing Impact of Catchment Policies", *Journal of Water Resources Planning and Management*, ASCE, New York, USA, pp.399-414.

Douglas, G.G., (1990a). "An Expert System for the Selection of a Flow Measurement Method", *Bachelor's Graduate Thesis*, University of Manitoba, Department of Civil Engineering, Winnipeg, Manitoba, Canada.

Douglas, G.G., (1990b). "Development of an Advisor System for the Selection of a Flow Measurement Method", *Water Resources Research Report*, No.16, University of Manitoba, Department of Civil Engineering, Winnipeg, Manitoba, Canada.

Douglas, G.G. and Simonovic, S.P., (1992). "The Design of a Computer-System for Developing Stage-Discharge Rating Curves in Stable Channels", from *Proceeding of the Canadian Hydrology Symposium, No. 19*, Winnipeg, Canada.

Environment Canada, (1992). HYDEX data base. Environment Canada, Ottawa.

Fenves, S.J., (1986). "What is an Expert System", from *Expert Systems in Civil Engineering*, ed.by C.N. Kostem and M.L. Maher, ASCE, New York, USA, pp.1-6.

Gaschnig, J., Reboh, R. and Reiter, J., (1981). "Development of a Knowledge-Based System for Water Resources Problems", *Technical Report SRI Project 1619*, SRI International.

Harrison, D., (1989). "XGraph", Version 11.3.2. University of California, Berkeley, USA.

Hydsys Pty Limited, (1991). "Reference Manual" for *HYDSYS/TS Time Series Data Management*, HYDSYS Pty Ltd, Weston Creek, Australia.

ISO, (1981). "Liquid flow measurement in open channels - Part 1: Establishment and operation of a gauging station", ISO Standard 1100/1-1981 from *Measurement of Liquid Flow in Open Channels - ISO Standards Handbook 16*, International Organization of Standards, Genève, Switzerland, pp.133-153.

ISO, (1982). "Liquid flow measurement in open channels - Part 2: Determination of the stage-discharge relationship", ISO Standard 1100/2-1982 from *Measurement of Liquid Flow in Open Channels - ISO Standards Handbook 16*, "Measurement of Liquid Flow in Open Channels", *ISO Standards Handbook 16*, International Organization of Standards, Genève, Switzerland, pp.154-186.

ISO, (1983). "Measurement of Liquid Flow in Open Channels", *ISO Standards Handbook 16*, International Organization of Standards, Genève, Switzerland.

Kao, J. and Liebman, J.C., (1991). "Computer-Aided System for Ground-Water Resources Management", *Journal of Computing in Civil Engineering*, Vol.5, No.3, pp.251-266.

Lavender, S.T., (1984). "Winter Rating Curves and Ice Volume Limited Water Levels", from *A Workshop on the Hydraulics of River Ice*, Fredericton, Canada, pp.279-295.

Maher, M.L., (1986). "Problem Solving using Expert System Technology", from *Expert Systems in Civil Engineering*, Ed.by C.N. Kostem and M.L. Maher, ASCE, New York, USA, pp.7-17.

Maher, M.L. and Allen, R., (1987). "Expert Systems Components", from *Expert Systems for Civil Engineers - Technology and Application*, Ed.by M.L. Maher, American Society of Civil Engineers, New York, USA, pp.3-14.

Neter, J., Wasserman, W. and Kutner, M.H., (1989). "Applied Linear Regression Models", 2nd Ed., Richard D. Irwin Inc., Homewood, Illinois, USA.

Neuron Data Inc., (1991). "Introduction Manual" for *Nexpert Object - Version 2.0*, Palo Alto, California, USA.

Rantz, S.E. and others. (1982a). "Measurement and Computation of Streamflow: Volume I. Measurement of Stage and Discharge": *United States Geological Survey Water-Supply Paper 2175*, Washington, D.C., USA.

Rantz, S.E. and others. (1982b). "Measurement and Computation of Streamflow: Volume II. Computation of Discharge": *United States Geological Survey Water-Supply Paper 2175*, Washington, D.C., USA.

Robertson, J.A. and Crowe, C.T., (1985). "Engineering Fluid Mechanics", 3rd Ed., Houghton-Mifflin Company, Boston, USA.

Rodgers, M.W. and Thompson, S.M., (1991). "Tideda Reference Manual", Publication No. 24, Hydrology Centre, DSIR Marine and Freshwater, Christchurch, New Zealand.

Savic, D.A., (1989). "REZES: The Intelligent Decision Support System for Reservoir Analysis", *Water Resources Research Report*, No.13, University of Manitoba, Department of Civil Engineering, Winnipeg, Manitoba, Canada.

Simonovic, S.P. and Savic, D.A., (1989). "Intelligent Decision Support and Reservoir Management and Operations", *Journal of Computing in Civil Engineering*, Vol.3, No.4, pp.367-385.

Simonovic, S.P., (1989). "Expert System Design of an Intelligent Decision Support for Surface Water Quantity Data Management", from *Computational Modelling and Experimental Methods in Hydraulics (HYDROCOMP '89)*, Ed.by Č. Maksimonvić, and M. Radojković, Elsevier Science Publishers Ltd., New York, USA., pp.383-393.

Simonovic, S.P., (1990). "An Expert System for the Selection of a Suitable Method for Flow Measurement in Open Channels", *Journal of Hydrology*, Vol.112, pp.237-256.

Simonovic, S.P., (1991). "Knowledge-Based Systems and Operational Hydrology", from *Canadian Journal of Civil Engineering*, Vol.18, No.1, pp.1-11.

Simonovic, S.P., (1992). "Lecture Notes" for course 23.729, *Intelligent Decision Support in Water Resources*, University of Manitoba, Winnipeg, Manitoba, Canada.

Smith, A.G., (1987). "Comparison of Methods of Extending Rating Curves", *Internal Report*, Planning and Studies Section of Water Resources Branch, Environment Canada, Vancouver, British Columbia, Canada.

Sommerville, I., (1985). "Software Engineering", 2nd Edition., Addison-Wesley Publishers Company Inc., Wokingham, England.

Soncini-Sess, R., Nardini, A., Gandolfi, C. and Kraszewski, A., (1991). "Computer-Aided Water Reservoir Management: A Prototype Two-Level DSS", from *Decision Support Systems - Water Resources Planning*, Ed.by D.P. Loucks, and J.R. da Costa, NATO ASI Series, Vol. G 26, Springer-Verlag, Berlin, Germany, pp.527-574.

Terzi, R.A., (1981). "Hydrometric Field Manual - Measurement of Streamflow", Environment Canada, Inland Waters Directorate, Ottawa, Canada.

Thierauf, R.J., (1988). "User-Oriented Decision Support Systems - Accent on Problem Finding", Prentice Hall Inc, Englewood Cliffs, New Jersey, USA.

APPENDIX A - Unix Scripts

The following list contains the names of the Unix program scripts referenced in Chapter 5 of this thesis. LRM is used to format an input string for the fortran program LR2, from a string of command line parameters. Output from LR2 is also re-formatted by LRM into Nexpert data format. GC is used in a similar manner with the fortran program GCRV to format input and output. XGR is used to call the XGraph plotting utility with preset options. Programming code for each script begins on the page number indicated below.

	Page
LRM (Linear Regression Model)	135
GC (Generate Curve)	136
XGR (Xgraph Run)	137

```
#*****************************************************************
#*  Unix Script:  LRM $1 $2 $3 $4                                *
#*****************************************************************
#*                                                               *
#*  Parameters:        $1 = input file containing s/d data       *
#*                          format:  D S  (one line per "D S" pair)  *
#*                            ...                                *
#*                     $2 = exponent for Stage data transformation (Float)  *
#*                     $3 = output file listing (fields: b0, b1, corr,  *
#*                          ss_error) in NXPDB format, (eg: MD_1.NXP)  *
#*                     $4 = outfile listing model report (eg: MD_1.txt)  *
#*                                                               *
#*     Note:  LRM creates a generic LRMOD.in input file from $1 and  *
#*            $2, calls LRMOD fortran program and then copies the  *
#*            output file LRMOD.out1 & ...out2 to $3 and $4 file-  *
#*            names respectively.                                *
#*                                                               *
#*     Example use:                                              *
#*               > LRM SD1.in 2.5 SD1.NXP SD1.txt                *
#*                                                               *
#*****************************************************************

# Construct LRMOD input file;                                    *
  echo $2 > LRMOD.in
  cat $1 >> LRMOD.in

# Model data in LRMOD.in;                                        *
  LR2

# *** Move LRMOD.out data into output file, $3;                  *
  cp  LRMOD.out1 $3

# *** Move LRMOD.out2 data into output filec $4;                 *
  cp  LRMOD.out2 $4

# *** Remove LRMOD.in file                                       *
  rm -f LRMOD.in

#*****************************************************************
#*     End of LRM script.                                        *
#*****************************************************************
```

```
#****************************************************************
#*  Unix Script:  GC $1 $2 $3 $4 $5                             *
#****************************************************************
#*                                                              *
#*  Parameters:      $1 = B0 model parameter                    *
#*                   $2 = B1 model parameter                    *
#*                   $3 = Exponent used in Y (stage) transformation *
#*                   $4 = Upperbound of Y (stage) defining curve *
#*                   $5 = Output file for generated curve.      *
#*                                                              *
#****************************************************************

#*** Create GENCRV.in input file ...                            *
 echo $1 > GCRV.in
 echo $2 >> GCRV.in
 echo $3 >> GCRV.in
 echo $4 >> GCRV.in

#*** Run GCRV program to generate points                        *
 GCRV

#*** Transfer generate curve points (GCRV.out 1&2) to           *
#*** output files with file extensions                          *
 cat GCRV.out1 >> $5.mdl
 cat GCRV.out2 >> $5.cl

#*** Clean up files                                             *
 rm -f GCRV.in
 rm -f GCRV.out1
 rm -f GCRV.out2

#****************************************************************
#*    End of GC script.                                         *
#****************************************************************
```

```
#******************************************************************
#*  Unix Script:  XGR $1 $2 $3 $4                                 *
#******************************************************************
#*                                                                *
#*  This script calls xgraph program to plot a set (or sets) of data.  *
#*                                                                *
#*  Parameters:       $1 = station name for xgraph main title     *
#*                    $2 = filename of data set(s) to be plotted in xgraph  *
#*                    $3 = upper limit of Y axis                  *
#*                    $4 = upper limit of X axis                  *
#*                                                                *
#*  Note:      The $2 filename is an xgraph data file of one (or more)  *
#*             data sets, each with a legend title preceeding it.  *
#*             Data sets are separated by a blank line.           *
#*                                                                *
#*  Example:          "legend1                                    *
#*                    X Y                                         *
#*                    ...                                         *
#*                                                                *
#*                    "legend2                                    *
#*                    X Y                                         *
#*                    ...                                         *
#*                                                                *
#******************************************************************

# xgraph -nl -P -M -ly 0,$3 -lx -0.5,$4 -m -t "Station - $1"
# -x "Discharge (cms)" -y "Stage, (m)" =600x585+530+280 $2

xgraph -nl -P -bb -M -ly 0,$3 -lx -0.5,$4 -m -t "Station - $1"
 -x "Discharge (cms)" -y "Stage, (m)" =605x575+535+285
 -display $DISPLAY $2

#******************************************************************
#*      End of XGR script.                                        *
#******************************************************************
```

APPENDIX B - Fortran Programs

The following list contains the names of the Fortran programs referenced in Chapter 5 of this thesis. LR2 is used to perform a linear regression analysis. GCRV is used to generate a set of data points representing a curve, for plotting in the XGraph utility. Programming code begins on the page number indicated below.

	Page
LR2 (Linear Regression 2)	139
GCRV (Generate Curve)	143

```
c       ************************************************************
c       * Fortran Program:  LR2                                    *
c       ************************************************************
c       *                                                          *
c       * Reads X,Y data from a file "LR.in", transforms the       *
c       * X data (X'=X^pow) and creates a linear regression        *
c       * models of the X',Y data.  Several output files           *
c       * store the results of the modelling.                      *
c       * Formats of the program parametes are as follows:         *
c       *                                                          *
c       * LR.in:      tpow   (X-axis transformation exponent)      *
c       *             err    (error allowance for rejects)         *
c       *             Y X    (X-Y coordinate point)                *
c       *             Y X                                          *
c       *             ...                                          *
c       *                                                          *
c       ************************************************************


c       **** Main Program: LR2                                     *

        REAL*4 tpow, alwerr, p_err
        DIMENSION ydat(200), xdat(200)


c       ************************************************************
c       * Read Transformation power and error allowance  (LR.in)   *
c       ************************************************************

        OPEN(3,FILE='LR.in')
        READ(3,'(F4.1)') tpow
        READ(3,'(F4.1)') alwerr

c       ************************************************************
c       *  Calculate Regression parameters                         *
c       ************************************************************
        i = 1
        xsum = 0
        ysum = 0
        xysum = 0
        ymax = 0
        xmax = 0
```

```
      DO WHILE ( i .LT. 200 )
        READ(3,*, END=10) ydat(i), xdat(i)
        txdat = xdat(i)**tpow
        ysum = ysum + ydat(i)
        xsum = xsum + txdat
        xysum = xysum + txdat * ydat(i)
        x2sum = x2sum + txdat**2.0
        y2sum = y2sum + ydat(i)**2.0
        IF ( ydat(i) .GT. ymax ) ymax = ydat(i)
        IF ( xdat(i) .GT. xmax ) xmax = xdat(i)
        i = i + 1
      END DO
 10   nobs = i-1
      xmean = xsum / nobs
      ymean = ysum / nobs
      b1 = ((xysum)-(xsum*ysum)/ nobs) / (x2sum - xsum**2/nobs)
      b0 = ymean - b1*xmean
      CLOSE(3)

c     ************************************************************
c     *  Print Regression parameters                              *
c     ************************************************************
      WRITE(*,160) b0, b1
 160  FORMAT('B0= ', F6.3, 5X, 'B1 = ', F6.3)


c     ************************************************************
c     * Calculate SSE and R2 values and perform reject check.     *
c     ************************************************************

      OPEN(4,FILE='LR.ol1')
      sse = 0.0
      ssr = 0.0
      numol = 0
      DO 100 j = 1, nobs, 1

c     ***  Calculate response of "ymod".                          *
        txdat = xdat(j)**tpow
        ymod = b0 + b1 * txdat

c     ***  Calc. Sqrd. dev. of model from reg.line.               *
        edev = ymod-ydat(j)
        sse = sse + edev*edev
```

```
c        ***  Calc. Sqrd. Dev. of reg.line. from mean                   *
           ssr = ssr + (ymod-ymean) * (ymod-ymean)

c        ***  Flag current pair as rejected if  % error of "edev"       *
c        ***  is larger than Error Allowance "alwerr"                   *
           p_err = ABS(edev / ymod) * 100
           IF ( p_err .GT. alwerr ) THEN
             WRITE(4,110) ydat(j), xdat(j)
             numol = numol + 1
           ENDIF

  100    CONTINUE
c        ***  Calculate R2 and STDEE values                             *
         stdee = (sse/(nobs-2))**0.5
         r2 = ssr / (sse+ssr)

  110    FORMAT( F8.3, ' ', F8.3)
         CLOSE(4)


c        *************************************************************
c        * Write b0, b1, corr, sse, numol "LR.out1".                  *
c        *************************************************************
         OPEN(4,FILE='LR.out1')
         WRITE(4,210)
  210    FORMAT('      b0|        b1|       corr|  stderest|',
     +   ' numol|')
         WRITE(4,220)
  220    FORMAT( 51('*') )
         WRITE(4,240) b0,b1,r2,stdee,numol
  240    FORMAT( 3(F10.4,'|'), F10.3, '|', I6,'|')
         WRITE(4,220)
         CLOSE(4)

c        *************************************************************
c        * Write Results for display text file (LR.out2).             *
c        *************************************************************
         OPEN(5,FILE='LR.out2')
         WRITE(5,*)
         WRITE(5,*)
         WRITE(5,310) tpow
  310    FORMAT('   Model Name:  LRM_', F4.2)
         WRITE(5,320) b1, tpow, b0
```

```
320   FORMAT('          Eqn:  D = (',F7.2, ') * S ^ (' ,
    +  F3.1 , ') + (', F7.2 , ')' )
      WRITE(5,330) r2
330   FORMAT('           R2:  ', F6.4 )
      WRITE(5,340) sse
340   FORMAT('          SEE:  ', F8.3 )
      WRITE(5,*)
      WRITE(5,350) xmax
350   FORMAT('      Max Stage = ', F8.3)
      WRITE(5,355) ymax
355   FORMAT('  Max Discharge = ', F8.3)
      WRITE(5,*)
      WRITE(5,360) numol, nobs, alwerr
360   FORMAT('  ', I3, ' of ', I3, ' observations were NOT ' ,
    +  'within allowable limits of +/- ', F4.1, ' %.')
      WRITE(5,*)
      CLOSE(5)

      STOP
      END

c     ***************************************************
c     *  End of program LR2.                             *
c     ***************************************************
```

```
c       ************************************************************
c       * Fortran Program:  GCRV                                   *
c       ************************************************************
c       *                                                          *
c       *  Accepts parametes b0, b1, and pow and generates a       *
c       *  set of 100 X,Y points.  The generated 100 points are    *
c       *  stored in an ouput file.  The X data is generated       *
c       *  from Y values from 0 to 1.00 according to the fol-      *
c       *  lowing formula:                                         *
c       *             X = b0 + b1*( Y**pow )                       *
c       *                                                          *
c       *  The input and output data are stored in GCRV.in         *
c       *  GCRV.out  respectively, as follows:                     *
c       *                                                          *
c       *  GCRV.in:  b0     (x-intercept)                          *
c       *            b1     (inv slope = x/y)                      *
c       *            pow    (transformation exponent)              *
c       *            ymax   (max y axis used in generation)        *
c       *                                                          *
c       *  GCRV.out:        X Y                                    *
c       *            ... 100 lines                                 *
c       *                                                          *
c       ************************************************************


c       ***  Main Program: GCRV                                    *

c       ***  Read input from "GCRV.in"                             *
        OPEN(2,FILE='GCRV.in')
        READ(2, '(F8.5)') b0
        READ(2, '(F8.5)') b1
        READ(2, '(F4.1)') pow
        READ(2, '(F7.2)') ymax
        CLOSE(2)

c       ***  Generate X points from Y points and write to file.    *
        OPEN(3,FILE='GCRV.out1')
        yincr = ymax/100
        ypoint = 0
        DO 100 i = 0, 100
          xpoint = b0 + b1*(ypoint)**(pow)
            WRITE(3,'(F8.3, X, F8.3)') xpoint, ypoint
          ypoint = ypoint + yincr
```

```
 100    CONTINUE
        CLOSE(3)

c       ***  Generate allowance limits, +/- 2.5% of model response            *

        OPEN(4,FILE='GCRV.out2')
        ypoint = 0
        DO 150 i = 0, 100
          xpt_pos = 1.025 * ( b0 + b1*(ypoint)**(pow))
          xpt_neg = 0.975 * ( b0 + b1*(ypoint)**(pow))
            WRITE(4,'(F8.3, X, F8.3)') xpt_neg, ypoint
            WRITE(4,'(F8.3, X, F8.3)') xpt_pos, ypoint
          ypoint = ypoint + yincr
 150    CONTINUE
        CLOSE(4)

        STOP
        END

c       ************************************************************
c       *  End of program GCRV.                                    *
c       ************************************************************
```

APPENDIX C - Nexpert Object Knowledge Bases

The following list contains the names of the Nexpert Object knowledge bases referenced in Chapters 5 and 6 of this thesis. Programming code begins on the page number indicated below.

		Page
A0_main.tkb	(Definitions)	146
A1_main.tkb	(System Operations)	160
A2_main.tkb	(Main Menu & Curve Use Modules)	165
A3_main.tkb	(Curve Development Module)	179
A4_main.tkb	(Outlier Analysis Module)	232
A5_main.tkb	(Curve Modification Module)	255

```
#****************************************************************
#*  Nexpert Object KB Code:  A0_main.tkb                         *
#****************************************************************
#*  DEFINITIONS - Master System                                  *
#****************************************************************


(@VERSION=   020)
(@PROPERTY=        Atr_Tbl @TYPE=String;)
(@PROPERTY=        B0              @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        B1              @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Crv_Id  @TYPE=Integer;)
(@PROPERTY=        Crv_Tbl         @TYPE=String;)
(@PROPERTY=        Crv_Type        @TYPE=String;)
(@PROPERTY=        Current  @TYPE=Boolean;)
(@PROPERTY=        Cursor   @TYPE=Integer;)
(@PROPERTY=        Dat_Tbl         @TYPE=String;)
(@PROPERTY=        Datemade        @TYPE=Date;@FORMAT="d\"-\"MMM\"-\"yy";)
(@PROPERTY=        Description     @TYPE=String;)
(@PROPERTY=        Discharge       @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        F_Day   @TYPE=Integer;)
(@PROPERTY=        Filename        @TYPE=String;)
(@PROPERTY=        FN_Conlim       @TYPE=String;)
(@PROPERTY=        FN_Curve        @TYPE=String;)
(@PROPERTY=        FN_Data         @TYPE=String;)
(@PROPERTY=        FN_Outliers     @TYPE=String;)
(@PROPERTY=        FN_Results      @TYPE=String;)
(@PROPERTY=        Index           @TYPE=Integer;)
(@PROPERTY=        L_Day   @TYPE=Integer;)
(@PROPERTY=        Last_Msr        @TYPE=Date;@FORMAT="d\"-\"MMM\"-\"yy";)
(@PROPERTY=        Max_D  @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Max_S  @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Min_D  @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Min_S  @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Msr_Date        @TYPE=Date;@FORMAT="d\"-\"MMM\"-\"yy";)
(@PROPERTY=        Msr_Id  @TYPE=Integer;)
(@PROPERTY=        Msrs_to_date    @TYPE=Integer;)
(@PROPERTY=        No_Flow_Stg     @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Num_Msrs        @TYPE=Integer;)
(@PROPERTY=        Num_obj         @TYPE=Integer;)
(@PROPERTY=        Num_OL          @TYPE=Integer;)
(@PROPERTY=        Obj_names       @TYPE=String;)
(@PROPERTY=        Plot            @TYPE=Boolean;)
(@PROPERTY=        Query           @TYPE=String;)
(@PROPERTY=        R2              @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Set             @TYPE=String;)
(@PROPERTY=        Stage           @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        StdEE           @TYPE=Float;@FORMAT="u.0000";)
(@PROPERTY=        Stn_Id  @TYPE=String;)
```

```
(@PROPERTY=         Stn_Name        @TYPE=String;)
(@PROPERTY=         Title           @TYPE=String;)
(@PROPERTY=         TX              @TYPE=Float;@FORMAT="u.00";)
(@PROPERTY=         Type            @TYPE=String;)
(@PROPERTY=         ZF_Count        @TYPE=Integer;)


(@CLASS=        Best_models
        (@PROPERTIES=
                B0
                B1
                F_Day
                FN_Conlim
                FN_Curve
                FN_Data
                FN_Outliers
                FN_Results
                L_Day
                Num_OL
                Plot
                R2
                StdEE
                Title
                TX
        )
)

(@CLASS=        Curves
        (@PROPERTIES=
                B0
                B1
                Crv_Type
                Datemade
                F_Day
                FN_Curve
                L_Day
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                Value   @TYPE=Integer;
        )
)

(@CLASS=        Datasets
        (@PROPERTIES=
                Current
                Description
                FN_Data
                Last_Msr
```

```
                        Max_D
                        Max_S
                        Min_D
                        Min_S
                        No_Flow_Stg
                        Num_Msrs
                        Plot
                        Query
                        Title
                        Type
                        ZF_Count
                )
        )

        (@CLASS=        Models
                (@PROPERTIES=
                        B0
                        B1
                        Crv_Id
                        Crv_Type
                        F_Day
                        FN_Conlim
                        FN_Curve
                        FN_Data
                        FN_Outliers
                        FN_Results
                        L_Day
                        Num_OL
                        Plot
                        R2
                        StdEE
                        Title
                        TX
                )
        )


        (@OBJECT=       A_Begin
                (@PROPERTIES=
                        Value   @TYPE=Boolean;
                )
        )

        (@OBJECT=       A_Begin_Sys0
                (@PROPERTIES=
                        Value   @TYPE=Boolean;
                )
        )

        (@OBJECT=       Allow_err
          (@PROPERTIES=
                    Value    @TYPE=Float;
          )
)

(@OBJECT=     answer
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     BEST_Model
          (@PROPERTIES=
                    B0
                    B1
                    Crv_Id
                    Crv_Type
                    F_Day
                    FN_Conlim
                    FN_Curve
                    FN_Data
                    FN_Outliers
                    FN_Results
                    L_Day
                    Num_OL
                    Plot
                    R2
                    StdEE
                    Title
                    TX
          )
)

(@OBJECT=     Bool_Answer
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     C_Crv
          (@CLASSES=
                    Curves
          )
          (@PROPERTIES=
                    B0
                    B1
                    Crv_Id
                    Crv_Type
                    Datemade
                    F_Day
                    FN_Curve
```

```
                L_Day
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                TX
                Value    @TYPE=Integer;
        )
)

(@OBJECT=       C_Stn
        (@PROPERTIES=
                Atr_Tbl
                Crv_Tbl
                Dat_Tbl
                Msrs_to_date
                Stn_Id
                Stn_Name
                Value    @TYPE=String;
        )
)

(@OBJECT=       Cnst_time1
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=       Cnst_time2
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=       Control_Next
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=       Control_Return
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       Count
        (@PROPERTIES=
                Value    @TYPE=Integer;
        )
```

```
)

(@OBJECT=     Counter
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     Crv_Entry
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     CrvDev_Call
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     CrvDev_Control
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Call
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     CrvMod_Control
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Cursor
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     CurveDevelopment_Open
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CurveDevelopment_Start
		(@PROPERTIES=
			Value	@TYPE=Boolean;
		)
)

(@OBJECT=	CurveModify_Open
		(@PROPERTIES=
			Value	@TYPE=Boolean;
		)
)

(@OBJECT=	CurveModify_Return
		(@PROPERTIES=
			Value	@TYPE=Boolean;
		)
)

(@OBJECT=	CurveModify_Start
		(@PROPERTIES=
			Value	@TYPE=Boolean;
		)
)

(@OBJECT=	db_access
		(@PROPERTIES=
			Value	@TYPE=String;
		)
)

(@OBJECT=	dummy_string
		(@PROPERTIES=
			Value	@TYPE=String;
		)
)

(@OBJECT=	dummy_params
		(@PROPERTIES=
			Value	@TYPE=String;
		)
)

(@OBJECT=	Lock_CrvDev
		(@PROPERTIES=
			Value	@TYPE=String;
		)
)

(@OBJECT=	Lock_CrvMod
		(@PROPERTIES=
			Value	@TYPE=String;
		)
```

```
)

(@OBJECT=     Lock_MainMenu
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Lock_OutAnal
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     MainMenu_Control
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     MainMenu_Open
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     MainMenu_Start
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     New_msr
        (@PROPERTIES=
                Discharge
                Msr_Date
                Msr_Id
                Set
                Stage
        )
)

(@OBJECT=     OutAnal_Call
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     OutAnal_Control
        (@PROPERTIES=
                Value    @TYPE=Boolean;
```

```
          )
)

(@OBJECT=     OutlierAnalysis_Open
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     OutlierAnalysis_Return
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     OutlierAnalysis_Start
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Phase
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)


(@OBJECT=     Phs1
          (@CLASSES=
                    Datasets
          )
          (@PROPERTIES=
                    Current
                    Description
                    FN_Data
                    Last_Msr
                    Max_D
                    Max_S
                    Min_D
                    Min_S
                    Num_Msrs
                    Plot
                    Query
                    Title
                    Type
          )
)

(@OBJECT=     Phs2
          (@CLASSES=
                Datasets
        )
        (@PROPERTIES=
                Current
                Description
                FN_Data
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                Num_Msrs
                Plot
                Query
                Title
                Type
        )
)

(@OBJECT=       Phs3
        (@CLASSES=
                Datasets
        )
        (@PROPERTIES=
                Current
                Description
                FN_Data
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                Num_Msrs
                Plot
                Query
                Title
                Type
        )
)


(@OBJECT=       Phs5
        (@CLASSES=
                Datasets
        )
        (@PROPERTIES=
                Current
                Description
                FN_Data
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                Num_Msrs
                Plot
                Query
                Title
                Type
        )
)



(@OBJECT=       Plot_Info
        (@PROPERTIES=
                Filename
                Index
                Max_D
                Max_S
                Num_obj
                Obj_names
                Title
        )
)

(@OBJECT=       Qry_Str_Prefix
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=       Query_String
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=       Start_day
        (@PROPERTIES=
                Value   @TYPE=Integer;
        )
)

(@OBJECT=       Stop_day
        (@PROPERTIES=
                Value   @TYPE=Integer;
        )
)

(@OBJECT=       System_Call
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=       System_Control
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       System_Return
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       Temp_Crv
        (@CLASSES=
                Curves
        )
        (@PROPERTIES=
                B0
                B1
                Crv_Type
                Datemade
                F_Day
                FN_Curve
                L_Day
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                Value    @TYPE=Integer;
        )
)

(@SLOT=         Datasets.Max_D
        (@INITVAL=    0)
)

(@SLOT=         Datasets.Max_S
        (@INITVAL=    0)
)

(@SLOT=         Datasets.Min_D
        (@INITVAL=    1000000)
)

(@SLOT=         Datasets.Min_S
        (@INITVAL=    1000000)
```

```
)

(@SLOT=         Datasets.Num_Msrs
        (@INITVAL=    0)
)

(@SLOT=         Datasets.Plot
        (@INITVAL=    TRUE)
)

(@SLOT=         Allow_err
        (@INITVAL=    30.0)
)

(@SLOT=         Cnst_time1
        @COMMENTS="See Get_Phs2_A hypothesis";
        (@INITVAL=    " (TO_NUMBER(TO_CHAR(date_of_msr,\
'DDD')) >= @V(Start_day) AND TO_NUMBER(TO_CHAR(date_of_msr,\
'DDD')) <= @V(Stop_day) ) ")
)

(@SLOT=         Cnst_time2
        @COMMENTS="See Get_Phs2_A hypothesis";
        (@INITVAL=    " (TO_NUMBER(TO_CHAR(date_of_msr,\
'DDD')) >= @V(Start_day) OR TO_NUMBER(TO_CHAR(date_of_msr,\
'DDD')) <= @V(Stop_day) ) ")
)

(@SLOT=         db_access
        @INFCAT=20;
        (@INITVAL=    "glen/mac")
)

(@SLOT=         Start_day
        (@INITVAL=    0)
)

(@SLOT=         Stop_day
        (@INITVAL=    0)
)

(@GLOBALS=
        @INHVALUP=FALSE;
        @INHVALDOWN=TRUE;
        @INHOBJUP=FALSE;
        @INHOBJDOWN=FALSE;
        @INHCLASSUP=FALSE;
        @INHCLASSDOWN=TRUE;
        @INHBREADTH=TRUE;
        @INHPARENT=FALSE;
        @PWTRUE=TRUE;
        @PWFALSE=TRUE;
        @PWNOTKNOWN=TRUE;
        @EXHBWRD=TRUE;
        @PTGATES=TRUE;
        @PFACTIONS=TRUE;
        @SOURCESON=TRUE;
        @CACTIONSON=TRUE;
        @SUGLIST=A_Begin;
)

#*****************************************************************
#*  End of KB Code:  A0_main.tkb                                 *
#*****************************************************************
```

```
#****************************************************************
#*  Nexpert Object KB Code:  A1_main.tkb                        *
#****************************************************************
#*  SYSTEM OPERATION - Master System                            *
#****************************************************************


(@VERSION=   020)

(@OBJECT=     System_Resets
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@SLOT=        System_Resets
        (@CACTIONS=
                (Reset   (CurveDevelopment_Open))
                (Reset   (CurveDevelopment_Start))
                (Reset   (CurveModify_Open))
                (Reset   (CurveModify_Start))
                (Reset   (CurveModify_Return))
                (Reset   (MainMenu_Open))
                (Reset   (MainMenu_Start))
                (Reset   (OutlierAnalysis_Open))
                (Reset   (OutlierAnalysis_Start))
                (Reset   (OutlierAnalysis_Return))
                (Reset   (System_Resets))
        )
)



(@RULE=        R1
        (@LHS=
                (=        (1)        (1))
        )
        (@HYPO=        A_Begin)
        (@RHS=
                (Strategy            (@EXHBWRD=FALSE;))
                (Do      ("MainMenu_Start")       (System_Call))
                (Do      (System_Control)         (System_Return))
        )
)

(@RULE=        R2
        (@LHS=
                (=        (1)        (1))
        )
        (@HYPO=         CurveDevelopment_Open)
        (@RHS=
                (Show   ("msg_20.txt")   (@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
                (LoadKB         ("A3_main.tkb") (@LEVEL=ENABLE;))
                (Show   ("msg_23.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
        )
)

(@RULE=         R3
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CurveDevelopment_Start)
        (@RHS=
                (Reset  (CurveDevelopment_Open))
                (Do     (CurveDevelopment_Open)         (CurveDevelopment_Open))
                (Do     ("ON")  (Lock_CrvDev))
                (Do     (C_Stn.Dat_Tbl) (Qry_Str_Prefix))
                (Do     (1)     (Phase))
                (Do     ("CrvDev_Ctl_0")        (CrvDev_Call))
                (Reset  (CrvDev_Control))
                (Do     (CrvDev_Control)        (CrvDev_Control))
                (UnloadKB       ("A3_main.tkb") (@LEVEL=WIPEOUT;))
        )
)

(@RULE=         R11
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CurveModify_Open)
        (@RHS=
                (Show   ("msg_40.txt")   (@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
                (LoadKB         ("A5_main.tkb") (@LEVEL=ENABLE;))
                (Show   ("msg_41.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
        )
)

(@RULE=         R13
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CurveModify_Start)
        (@RHS=
                (Reset  (CurveModify_Open))
                (Do     (CurveModify_Open)      (CurveModify_Open))
                (Do     ("CrvMod_Ctl_2")        (CrvMod_Call))
                (Do     ("ON")  (Lock_CrvMod))
                (Reset  (CrvMod_Control))
                (Do     (CrvMod_Control)        (CrvMod_Control))
                (UnloadKB       ("A5_main.tkb") (@LEVEL=WIPEOUT;))
        )
)


(@RULE=         R20
        @COMMENTS="This rule is run after returning from Curve Modification, if the modelling
procedure was required. It continues by re-calling Curve Development with Phase=5. This starts a specific
set of actions within Curve Development to model the newly combined OLD & NEW datasets. These are
stored in ds.OLD_NEW.";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CurveModify_Return)
        (@RHS=
                (Reset  (CurveDevelopment_Open))
                (Do     (CurveDevelopment_Open)         (CurveDevelopment_Open))
                (Do     ("ON")  (Lock_CrvDev))
                (Do     (5)     (Phase))
                (Do     ("CrvDev_Ctl_0")        (CrvDev_Call))
                (Reset  (CrvDev_Control))
                (Do     (CrvDev_Control)        (CrvDev_Control))
                (UnloadKB       ("A3_main.tkb") (@LEVEL=WIPEOUT;))
        )
)



(@RULE=         R4
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         MainMenu_Open)
        (@RHS=
                (Show   ("Txt/title.txt")   (@KEEP=FALSE;@WAIT=TRUE;@RECT=0,0,600,400;\
))
                (Show   ("msg_5.txt")       (@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
                (LoadKB         ("A2_main.tkb") (@LEVEL=ENABLE;))
        )
)

(@RULE=         R5
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         MainMenu_Start)
        (@RHS=
                (Do     (MainMenu_Open)         (System_Return))
                (Do      ("Control_3")     (Control_Next))
                (Do      ("ON")  (Lock_MainMenu))
                (Reset   (MainMenu_Control))
                (Do      (MainMenu_Control)      (MainMenu_Control))
                (UnloadKB        ("A2_main.tkb") (@LEVEL=WIPEOUT;))
        )
)

(@RULE=         R6
        (@LHS=
                (=       (1)      (1))
        )
        (@HYPO=         OutlierAnalysis_Open)
        (@RHS=
                (Show   ("msg_30.txt")    (@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
                (LoadKB          ("A4_main.tkb") (@LEVEL=ENABLE;))
                (Show   ("msg_31.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
        )
)

(@RULE=         R7
        @COMMENTS="This rule is run after returning from Outlier Analysis. It continues by re-calling
Curve Development with Phase=3. This starts a specific set of actions within Curve Development to model
the non-outlier measurements filtered out in the Outlier Analysis";
        (@LHS=
                (=       (1)      (1))
        )
        (@HYPO=         OutlierAnalysis_Return)
        (@RHS=
                (Reset  (CurveDevelopment_Open))
                (Do     (CurveDevelopment_Open)          (CurveDevelopment_Open))
                (Do     ("ON")  (Lock_CrvDev))
                (Do     (3)      (Phase))
                (Do     ("CrvDev_Ctl_0")         (CrvDev_Call))
                (Reset  (CrvDev_Control))
                (Do     (CrvDev_Control)         (CrvDev_Control))
                (UnloadKB        ("A3_main.tkb") (@LEVEL=WIPEOUT;))
        )
)

(@RULE=         R8
        (@LHS=
                (=       (1)      (1))
        )
        (@HYPO=         OutlierAnalysis_Start)
        (@RHS=
                (Reset  (OutlierAnalysis_Open))
                (Do     (OutlierAnalysis_Open)   (OutlierAnalysis_Open))
                (Do     ("OutAnal_Ctl_0")        (OutAnal_Call))
                (Do     ("ON")  (Lock_OutAnal))
```

```
		(Show	("pointer.txt")	(@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
		(Reset	(OutAnal_Control))
		(Do	(OutAnal_Control)	(OutAnal_Control))
		(UnloadKB	("A4_main.tkb")	(@LEVEL=WIPEOUT;))
	)
)

(@RULE=	R10
	(@LHS=
		(Is	(System_Call)	("EndSession"))
	)
	(@HYPO=	System_Control)
	(@RHS=
		(Show	("msg_7.txt")	(@KEEP=FALSE;@WAIT=FALSE;))
		(Execute	("ControlSession")	(@STRING="@RESTART";))
	)
)

(@RULE=	R9
	(@LHS=
		(IsNot	(System_Call)	("EndSession"))
	)
	(@HYPO=	System_Control)
	(@RHS=
		(Do	(\System_Call\Value)	(System_Return))
		(Do	(TRUE)		(System_Resets))
		(Reset	(System_Control))
	)
)


#*****************************************************************
#*  End of KB Code:  A1_main.tkb                                 *
#*****************************************************************
```

```
#****************************************************************
#*  Nexpert Object KB Code:  A2_main.tkb                       *
#****************************************************************
#*  Main Menu & Curve Use Modules                               *
#****************************************************************


(@VERSION=   020)

(@OBJECT=     Control_20
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Control_3
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Control_30
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Control_4
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Control_7
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Control_8
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Control_Complete
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
```

```
)

(@OBJECT=     Control_Resets
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Crv_Check
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Crv_Duties
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Crv_Exist
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Crv_Log1
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Crv_Log2
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Crv_Log3
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     DDTBL_File
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     dum_bool
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=   Get_DDTBL_File
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=   Main_Process
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=   P1
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=   Print_DDTBL
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=   Print_Prompt_1
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=   Seq_Ret_Curves
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=   Seq_Ret_Stations
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=   Show_Stn
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
```

```
)

(@OBJECT=     Stn_Check
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Stn_Duties
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Stn_Entry
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Stn_Log1
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Stn_Log2
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Stn_Log3
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@SLOT=       Control_Resets
          (@CACTIONS=
                    (Reset   (Control_3))
                    (Reset   (Control_4))
                    (Reset   (Control_20))
                    (Reset   (Control_30))
                    (Reset   (Control_7))
                    (Reset   (Control_8))
                    (Reset   (Control_Resets))
          )
)

(@SLOT=       Crv_Entry
        @PROMPT="ENTER  Curve Id.  NUMBER TO SELECT AN \"ACTIVE CURVE\" ...";
)

(@SLOT=         Crv_Log1
        (@CACTIONS=
                (Execute          ("cat clsum.hd > clsum.txt")        (@TYPE=EXE;))
                (Reset    (Crv_Log1))
        )
)

(@SLOT=         Crv_Log2
        (@CACTIONS=
                (Execute          ("echo @V(Temp_Crv.Value) > clsum.IN")          (@TYPE=EXE;))
                (Execute          ("echo @V(Temp_Crv.Last_Msr) >> clsum.IN")     (@TYPE=EXE;))
                (Do     (STRCAT(Temp_Crv.Crv_Type,STRCAT(":
",STRCAT(INT2STR(Temp_Crv.F_Day),\
STRCAT(" to ",INT2STR(Temp_Crv.L_Day))))))    (dummy_string))
                (Execute          ("echo @V(dummy_string) >> clsum.IN")  (@TYPE=EXE;))
                (Execute          ("echo @V(Temp_Crv.Min_S) >> clsum.IN")        (@TYPE=EXE;))
                (Execute          ("echo @V(Temp_Crv.Max_S) >> clsum.IN")        (@TYPE=EXE;))
                (Execute          ("echo @V(Temp_Crv.Min_D) >> clsum.IN")        (@TYPE=EXE;))
                (Execute          ("echo @V(Temp_Crv.Max_D) >> clsum.IN")        (@TYPE=EXE;))
                (Execute          ("clsum")          (@TYPE=EXE;))
                (Execute          ("cat clsum.OUT >> clsum.txt")    (@TYPE=EXE;))
                (Reset    (Crv_Log2))
        )
)

(@SLOT=         Crv_Log3
        (@CACTIONS=
                (Execute          ("cat clsum.ft >> clsum.txt")        (@TYPE=EXE;))
                (Reset    (Crv_Log3))
        )
)

(@SLOT=         DDTBL_File
        @PROMPT="Enter File Name containing water level records ...";
)

(@SLOT=         Main_Process
        @PROMPT="WHAT WOULD YOU LIKE TO DO NEXT ...";
)

(@SLOT=         P1
        @COMMENTS="Prepare DDTAB input files and execute.";
        (@CACTIONS=
                (Execute          ("cat @V(DDTBL_File.Value) > ddtbl.IN2")          (@TYPE=EXE;))
                (Execute          ("echo @V(C_Crv.TX) > ddtbl.IN1")        (@TYPE=EXE;))
                (Execute          ("echo @V(C_Crv.B0) >> ddtbl.IN1")       (@TYPE=EXE;))
                (Execute          ("echo @V(C_Crv.B1) >> ddtbl.IN1")       (@TYPE=EXE;))
                (Execute          ("ddtbl")          (@TYPE=EXE;))
                (Execute        ("cat ddtbl.OUT")        (@TYPE=EXE;))
                (Execute        ("cat ddtbl.OUT > ddtbl.txt")        (@TYPE=EXE;))
                (Reset   (P1))
        )
)

(@SLOT=         Print_Prompt_1
        @PROMPT="WOULD YOU LIKE A HARD COPY OF THE DISCHARGE TABLE ?";
)

(@SLOT=         Show_Stn
        (@CACTIONS=
                (Do      (STRCAT("'",STRCAT(C_Stn.Stn_Name,"'")))      (dummy_string))
                (Execute        ("mg1 @V(C_Stn.Stn_Id) @V(dummy_string)")      (@TYPE=EXE;))
                (Show    ("msg_1.txt")     (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset   (Show_Stn))
        )
)

(@SLOT=         Stn_Entry
        @PROMPT="ENTER  Station Id.  TO SELECT AN \"ACTIVE STATION\" ...";
)

(@SLOT=         Stn_Log1
        (@CACTIONS=
                (Execute        ("cat stnlst.hd > stnlst.txt")        (@TYPE=EXE;))
                (Reset   (Stn_Log1))
        )
)

(@SLOT=         Stn_Log2
        (@CACTIONS=
                (Execute        ("echo @V(C_Stn.Stn_Id) > stnlst.IN")      (@TYPE=EXE;))
                (Execute        ("echo @V(C_Stn.Stn_Name) >> stnlst.IN")        (@TYPE=EXE;))
                (Execute        ("stnlst")        (@TYPE=EXE;))
                (Execute        ("cat stnlst.OUT >> stnlst.txt")      (@TYPE=EXE;))
                (Reset   (Stn_Log2))
        )
)

(@SLOT=         Stn_Log3
        (@CACTIONS=
                (Execute        ("cat stnlst.ft >> stnlst.txt")        (@TYPE=EXE;))
                (Reset   (Stn_Log3))
        )
)

(@RULE=         Modify_Curve_Part1
        (@LHS=
                (Is      (Main_Process)   ("Modify an Existing Curve"))
        )
```

```
          (@HYPO=        Control_20)
          (@RHS=
                    (Do       ("Control_7")     (Control_Next))
          )
)

(@RULE=         Make_New_Curve_Part1
          @COMMENTS="Signals to MainMenu_Control to exit from main menu and Start the Curve
Development Module";
          (@LHS=
                    (Is       (Main_Process)   ("Make New Curve"))
          )
          (@HYPO=        Control_20)
          (@RHS=
                    (Do       ("CurveDevelopment_Start")        (System_Call))
                    (Do       ("OFF")           (Lock_MainMenu))
          )
)

(@RULE=         Exit_System
          @COMMENTS="Signals to MainMenu_Control to exit from main menu and End the system
session";
          (@LHS=
                    (Is       (Main_Process)   ("Exit System"))
          )
          (@HYPO=        Control_20)
          (@RHS=
                    (Do       ("EndSession")   (System_Call))
                    (Do       ("OFF")          (Lock_MainMenu))
          )
)

(@RULE=         Compute_Dishcharge_Table_Part1
          (@LHS=
                    (Is       (Main_Process)   ("Compute Daily Discharges"))
          )
          (@HYPO=        Control_20)
          (@RHS=
                    (Do       ("Control_7")     (Control_Next))
          )
)

(@RULE=         Retrv_Station_List
          (@LHS=
                    (=        (1)       (1))
          )
          (@HYPO=        Control_3)
          (@RHS=
                    (Show     ("msg_9.txt")     (@KEEP=FALSE;@WAIT=FALSE;))
                    (Do       (0)       (Counter))
                    (Do       (0)       (Cursor))
```

```
                (Do     (TRUE)          (Stn_Log1))
                (Do     (Seq_Ret_Stations)      (Seq_Ret_Stations))
                (Reset  (Seq_Ret_Stations))
                (Do     (TRUE)          (Stn_Log3))
                (Show   ("stnlst.txt")  (@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
                (Do     ("Control_4")   (Control_Next))
        )
)

(@RULE=         Modify_Curve_Part2
        (@LHS=
                (Is     (Main_Process)  ("Modify an Existing Curve"))
        )
        (@HYPO=         Control_30)
        (@RHS=
                (Do     ("CurveModify_Start")   (System_Call))
                (Do     ("OFF")         (Lock_MainMenu))
        )
)

(@RULE=         Compute_Dishcharge_Table_Part2
        (@LHS=
                (Is     (Main_Process)  ("Compute Daily Discharges"))
        )
        (@HYPO=         Control_30)
        (@RHS=
                (Do     (Get_DDTBL_File)        (Get_DDTBL_File))
                (Do     (TRUE)          (P1))
                (Show   ("msg_6.txt")   (@KEEP=FALSE;@WAIT=TRUE;))
                (Show   ("ddtbl.txt")   (@KEEP=FALSE;@WAIT=TRUE;@RECT=5,5,1150,820;\
))
                (Show   ("msg_6.txt")   (@KEEP=FALSE;@WAIT=TRUE;))
                (Reset  (Print_DDTBL))
                (Do     (Print_DDTBL)   (Print_DDTBL))
                (Reset  (DDTBL_File))
                (Reset  (Main_Process))
                (Do     ("Control_20")  (Control_Next))
                (Do     ("ON")  (Lock_MainMenu))
        )
)

(@RULE=         Assign_Station
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         Control_4)
        (@RHS=
                (Reset  (Stn_Entry))
                (Do     (STRUPPER(Stn_Entry)) (Stn_Entry))
                (Reset  (Cursor))
```

```
                    (Retrieve           ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=C_Stn.Stn_Id;\
@FIELDS="STN_ID";@QUERY="STN_LIST where STN_ID = '@V(Stn_Entry)'";\
@CURSOR=Cursor;))
                    (Reset   (Stn_Check))
                    (Do      (Stn_Check)      (Stn_Check))
          )
)

(@RULE=         Retrv_Curve_List
          (@LHS=
                    (=       (1)       (1))
          )
          (@HYPO=        Control_7)
          (@RHS=
                    (Show    ("msg_12.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
                    (Do      (0)       (Counter))
                    (Do      (0)       (Cursor))
                    (Do      (TRUE)              (Crv_Log1))
                    (Do      (Seq_Ret_Curves)            (Seq_Ret_Curves))
                    (Reset   (Seq_Ret_Curves))
                    (Do      (TRUE)              (Crv_Log3))
                    (Reset   (Crv_Exist))
                    (Do      (Crv_Exist)        (Crv_Exist))
          )
)

(@RULE=         Get_Current_Curve
          (@LHS=
                    (=       (1)       (1))
          )
          (@HYPO=        Control_8)
          (@RHS=
                    (Reset   (Crv_Entry))
                    (Do      (Crv_Entry)        (Crv_Entry))
                    (Reset   (Cursor))
                    (Retrieve           ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=C_Crv.Value;\
@FIELDS="COUNT(CRV_ID)";@QUERY="@V(C_Stn.Crv_Tbl) where CRV_ID = @V(Crv_Entry)";\
@CURSOR=Cursor;))
                    (Reset   (Crv_Check))
                    (Do      (Crv_Check)      (Crv_Check))
          )
)

(@RULE=         Valid_Curve
          @INFCAT=10;
          @COMMENTS="Sets the CURRENT CURVE ...";
          (@LHS=
                    (>=      (Cursor)            (0))
          )
          (@HYPO=        Crv_Check)
          (@RHS=
                    (Reset   (Cursor))
                    (Retrieve          ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=C_Crv.Crv_Id,C_Crv.Last_Msr,
C_Crv.Crv_Type,C_Crv.B0,C_Crv.B1,C_Crv.F_Day,C_Crv.L_Day,
C_Crv.Min_S,C_Crv.Max_S,C_Crv.Min_D,C_Crv.Max_D,
C_Crv.Datemade,C_Crv.TX;@FIELDS="CRV_ID","LAST_MSR",
"TYPE","B0","B1","F_DAY","L_DAY","MIN_S",
"MAX_S","MIN_D","MAX_D","DATEMADE","TX";@QUERY="@V(C_Stn.Crv_Tbl) where CRV_ID
=
 @V(Crv_Entry)";@CURSOR=Cursor;))
                    (Do      ("Control_30")   (Control_Next))
          )
)

(@RULE=        Invalid_Curve
          @INFCAT=5;
          (@LHS=
                    (<       (Cursor)         (0))
                    (>       (C_Crv)          (0))
          )
          (@HYPO=        Crv_Check)
          (@RHS=
                    (Show    ("msg_11.txt")   (@KEEP=FALSE;@WAIT=TRUE;@RECT=5,400,500,\
300;))
                    (Show    ("clsum.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
                    (Do      ("Control_8")    (Control_Next))
          )
)

(@RULE=        Curve_Write_Check
          (@LHS=
                    (>=      (Cursor)         (0))
                    (>       (Temp_Crv)       (0))
          )
          (@HYPO=        Crv_Duties)
          (@RHS=
                    (Do      (TRUE)           (Crv_Log2))
          )
)

(@RULE=        No_Curves_Exist_chk
          @COMMENTS="Checks for at least one curve stored";
          (@LHS=
                    (<=      (Counter)        (1))
          )
          (@HYPO=        Crv_Exist)
          (@RHS=
                    (Show    ("msg_13.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
                    (Reset   (Main_Process))
```

```
                (Do      ("Control_20")   (Control_Next))
        )
)

(@RULE=         Curve_Exist_check
        @COMMENTS="Checks for at least one curve stored";
        (@LHS=
                (>       (Counter)        (1))
        )
        (@HYPO=         Crv_Exist)
        (@RHS=
                (Show    ("clsum.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
                (Do      ("Control_8")    (Control_Next))
        )
)

(@RULE=         R116
        @INFCAT=5;
        (@LHS=
                (Name    (DDTBL_File)     (DDTBL_File))
                (Execute          ("FileExist")
(@STRING="@FILE=@V(DDTBL_File.Value),@RETURN=dum_bool";\
))
                (Yes     (dum_bool))
        )
        (@HYPO=         Get_DDTBL_File)
)

(@RULE=         R115
        (@LHS=
                (No      (dum_bool))
        )
        (@HYPO=         Get_DDTBL_File)
        (@RHS=
                (Show    ("badfile")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset   (DDTBL_File))
                (Reset   (Get_DDTBL_File))
        )
)

(@RULE=         R118
        (@LHS=
                (Is      (Lock_MainMenu)          ("OFF"))
        )
        (@HYPO=         MainMenu_Control)
)

(@RULE=         R117
        @INFCAT=5;
        (@LHS=
                (Is      (Lock_MainMenu)          ("ON"))
        )
        (@HYPO=     MainMenu_Control)
        (@RHS=
                (Do     (\Control_Next\Value)   (Control_Complete))
                (Do     (TRUE)          (Control_Resets))
                (Reset  (MainMenu_Control))
        )
)

(@RULE=     R120
        (@LHS=
                (Is     (Print_Prompt_1)        ("YES"))
        )
        (@HYPO=     Print_DDTBL)
        (@RHS=
                (Execute        ("lpr -Psparc -h ddtbl.OUT")      (@TYPE=EXE;))
                (Reset  (Print_Prompt_1))
        )
)

(@RULE=     R119
        (@LHS=
                (Is     (Print_Prompt_1)        ("NO"))
        )
        (@HYPO=     Print_DDTBL)
)

(@RULE=     R121
        (@LHS=
                (>=     (Cursor)        (0))
        )
        (@HYPO=     Seq_Ret_Curves)
        (@RHS=
                (Do     (Counter+1)     (Counter))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=Temp_Crv.Value,\
Temp_Crv.Crv_Type,Temp_Crv.F_Day,Temp_Crv.L_Day,\
Temp_Crv.Datemade,Temp_Crv.Min_S,Temp_Crv.Max_S,\
Temp_Crv.Min_D,Temp_Crv.Max_D,Temp_Crv.Last_Msr;\
@FIELDS="CRV_ID","TYPE","F_DAY","L_DAY","DATEMADE",\
"MIN_S","MAX_S","MIN_D","MAX_D","LAST_MSR";\
@QUERY="@V(C_Stn.Crv_Tbl)";@CURSOR=Cursor;\
))
                (Do     (Crv_Duties)    (Crv_Duties))
                (Reset  (Crv_Duties))
                (Reset  (Seq_Ret_Curves))
        )
)

(@RULE=     R122
        (@LHS=
                (>=     (Cursor)        (0))
        )
        (@HYPO=         Seq_Ret_Stations)
        (@RHS=
                (Do     (Counter+1)     (Counter))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@FWRD=TRUE;@END="RELEASE";\
@SLOTS=C_Stn.Stn_Id,C_Stn.Stn_Name;@FIELDS="STN_ID",\
"NAME";@QUERY="STN_LIST";@CURSOR=Cursor;))
                (Do     (Stn_Duties)    (Stn_Duties))
                (Reset  (Stn_Duties))
                (Reset  (Seq_Ret_Stations))
        )
)

(@RULE=         Valid_Station
        @INFCAT=10;
        @COMMENTS="Sets the CURRENT CURVE ...";
        (@LHS=
                (>=     (Cursor)        (0))
        )
        (@HYPO=         Stn_Check)
        (@RHS=
                (Reset  (Cursor))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=C_Stn.Stn_Name,\
C_Stn.Dat_Tbl,C_Stn.Crv_Tbl,C_Stn.Atr_Tbl;\
@FIELDS="NAME","NXP_VIEW","LRM_TAB","ATR_TAB";\
@QUERY="STN_LIST where STN_ID = '@V(C_Stn.Stn_Id)'";\
@CURSOR=Cursor;))
                (Do     (TRUE)          (Show_Stn))
                (Do     ("Control_20")  (Control_Next))
        )
)

(@RULE=         Invalid_Station
        @INFCAT=5;
        (@LHS=
                (<      (Cursor)        (0))
        )
        (@HYPO=         Stn_Check)
        (@RHS=
                (Show   ("msg_10.txt")  (@KEEP=FALSE;@WAIT=TRUE;@RECT=5,400,500,\
300;))
                (Show   ("stnlst.txt")  (@KEEP=FALSE;@WAIT=FALSE;))
                (Do     ("Control_4")   (Control_Next))
        )
)

(@RULE=         Station_Write_Check
        (@LHS=
          (>=      (Cursor)        (0))
      )
      (@HYPO=      Stn_Duties)
      (@RHS=
          (Do      (TRUE)          (Stn_Log2))
      )
)


#*****************************************************************
#*  End of KB Code:  A2_main.tkb                                 *
#*****************************************************************
```

```
#*****************************************************************
#*  Nexpert Object KB Code:  A3_main.tkb                         *
#*****************************************************************
#*  Curve Development Module                                     *
#*****************************************************************


(@VERSION=   020)
(@PROPERTY=          Date_Made       @TYPE=Date;)
(@PROPERTY=          FD      @TYPE=Integer;)
(@PROPERTY=          FM      @TYPE=Integer;)
(@PROPERTY=          LD      @TYPE=Integer;)
(@PROPERTY=          Level_1         @TYPE=Integer;)
(@PROPERTY=          Level_2         @TYPE=Integer;)
(@PROPERTY=          LM      @TYPE=Integer;)
(@PROPERTY=          Opt_p1  @TYPE=String;)
(@PROPERTY=          Opt_p2  @TYPE=String;)
(@PROPERTY=          Opt_p3  @TYPE=String;)
(@PROPERTY=          Opt_p5  @TYPE=String;)
(@PROPERTY=          Start_Day       @TYPE=Integer;)
(@PROPERTY=          Stop_Day        @TYPE=Integer;)


(@OBJECT=     Alter_Progress
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Another_model
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Best_Model_Crit
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Best_NOL
        (@PROPERTIES=
                Value    @TYPE=Integer;
        )
)

(@OBJECT=     Best_R2
        (@PROPERTIES=
                Value   @TYPE=Float;
        )
)

(@OBJECT=     Best_StdEE
        (@PROPERTIES=
                Value   @TYPE=Float;
        )
)

(@OBJECT=     Best_Xtras
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Chk_Exponent
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Choose_best
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Clear_Files
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Created_New_Curve
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     CrvDev_Ctl_0
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     CrvDev_Ctl_1
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
```

```
)

(@OBJECT=     CrvDev_Ctl_2
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_3
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_4
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_M
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_OL_Load
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_R
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_Resets
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_SBM
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvDev_Ctl_SW2
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Ctrl_0
          (@PROPERTIES=
                    Opt_p1
                    Opt_p2
          )
)

(@OBJECT=     Ctrl_1
          (@PROPERTIES=
                    Opt_p1
                    Opt_p2
                    Opt_p3
          )
)

(@OBJECT=     Ctrl_2
          (@PROPERTIES=
                    Opt_p1
                    Opt_p2
                    Opt_p3
                    Opt_p5
          )
)

(@OBJECT=     Ctrl_3
          (@PROPERTIES=
                    Opt_p1
                    Opt_p2
                    Opt_p3
          )
)

(@OBJECT=     ctrl_inits
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Current_DS
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Cursor2
          (@PROPERTIES=
                Value    @TYPE=Integer;
        )
)

(@OBJECT=       Deactivate_plots
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       Do_Next_Task
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       DS_Create
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       DS_Name
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=       DS_num
        (@PROPERTIES=
                Value    @TYPE=Integer;
        )
)

(@OBJECT=       DS_Remove
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       DS_Reset
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=       dummy_file
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)
```

```
(@OBJECT=	dummy_float
	(@PROPERTIES=
		Value	@TYPE=Float;
	)
)

(@OBJECT=	dummy_infile
	(@PROPERTIES=
		Value	@TYPE=String;
	)
)

(@OBJECT=	dummy_integer
	(@PROPERTIES=
		Value	@TYPE=Integer;
	)
)

(@OBJECT=	dummy_obj
	(@PROPERTIES=
		Value	@TYPE=String;
	)
)

(@OBJECT=	dummy_outfile
	(@PROPERTIES=
		Value	@TYPE=String;
	)
)

(@OBJECT=	dummy_title
	(@PROPERTIES=
		Value	@TYPE=String;
	)
)

(@OBJECT=	File_Erase_1
	(@PROPERTIES=
		Value	@TYPE=Boolean;
	)
)

(@OBJECT=	Find_best
	(@PROPERTIES=
		Value	@TYPE=Boolean;
	)
)

(@OBJECT=	Find_best_R2_AD
	(@PROPERTIES=
		Value	@TYPE=Boolean;
```

```
          )
)

(@OBJECT=     Find_best_R2_SD
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Find_best_SSE_AD
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Find_best_SSE_SD
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     First_day
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     First_month
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     Gen_Models
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Get_Seq_Data
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Init_Ctrls
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)
```

```
(@OBJECT=      Input_Exp
         (@PROPERTIES=
                   Value    @TYPE=Float;
         )
)

(@OBJECT=      Input_TimeRange
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=      Last_day
         (@PROPERTIES=
                   Value    @TYPE=Integer;
         )
)

(@OBJECT=      Last_month
         (@PROPERTIES=
                   Value    @TYPE=Integer;
         )
)

(@OBJECT=      Level_1
         (@PROPERTIES=
                   Type
                   Value    @TYPE=Integer;
         )
)

(@OBJECT=      List_curve_A
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=      List_curve_B
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=      List_data_A
         (@PROPERTIES=
                   Value    @TYPE=Boolean;
         )
)

(@OBJECT=      List_data_B
         (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      Loop_1
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      MakeNewCurve
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      MD_Name
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=      Misc_duties
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      ML_Create
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      ML_Name
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=      ML_Remove
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      ML_Remove_All
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)
```

```
(@OBJECT=     Model
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Model_Count
        (@PROPERTIES=
                Value    @TYPE=Integer;
        )
)

(@OBJECT=     Model_inc
        (@PROPERTIES=
                Value    @TYPE=Integer;
        )
)

(@OBJECT=     Model_response
        (@PROPERTIES=
                Value    @TYPE=Float;
        )
)

(@OBJECT=     neg_limit
        (@PROPERTIES=
                Value    @TYPE=Float;
        )
)

(@OBJECT=     New_Task2
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Next_CrvId
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Next_DS
        (@PROPERTIES=
                Value    @TYPE=Integer;
        )
)

(@OBJECT=     Next_MD
        (@PROPERTIES=
                Value    @TYPE=Integer;
```

```
          )
)

(@OBJECT=     No_Accepted_Model
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Num_models
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     OL_Report
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Out_side
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Outlier_count
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     P2_Ice
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     P2_Stage
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     P2_Time
          (@PROPERTIES=
                    FD
                    FM
                    LD
                    LM
          )
)

(@OBJECT=     Phase_1_junk
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Phase_2_route
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Phs2_A
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Plot_Settings
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)


(@OBJECT=     Prepare_For_Evaluation
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Print_Record
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Qry_Str_Suffix
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Query_Prefix
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)
```

```
(@OBJECT=     Query_Suffix
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Run_Xgraph
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Reset_Datasets
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)
(@OBJECT=     Seq_ML_Remove
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Set_Axis
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Set_Qry_Suffix
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Show_Phase_Title
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Start_Progress
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Station_Entry
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Store_Outlier
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Stored
          (@PROPERTIES=
                    B0
                    B1
                    Crv_Id
                    Crv_Type
                    Date_Made
                    F_Day
                    L_Day
                    Last_Msr
                    Max_D
                    Max_S
                    Min_D
                    Min_S
                    No_Flow_Stg
                    TX
          )
)

(@OBJECT=     Sub_Data
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     T1
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T10
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T11_A
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)
```

```
(@OBJECT=     Get_Zero_Flow
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T12
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T13
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T15
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T20
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T21
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T22
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T23
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T24
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T3
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T40
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T41
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T42
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)
(@OBJECT=     T8
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     T9
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Tmp_String
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Total_OLs
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     Trib_High_DS
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Trib_High_US
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Trib_Low_DS
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Trib_Low_US
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Tributary_q1
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     tweetie
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     Weed
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Wind
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Wind_direction
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
```

```
)

(@OBJECT=     XG_put_curve
         (@PROPERTIES=
                  Value    @TYPE=Boolean;
         )
)

(@OBJECT=     XG_put_data
         (@PROPERTIES=
                  Value    @TYPE=Boolean;
         )
)

(@OBJECT=     XG_put_outliers
         (@PROPERTIES=
                  Value    @TYPE=Boolean;
         )
)

(@OBJECT=     Zero_count
         (@PROPERTIES=
                  Value    @TYPE=Integer;
         )
)

(@SLOT=          Another_model
         @PROMPT="Would your like to run another model?";
)

(@SLOT=          Best_Model_Crit
         @PROMPT="What method is to be used in determining the BEST MODEL ?";
)

(@SLOT=          Best_R2
         @FORMAT="u.0000";
)

(@SLOT=          Best_StdEE
         @FORMAT="u.0000";
)

(@SLOT=          Best_Xtras
         @COMMENTS="This slot renames the UNIX filenames of the BEST model files.  All model files
(ML?.*) are removed when this KB is unloaded.";
         (@CACTIONS=
                  (Execute         ("mv @V(BEST_Model.FN_Curve) bML.mdl")        (@TYPE=EXE;))
                  (Execute         ("mv @V(BEST_Model.FN_Outliers) bML.ol")      (@TYPE=EXE;))
                  (Execute         ("mv @V(BEST_Model.FN_Conlim) bML.cl")        (@TYPE=EXE;))
                  (Execute         ("mv @V(BEST_Model.FN_Results) bML.txt")      (@TYPE=EXE;))
                  (Do      ("bML.mdl")      (BEST_Model.FN_Curve))
```

```
                (Do     ("bML.ol")      (BEST_Model.FN_Outliers))
                (Do     ("bML.cl")      (BEST_Model.FN_Conlim))
                (Do     ("bML.txt")     (BEST_Model.FN_Results))
                (Reset  (Cursor))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="release";@SLOTS=BEST_Model.Crv_Id;\
@FIELDS="max(crv_id) + 1";@QUERY="@V(C_Stn.Crv_Tbl)";\
@CURSOR=Cursor;))
                (Do     (\DS_Name\.Type)        (BEST_Model.Crv_Type))
                (Do     ("BEST")        (\DS_Name\.Title))
                (Reset  (Best_Xtras))
        )
)

(@SLOT=         Choose_best
        (@CACTIONS=
                (Do     (Find_best_SSE_AD)      (Find_best_SSE_AD))
                (Reset  (Find_best_SSE_AD))
                (DeleteObject   (<|Models|>)    (|Models|))
                (Reset  (Choose_best))
        )
)

(@SLOT=         Clear_Files
        (@CACTIONS=
                (Execute        ("rm -f P1.dat")  (@TYPE=EXE;))
                (Execute        ("rm -f P2.dat")  (@TYPE=EXE;))
                (Execute        ("rm -f xg.dat")  (@TYPE=EXE;))
                (Execute        ("rm -f OL_*")    (@TYPE=EXE;))
                (Execute        ("rm -f AD_*")    (@TYPE=EXE;))
                (Execute        ("rm -f SD_*")    (@TYPE=EXE;))
                (Reset  (Clear_Files))
        )
)

(@SLOT=         CrvDev_Ctl_Resets
        (@CACTIONS=
                (Reset  (CrvDev_Ctl_0))
                (Reset  (CrvDev_Ctl_1))
                (Reset  (CrvDev_Ctl_2))
                (Reset  (CrvDev_Ctl_3))
                (Reset  (CrvDev_Ctl_4))
                (Reset  (CrvDev_Ctl_M))
                (Reset  (CrvDev_Ctl_R))
                (Reset  (CrvDev_Ctl_SW2))
                (Reset  (CrvDev_Ctl_SBM))
                (Reset  (CrvDev_Ctl_Resets))
        )
)

(@SLOT=         Ctrl_0.Opt_p1
        @PROMPT="You are at Level 0 (Data Selection) of Phase 1.   PROCEED WITH  ...";
)

(@SLOT=         Ctrl_1.Opt_p1
        @PROMPT="You are at Level 1 (Modelling) of Phase 1.   PROCEED WITH  ...";
)

(@SLOT=         Ctrl_1.Opt_p2
        @PROMPT="You are at Level 1 (Modelling) of Phase 2.   PROCEED WITH ...";
)

(@SLOT=         Ctrl_1.Opt_p3
        @PROMPT="You are at Level 1 (Modelling) of Phase 3.   PROCEED WITH ...";
)

(@SLOT=         Ctrl_2.Opt_p1
        @PROMPT="You are at Level 2 (Model Evaluation) of Phase 1.    PROCEED WITH  ...";
)

(@SLOT=         Ctrl_2.Opt_p2
        @PROMPT="You are at Level 2 (Model Evaluation) of Phase 2.   PROCEED WITH ...";
)

(@SLOT=         Ctrl_2.Opt_p3
        @PROMPT="You are at Level 2 (Model Evaluation) of Phase 3.   PROCEED WITH ...";
)

(@SLOT=         Ctrl_2.Opt_p5
        @PROMPT="You are at Level 2 (Model Evaluation) of Phase 5.   PROCEED WITH ...";
)

(@SLOT=         Ctrl_3.Opt_p1
        @PROMPT="You are at Level 3 (Model Processing) of Phase 1.   PROCEED WITH  ...";
)

(@SLOT=         Ctrl_3.Opt_p2
        @PROMPT="You are at Level 3 (Model Processing) of Phase 2.   PROCEED WITH  ...";
)

(@SLOT=         Ctrl_3.Opt_p3
        @PROMPT="You are at Level 3 (Model Processing) of Phase 3.   PROCEED WITH  ...";
)

(@SLOT=         Deactivate_plots
        (@CACTIONS=
                (Do     (FALSE)         (\Current_DS\.Plot))
                (Let    (<|Models|>.Plot)          (FALSE))
                (Reset  (Deactivate_plots))
        )
)
```

```
(@SLOT=       DS_Create
       (@CACTIONS=
              (Do      (STRCAT(DS_Name,".*"))          (dummy_string))
              (Execute          ("rm -f @V(dummy_string)")       (@TYPE=EXE;))
              (Reset   (DS_Create))
       )
)

(@SLOT=       DS_Remove
       (@CACTIONS=
              (Execute          ("ResetFrame")   (@ATOMID=\DS_Name.value\;))
              (Do      (STRCAT(DS_Name,".*"))          (dummy_string))
              (Execute          ("rm -f @V(dummy_string)")       (@TYPE=EXE;))
              (Do      (0)      (Start_day))
              (Do      (0)      (Stop_day))
              (Reset   (DS_Remove))
       )
)

(@SLOT=       DS_Reset
       (@CACTIONS=
              (Execute          ("ResetFrame")   (@ATOMID=\DS_Name.value\;))
              (Do      (STRCAT(DS_Name,".dat"))        (dummy_string))
              (Execute          ("rm -f @V(dummy_string)")       (@TYPE=EXE;))
              (Reset   (DS_Reset))
       )
)

(@SLOT=       dummy_float
       @FORMAT="u.0000";
)

(@SLOT=       File_Erase_1
       (@CACTIONS=
              (Execute          ("rm -f xgdat")   (@TYPE=EXE;))
              (Reset   (File_Erase_1))
       )
)

(@SLOT=       First_day
       @PROMPT="\"Enter DAY of First Day in the time range (value = 1..31) ...";
)

(@SLOT=       First_month
       @PROMPT="\"Enter MONTH of First Day in the time range (value = 1..12) ...";
)

(@SLOT=       Init_Ctrls
       @COMMENTS="This Slot is used to Rest all Control option values.";
       (@CACTIONS=
              (Reset   (Ctrl_0.Opt_p1))
```

```
                (Reset   (Ctrl_0.Opt_p2))
                (Reset   (Ctrl_1.Opt_p1))
                (Reset   (Ctrl_1.Opt_p2))
                (Reset   (Ctrl_1.Opt_p3))
                (Reset   (Ctrl_2.Opt_p1))
                (Reset   (Ctrl_2.Opt_p2))
                (Reset   (Ctrl_2.Opt_p3))
                (Reset   (Ctrl_2.Opt_p5))
                (Reset   (Ctrl_3.Opt_p1))
                (Reset   (Ctrl_3.Opt_p2))
                (Reset   (Ctrl_3.Opt_p3))
                (Reset   (Init_Ctrls))
        )
)

(@SLOT=         Input_Exp
        @PROMPT="Please enter a REAL number for the Transformation Exponent (TX), in the box
provided ...";
)

(@SLOT=         Input_TimeRange
        (@CACTIONS=
                (Execute        ("resetframe")    (@ATOMID=P2_Time;))
                (Do     (P2_Time.FM)    (P2_Time.FM))
                (Do     (P2_Time.FD)    (P2_Time.FD))
                (Do     (P2_Time.LM)    (P2_Time.LM))
                (Do     (P2_Time.LD)    (P2_Time.LD))
                (Do     (STRCAT("1900,",INT2STR(P2_Time.FM)))       (Tmp_String))
                (Do     (STRCAT(Tmp_String,","))        (Tmp_String))
                (Do     (STRCAT(Tmp_String,INT2STR(P2_Time.FD)))    (Tmp_String))
                (Do     (YEARDAY(STR2DATE(Tmp_String,"yyyy,m,d")))          (Start_day))
                (Do     (STRCAT("1900,",INT2STR(P2_Time.LM)))       (Tmp_String))
                (Do     (STRCAT(Tmp_String,","))        (Tmp_String))
                (Do     (STRCAT(Tmp_String,INT2STR(P2_Time.LD)))    (Tmp_String))
                (Do     (YEARDAY(STR2DATE(Tmp_String,"yyyy,m,d")))          (Stop_day))
                (Reset  (Input_TimeRange))
        )
)

(@SLOT=         Last_day
        @PROMPT="\"Enter DAY of Last Day in the time range (value = 1..31) ...";
)

(@SLOT=         Last_month
        @PROMPT="\"Enter MONTH of Last Day in the time range (value = 1..12) ...";
)

(@SLOT=         ML_Create
        (@CACTIONS=
                (CreateObject    (\ML_Name\)    (|Models|))
                (Do     (STRCAT(ML_Name,".*"))          (dummy_string))
```

```
                    (Execute        ("rm -f @V(dummy_string)")      (@TYPE=EXE;))
                    (Do     (Start_day)       (\ML_Name\F_Day))
                    (Do     (Stop_day)        (\ML_Name\L_Day))
                    (Reset  (ML_Create))
          )
)

(@SLOT=        ML_Remove
          @COMMENTS="Deletes Object and Unix files named by ML_Name ";
          (@CACTIONS=
                    (DeleteObject    (\ML_Name\))
                    (Do     (STRCAT(ML_Name,".*"))        (dummy_string))
                    (Execute        ("rm -f @V(dummy_string)")      (@TYPE=EXE;))
                    (Reset  (ML_Remove))
          )
)

(@SLOT=        ML_Remove_All
          (@CACTIONS=
                    (Do     (1)       (Counter))
                    (Do     (Seq_ML_Remove)       (Seq_ML_Remove))
                    (Reset  (Seq_ML_Remove))
                    (Reset  (ML_Remove_All))
          )
)

(@SLOT=        Next_DS
          (@INITVAL=    1)
)

(@SLOT=        Next_MD
          (@INITVAL=    1)
)

(@SLOT=        Num_models
          @PROMPT="How many Linear Regression models would you like to fit to the data?";
)

(@SLOT=        P2_Ice
          @PROMPT="Data should be stratified under one of the following conditions. Select ONE ...";
)

(@SLOT=        P2_Stage
          @PROMPT="Data should be stratified under one of the following conditions. Select ONE ...";
)

(@SLOT=        P2_Time.FD
          @PROMPT="\"Enter DAY of First Day in the time range (value = 1..31) ...";
)

(@SLOT=        P2_Time.FM
        @PROMPT="\"Enter MONTH of First Day in the time range (value = 1..12) ...";
)

(@SLOT=         P2_Time.LD
        @PROMPT="\"Enter DAY of Last Day in the time range (value = 1..31) ...";
)

(@SLOT=         P2_Time.LM
        @PROMPT="\"Enter MONTH of Last Day in the time range (value = 1..12) ...";
)

(@SLOT=         Phase.Type
        @PROMPT="How should the data be stratified, specifically?  By ...";
)

(@SLOT=         Phase
        (@INITVAL=    1)
)

(@SLOT=         Phs2_A
        @PROMPT="How should the data be stratified?  By ...";
)

(@SLOT=         Plot_Info.Filename
        (@INITVAL=    "xg.dat")
)

(@SLOT=         Prepare_For_Evaluation
        (@CACTIONS=
                (Execute        ("cat  mmsum.foot >> mmsum.txt")        (@TYPE=EXE;))
                (Show   ("mmsum.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset  (Best_Model_Crit))
                (Reset  (Prepare_For_Evaluation))
        )
)

(@SLOT=         Run_Xgraph
        (@CACTIONS=
                (Do     (C_Stn.Stn_Id)  (Plot_Info.Title))
                (Do     (Set_Axis)      (Set_Axis))
                (Reset  (Set_Axis))
                (Do     (STRCAT(FLOAT2STR(Plot_Info.Max_S*1.25),STRCAT(" ",\
FLOAT2STR(Plot_Info.Max_D*1.25))))   (dummy_string))
                (Execute        ("XGR   @V(Plot_Info.Title)   @v(Plot_Info.Filename)
@V(dummy_string)")   (@TYPE=EXE;))
                (Reset  (Run_Xgraph))
        )
)

(@SLOT=         Reset_Datasets
        (@CACTIONS=
                (Do       (1)       (count))
                (Reset    (Plot_Settings))
                (Do       (Plot_Settings)    (Plot_Settings))
                (Reset    (Reset_Datasets))
        )
)

(@SLOT=         Start_Progress
        (@CACTIONS=
                (Execute          ("cat mmsum.head > mmsum.txt")          (@TYPE=EXE;))
                (Reset    (Start_Progress))
        )
)

(@SLOT=         Stored.Date_Made
        @FORMAT="d\"-\"MMM\"-\"yy";
)

(@SLOT=         Stored.F_Day
        (@INITVAL=    0)
)

(@SLOT=         Stored.L_Day
        (@INITVAL=    0)
)

(@SLOT=         Sub_Data
        @PROMPT="What data set would you like to model next?";
)

(@SLOT=         T10
        @COMMENTS="Retreives data according to QUERY_STRING.";
        (@CACTIONS=
                (Do       ("xgdat")          (Plot_Info.Filename))
                (Execute          ("rm -f @V(Plot_Info.Filename)") (@TYPE=EXE;))
                (Execute          ("rm -f mmsum.txt")       (@TYPE=EXE;))
                (Do       (TRUE)             (DS_Create))
                (Do       (0)       (Model_Count))
                (Do       (TRUE)             (Start_Progress))
                (Reset    (T10))
        )
)

(@SLOT=         T11_A
        @COMMENTS="Object dummy_file stores the UNIX filename into which the extrated
measurements are copied.";
        (@CACTIONS=
                (Show     ("msg_8.txt")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Do       (STRCAT(DS_Name,".dat"))         (\DS_Name\FN_Data))
                (Do       (\DS_Name\FN_Data)    (dummy_file))
                (Do       (Phase.Type)       (\DS_Name\Type))
                (Do      (STRUPPER(\DS_Name\FN_Data))          (\DS_Name\Title))
                (Do      (TRUE)            (\DS_Name\Plot))
                (Do      (0.0)     (\DS_Name\No_Flow_Stg))
                (Reset   (Get_Zero_Flow))
                (Do      (Get_Zero_Flow)            (Get_Zero_Flow))
                (Reset   (Cursor))
                (Retrieve            ("@V(db_access)")
(@TYPE=ORACLE;@SLOTS=\DS_Name\Max_D,\DS_Name\Max_S,
\DS_Name\Num_Msrs,\DS_Name\Min_S,\DS_Name\Min_D,
\DS_Name\Last_Msr;@FIELDS="MAX(DISCHARGE)",
"MAX(STAGE)","COUNT(DATE_OF_MSR)","MIN(STAGE)",
"MIN(DISCHARGE)","MAX(DATE_OF_MSR)";@QUERY="@V(Query_String)";
@CURSOR=Cursor;))
                (Do      (0)       (Cursor))
                (Do      (0)       (Count))
                (Reset   (Get_Seq_Data))
                (Do      (Get_Seq_Data)  (Get_Seq_Data))
                (Reset   (T11_A))
        )
)

(@SLOT=         T12
        @COMMENTS="Prepare Datasets for XGRAPH plotting";
        (@CACTIONS=
                (Do      (List_data_A)     (List_data_A))
                (Reset   (List_data_A))
                (Do      (STRCAT(INT2STR(\DS_Name\.Num_Msrs),STRCAT("
",STRCAT(INT2STR(Zero_count),STRCAT(" ",Phase.Type)))))        (dummy_params))
                (Execute            ("mg2 @V(dummy_params)")       (@TYPE=EXE;))
                (Show    ("mg_2.txt")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Do      (TRUE)            (Run_Xgraph))
                (Reset   (T12))
        )
)

(@SLOT=         T13
        @COMMENTS="Special slot for creating the Phase 3 data set.  Its routine is similar to slots T10,
T11_A, and T12 executed in sequence from CrvDev_Ctl_R rule.";
        (@CACTIONS=
                (Do      (0)       (Model_Count))
                (Do      (TRUE)            (Start_Progress))
                (Do      (TRUE)            (Phs3.Plot))
                (Do      (Phs3.FN_Data)  (dummy_file))
                (Do      (STRCAT("\\"",Phs3.Title))         (dummy_title))
                (Execute            ("echo @V(dummy_title) > @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute            ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do      (TRUE)            (Run_Xgraph))
                (Reset   (T13))
        )
```

```
)


(@SLOT=      T15
        @COMMENTS="Special slot for readying the Phase 5 data set, created from OLD_NEW in
CrvMod module.  Its routine is similar to slots T10, T11_A, and T12 executed in sequence from
CrvDev_Ctl_R rule.";
        (@CACTIONS=
                (Do      (0)      (Model_Count))
                (Do      (TRUE)         (Start_Progress))
                (Do      (TRUE)         (\DS_Name\Plot))
                (Reset   (T15))
        )
)


(@SLOT=      T20
        (@CACTIONS=
                (Do      (STRCAT(ML_Name,".*"))       (dummy_string))
                (Execute         ("rm @V(dummy_string)")        (@TYPE=EXE;))
                (Show    ("Txt/inpexp.txt")
(@KEEP=FALSE;@WAIT=FALSE;@RECT=5,390,500,\
350;))
                (Reset   (Input_Exp))
                (Do      (Chk_Exponent)  (Chk_Exponent))
                (Reset   (Chk_Exponent))
                (Do      (Input_Exp)      (\ML_Name\TX))
                (Show    ("msg_4.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
                (Do      (\DS_Name\FN_Data)    (\ML_Name\FN_Data))
                (Do      (STRCAT(ML_Name,STRCAT("_",FLOAT2STR(Input_Exp))))
(\ML_Name\Title))
                (Do      (\ML_Name\FN_Data)    (dummy_infile))
                (Do      (STRCAT(ML_Name,".txt"))       (\ML_Name\FN_Results))
                (Do      (STRCAT(ML_Name,".ol"))        (\ML_Name\FN_Outliers))
                (Do      (STRCAT(\ML_Name\FN_Data,STRCAT(" ",STRCAT(ML_Name,\
STRCAT(" ",STRCAT(FLOAT2STR(\ML_Name\TX),\
STRCAT(" ",FLOAT2STR(Allow_err)))))))         (dummy_params))
                (Execute         ("LRM2 @V(dummy_params)")   (@TYPE=EXE;))
                (Do      (TRUE)         (\ML_Name\Plot))
                (Reset   (T20))
        )
)

(@SLOT=      T21
        (@CACTIONS=
                (Do      (0)      (Cursor2))
                (Do      (STRCAT(ML_Name,".NXP"))    (dummy_outfile))
                (Retrieve        ("LR2_a.NXP")
(@TYPE=NXPDB;@SLOTS=\ML_Name\B0,\ML_Name\B1,\
\ML_Name\R2,\ML_Name\StdEE,\ML_Name\Num_OL;\
@FIELDS="b0","b1","corr","stderest","numol";\
@CURSOR=Cursor2;))
                (Reset   (T21))
        )
)

(@SLOT=         T22
        @COMMENTS="Adds new model to PROGRESS LISTING and Displays LISTING";
        (@CACTIONS=
                (Do      (Alter_Progress)  (Alter_Progress))
                (Reset   (Alter_Progress))
                (Show    ("mmsum.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset   (T22))
        )
)

(@SLOT=         T23
        @COMMENTS="Generate Curve Points for Plotting, based on Model Parameters Generated.";
        (@CACTIONS=
                (Do      (STRCAT(FLOAT2STR(\ML_Name\.B0),STRCAT("     ",
STRCAT(FLOAT2STR(\ML_Name\B1)," "))))      (dummy_params))
                (Do      (STRCAT(dummy_params,STRCAT(FLOAT2STR(\ML_Name\TX),STRCAT("
",STRCAT(FLOAT2STR(\DS_Name\Max_S), STRCAT(" ",ML_Name))))))   (dummy_params))
                (Execute            ("GC @V(dummy_params)")       (@TYPE=EXE;))
                (Do      (STRCAT(ML_Name,".mdl"))       (\ML_Name\FN_Curve))
                (Do      (STRCAT(ML_Name,".cl"))        (\ML_Name\FN_Conlim))
                (Reset   (T23))
        )
)

(@SLOT=         T24
        @COMMENTS="Display Current Model, Dataset and Outliers";
        (@CACTIONS=
                (Do      (TRUE)          (XG_put_data))
                (Do      (TRUE)          (XG_put_curve))
                (Do      (TRUE)          (XG_put_outliers))
                (Do      (TRUE)          (Run_Xgraph))
                (Reset   (T24))
        )
)



(@SLOT=         T3
        @COMMENTS="Prepare Query String Specification";
        (@CACTIONS=
                (Reset   (Set_Qry_Suffix))
                (Do      (Set_Qry_Suffix)           (Set_Qry_Suffix))
                (Do      (STRCAT(Query_Prefix,Query_Suffix))    (Query_String))
        )
)
```

```
(@SLOT=        T40
        (@CACTIONS=
                (Do      (STRCAT(" Model ... ",ML_Name))      (dummy_string))
                (Execute         ("echo @V(dummy_string) >> mmsum.txt")         (@TYPE=EXE;))
                (Execute         ("echo    Selected as BEST CURVE by @V(Best_Model_Crit) >>
mmsum.txt")     (@TYPE=EXE;))
                (Show    ("mmsum.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset   (T40))
        )
)

(@SLOT=        T41
        (@CACTIONS=
                (Show    ("msg_3.txt")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Do      (Start_day)        (Stored.F_Day))
                (Do      (Stop_day)         (Stored.L_Day))
                (Do      (BEST_Model.B0)         (Stored.B0))
                (Do      (BEST_Model.B1)         (Stored.B1))
                (Do      (BEST_Model.TX)         (Stored.TX))
                (Do      (\DS_Name\Max_D)        (Stored.Max_D))
                (Do      (\DS_Name\Max_S)        (Stored.Max_S))
                (Do      (\DS_Name\Min_D)        (Stored.Min_D))
                (Do      (\DS_Name\Min_S)        (Stored.Min_S))
                (Do      (\DS_Name\Last_Msr)     (Stored.Last_Msr))
                (Do      (\DS_Name\No_Flow_Stg)          (Stored.No_Flow_Stg))
                (Do      (BEST_Model.Crv_Type)           (Stored.Crv_Type))
                (Do      (BEST_Model.Crv_Id)     (Stored.Crv_Id))
                (Do      (NOW())            (Stored.Date_Made))
                (Write   ("@V(db_access)")
(@TYPE=ORACLE;@FILL=ADD;@END="commit";@NAME="!crv_id!";
@PROPS=Crv_Id,Crv_Type,B0,B1,Max_D,Max_S,No_Flow_Stg,
F_Day,L_Day,Min_D,Min_S,Date_Made,TX,Last_Msr;
@FIELDS="CRV_ID","TYPE","B0","B1","MAX_D",
"MAX_S","NO_FLOW_STG","F_DAY","L_DAY","MIN_D","MIN_S","DATEMADE",
"TX","LAST_MSR";@QUERY="@V(C_Stn.Crv_Tbl)";
@ATOMS=Stored;))
                (Reset   (T41))
        )
)

(@SLOT=        T42
        @COMMENTS="This Slot performs an Oracle SQL DELETE of the current curve from the
CRV_TBL table, to be replaced by the new updated curve (see RULE: Phase5_Store_Curve).";
        (@CACTIONS=
                (Retrieve        ("db_access")    (@TYPE=ORACLE;@BEGIN="DELETE    FROM
@V(C_Stn.Crv_Tbl) WHERE CRV_ID = @V(C_Crv.Crv_Id)";@END="COMMIT;RELEASE";))
                (Reset   (T42))
        )
)

(@SLOT=        T8
        (@CACTIONS=
                (Reset   (Do_Next_Task))
                (Do      (Do_Next_Task) (Do_Next_Task))
        )
)

(@SLOT=         tweetie
        @INFCAT=500;
)

(@SLOT=         XG_put_curve
        (@CACTIONS=
                (Do      (\ML_Name\FN_Curve)  (dummy_file))
                (Do      (STRCAT("\\"",\ML_Name\Title))          (dummy_title))
                (Execute            ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute            ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do      ("")       (dummy_string))
                (Execute            ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Reset   (XG_put_curve))
        )
)

(@SLOT=         XG_put_data
        (@CACTIONS=
                (Do      (\DS_Name\FN_Data)     (dummy_file))
                (Do      (STRCAT("\\"",\DS_Name\Title)) (dummy_title))
                (Execute            ("echo @V(dummy_title) > @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute            ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do      ("")       (dummy_string))
                (Execute            ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Reset   (XG_put_data))
        )
)

(@SLOT=         XG_put_outliers
        (@CACTIONS=
                (Do      (\ML_Name\FN_Outliers)          (dummy_file))
                (Do      ("\\"Outliers")    (dummy_title))
                (Execute            ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute            ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do      ("")       (dummy_string))
                (Execute            ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")     (@TYPE=EXE;))
                (Reset   (XG_put_outliers))
        )
)

(@RULE=     R15
        @COMMENTS="Adds each model successively to the MODEL SUMMARY LIST";
        (@LHS=
                (>=     (Model_Count-Counter)   (0))
        )
        (@HYPO=         Alter_Progress)
        (@RHS=
                (Do     (STRCAT("ML",INT2STR(Counter)))       (ML_Name))
                (Execute        ("echo @V(ML_Name) > mmsum.in")       (@TYPE=EXE;))
                (Do     (\ML_Name\TX)           (dummy_float))
                (Execute        ("echo @V(dummy_float) >> mmsum.in")  (@TYPE=EXE;))
                (Do     (\ML_Name\Num_OL)   (dummy_integer))
                (Execute        ("echo @V(dummy_integer) >> mmsum.in")        (@TYPE=EXE;))
                (Do     (\ML_Name\StdEE)        (dummy_float))
                (Execute        ("echo @V(dummy_float) >> mmsum.in")  (@TYPE=EXE;))
                (Do     (\ML_Name\R2)           (dummy_float))
                (Execute        ("echo @V(dummy_float) >> mmsum.in")  (@TYPE=EXE;))
                (Execute        ("mmsum")       (@TYPE=EXE;))
                (Execute        ("cat mmsum.out >> mmsum.mid")        (@TYPE=EXE;))
                (Do     (Counter+1)     (Counter))
                (Reset  (Alter_Progress))
        )
)

(@RULE=     R14
        @COMMENTS="Adds model to MODEL SUMMARY LIST";
        (@LHS=
                (Is     (CrvDev_Call)   ("CrvDev_Ctl_M"))
        )
        (@HYPO=         Alter_Progress)
        (@RHS=
                (Execute        ("echo @V(ML_Name) > mmsum.in")       (@TYPE=EXE;))
                (Do     (\ML_Name\TX)           (dummy_float))
                (Execute        ("echo @V(dummy_float) >> mmsum.in")  (@TYPE=EXE;))
                (Do     (\ML_Name\Num_OL)   (dummy_integer))
                (Execute        ("echo @V(dummy_integer) >> mmsum.in")        (@TYPE=EXE;))
                (Do     (\ML_Name\StdEE)        (dummy_float))
                (Execute        ("echo @V(dummy_float) >> mmsum.in")  (@TYPE=EXE;))
                (Do     (\ML_Name\R2)           (dummy_float))
                (Execute        ("echo @V(dummy_float) >> mmsum.in")  (@TYPE=EXE;))
                (Execute        ("mmsum")       (@TYPE=EXE;))
                (Execute        ("cat mmsum.out >> mmsum.txt")        (@TYPE=EXE;))
        )
)

(@RULE=     R17
        (@LHS=
                (<      (Input_Exp)     (1.5))
```

```
          )
          (@HYPO=       Chk_Exponent)
          (@RHS=
                    (Execute          ("message")       (@STRING="@TEXT=A transformation exponent less
than 1.5 is not permitted!  Pres\
s OK and re-entered a value between 1.5 and 3.5 ...,\
@OK";))
                    (Reset   (Input_Exp))
                    (Reset   (Chk_Exponent))
          )
)

(@RULE=       R16
          (@LHS=
                    (>        (Input_Exp)       (3.5))
          )
          (@HYPO=       Chk_Exponent)
          (@RHS=
                    (Execute          ("message")       (@STRING="@TEXT=A  transformation  exponent
greater than 3.5 is not permitted!  P\
ress OK and re-entered a value between 1.5 and 3.5 ...,\
@OK";))
                    (Reset   (Input_Exp))
                    (Reset   (Chk_Exponent))
          )
)

(@RULE=       CrvDev_System_Control
          @COMMENTS="This rule transfers returns control back to the main system.";
          (@LHS=
                    (Is       (Lock_CrvDev)  ("OFF"))
          )
          (@HYPO=       CrvDev_Control)
          (@RHS=
                    (Execute          ("rm -f ML?.ol")          (@TYPE=EXE;))
                    (Execute          ("rm -f ML?.mdl")         (@TYPE=EXE;))
                    (Execute          ("rm -f ML?.txt")         (@TYPE=EXE;))
                    (Execute          ("rm -f ML?.cl") (@TYPE=EXE;))
          )
)

(@RULE=       CrvDev_System_Control
          @COMMENTS="This rule controls the next control call.";
          (@LHS=
                    (Is       (Lock_CrvDev)  ("ON"))
          )
          (@HYPO=       CrvDev_Control)
          (@RHS=
                    (Do      (\CrvDev_Call\.Value)     (Control_Return))
                    (Do      (TRUE)            (CrvDev_Ctl_Resets))
                    (Reset   (CrvDev_Control))
```

```
        )
)

(@RULE=      Phase5_Start_Modelling
        (@LHS=
                (=      (Phase)  (5))
        )
        (@HYPO=         CrvDev_Ctl_0)
        (@RHS=
                (Do     ("Phs5")          (DS_Name))
                (Do     (TRUE)            (Reset_Datasets))
                (Do     (TRUE)            (T15))
                (Do     ("CrvDev_Ctl_1")          (CrvDev_Call))
        )
)

(@RULE=      Phase3_Start_Modelling
        (@LHS=
                (=      (Phase)  (3))
        )
        (@HYPO=         CrvDev_Ctl_0)
        (@RHS=
                (Do     ("Phs3")          (DS_Name))
                (Do     (TRUE)            (Reset_Datasets))
                (Do     (TRUE)            (T13))
                (Do     ("CrvDev_Ctl_1")          (CrvDev_Call))
        )
)

(@RULE=      Phase2_Set_Qry_Suffix
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_0.Opt_p2)  ("Select"))
                (Yes    (Set_Qry_Suffix))
        )
        (@HYPO=         CrvDev_Ctl_0)
        (@RHS=
                (Do     (STRCAT(" WHERE ",Qry_Str_Suffix))    (Qry_Str_Suffix))
                (Do     (STRCAT(Qry_Str_Prefix,Qry_Str_Suffix))          (Query_String))
                (Do     ("Phs2")          (DS_Name))
                (Do     (TRUE)            (Reset_Datasets))
                (Do     ("CrvDev_Ctl_R")          (CrvDev_Call))
        )
)

(@RULE=      Phase2_Exit_System
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_0.Opt_p2)  ("Exit"))
        )
        (@HYPO=         CrvDev_Ctl_0)
        (@RHS=
                (Do     ("EndSession")  (System_Call))
                (Do     ("OFF")         (Lock_CrvDev))
        )
)

(@RULE=         Phase1_Start_Modelling
        (@LHS=
                (=      (Phase)  (1))
                (Is     (Ctrl_0.Opt_p1)  ("All S-D Measurements"))
        )
        (@HYPO=         CrvDev_Ctl_0)
        (@RHS=
                (Do     (Qry_Str_Prefix) (Query_String))
                (Do     ("ALL")         (Phase.Type))
                (Do     ("Phs1")        (DS_Name))
                (Do     (TRUE)          (Reset_Datasets))
                (Do     ("CrvDev_Ctl_R")        (CrvDev_Call))
        )
)

(@RULE=         Phase1_Set_Qry_Suffix
        (@LHS=
                (=      (Phase)  (1))
                (Is     (Ctrl_0.Opt_p1)  ("Select Specific Measurements"))
        )
        (@HYPO=         CrvDev_Ctl_0)
        (@RHS=
                (Do     (TRUE)          (ctrl_inits))
                (Do     (2)     (Phase))
                (Do     ("Select")      (Ctrl_0.Opt_p2))
                (Do     ("CrvDev_Ctl_0")        (CrvDev_Call))
        )
)

(@RULE=         Phase5_Begin_Modelling
        (@LHS=
                (=      (Phase)  (5))
        )
        (@HYPO=         CrvDev_Ctl_1)
        (@RHS=
                (Do     (Model_Count+1)         (Model_Count))
                (Do     (STRCAT("ML",INT2STR(Model_Count)))         (ML_Name))
                (Do     (TRUE)          (ML_Create))
                (Do     ("CrvDev_Ctl_M")        (CrvDev_Call))
        )
)

(@RULE=         Phase3_Begin_Modelling
        (@LHS=
                (=      (Phase)  (3))
        )
        (@HYPO=     CrvDev_Ctl_1)
        (@RHS=
                (Do     (Model_Count+1)         (Model_Count))
                (Do     (STRCAT("ML",INT2STR(Model_Count)))         (ML_Name))
                (Do     (TRUE)          (ML_Create))
                (Do     ("CrvDev_Ctl_M")        (CrvDev_Call))
        )
)

(@RULE=     Phase2_Restart_Phase_2
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_1.Opt_p2)  ("Redefine measurement selection"))
        )
        (@HYPO=     CrvDev_Ctl_1)
        (@RHS=
                (Do     (TRUE)          (DS_Remove))
                (Do     ("CrvDev_Ctl_SW2")      (CrvDev_Call))
        )
)

(@RULE=     Phase2_Exit_System
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_1.Opt_p2)  ("Exit Curve Dev."))
        )
        (@HYPO=     CrvDev_Ctl_1)
        (@RHS=
                (Do     ("EndSession")  (System_Call))
                (Do     ("OFF")         (Lock_CrvDev))
        )
)

(@RULE=     Phase2_Begin_Modelling
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_1.Opt_p2)  ("Model Data"))
        )
        (@HYPO=     CrvDev_Ctl_1)
        (@RHS=
                (Do     (Model_Count+1)         (Model_Count))
                (Do     (STRCAT("ML",INT2STR(Model_Count)))         (ML_Name))
                (Do     (TRUE)          (ML_Create))
                (Do     ("CrvDev_Ctl_M")        (CrvDev_Call))
        )
)

(@RULE=     Phase1_Goto_Phase_2
        (@LHS=
                (=      (Phase)  (1))
                (Is       (Ctrl_1.Opt_p1)  ("Select Specific Measurments"))
        )
        (@HYPO=           CrvDev_Ctl_1)
        (@RHS=
                (Do       (TRUE)           (DS_Remove))
                (Do       ("CrvDev_Ctl_SW2")      (CrvDev_Call))
        )
)

(@RULE=         Phase1_Exit_System
        (@LHS=
                (=        (Phase)  (1))
                (Is       (Ctrl_1.Opt_p1)  ("Exit Curve Dev"))
        )
        (@HYPO=           CrvDev_Ctl_1)
        (@RHS=
                (Do       ("EndSession")   (System_Call))
                (Do       ("OFF")          (Lock_CrvDev))
        )
)

(@RULE=         Phase1_Begin_Modelling
        (@LHS=
                (=        (Phase)  (1))
                (Is       (Ctrl_1.Opt_p1)  ("Model Data"))
        )
        (@HYPO=           CrvDev_Ctl_1)
        (@RHS=
                (Do       (Model_Count+1)         (Model_Count))
                (Do       (STRCAT("ML",INT2STR(Model_Count)))           (ML_Name))
                (Do       (TRUE)           (ML_Create))
                (Do       ("CrvDev_Ctl_M")        (CrvDev_Call))
        )
)

(@RULE=         Phase5_Evaluate_Model
        (@LHS=
                (=        (Phase)  (5))
                (Is       (Ctrl_2.Opt_p5)  ("Evaluate Models"))
        )
        (@HYPO=           CrvDev_Ctl_2)
        (@RHS=
                (Do       (TRUE)           (Prepare_For_Evaluation))
                (Do       ("CrvDev_Ctl_SBM")      (CrvDev_Call))
        )
)

(@RULE=         Phase5_Another_Model
        (@LHS=
                (=        (Phase)  (5))
                (Is       (Ctrl_2.Opt_p5)  ("Generate Another Model"))
        )
        (@HYPO=        CrvDev_Ctl_2)
        (@RHS=
                (Reset  (Ctrl_2.Opt_p5))
                (Do     ("CrvDev_Ctl_1")        (CrvDev_Call))
        )
)

(@RULE=         Phase3_Evaluate_Model
        (@LHS=
                (=      (Phase)  (3))
                (Is     (Ctrl_2.Opt_p3)  ("Evaluate Models"))
        )
        (@HYPO=        CrvDev_Ctl_2)
        (@RHS=
                (Do     (TRUE)          (Prepare_For_Evaluation))
                (Do     ("CrvDev_Ctl_SBM")      (CrvDev_Call))
        )
)

(@RULE=         Phase3_Another_Model
        (@LHS=
                (=      (Phase)  (3))
                (Is     (Ctrl_2.Opt_p3)  ("Generate Another Model"))
        )
        (@HYPO=        CrvDev_Ctl_2)
        (@RHS=
                (Reset  (Ctrl_2.Opt_p3))
                (Do     ("CrvDev_Ctl_1")        (CrvDev_Call))
        )
)

(@RULE=         Phase2_Evaluate_Model
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_2.Opt_p2)  ("Evaluate Models"))
        )
        (@HYPO=        CrvDev_Ctl_2)
        (@RHS=
                (Do     (TRUE)          (Prepare_For_Evaluation))
                (Do     ("CrvDev_Ctl_SBM")      (CrvDev_Call))
        )
)

(@RULE=         Phase2_Another_Model
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_2.Opt_p2)  ("Generate Another Model"))
        )
        (@HYPO=        CrvDev_Ctl_2)
        (@RHS=
                (Reset   (Ctrl_2.Opt_p2))
                (Do      ("CrvDev_Ctl_1")          (CrvDev_Call))
        )
)

(@RULE=         Phase1_Evaluate_Model
        (@LHS=
                (=       (Phase)  (1))
                (Is      (Ctrl_2.Opt_p1)  ("Evaluate Models"))
        )
        (@HYPO=          CrvDev_Ctl_2)
        (@RHS=
                (Do      (TRUE)            (Prepare_For_Evaluation))
                (Do      ("CrvDev_Ctl_SBM")      (CrvDev_Call))
        )
)

(@RULE=         Phase1_Another_Model
        (@LHS=
                (=       (Phase)  (1))
                (Is      (Ctrl_2.Opt_p1)  ("Generate Another Model"))
        )
        (@HYPO=          CrvDev_Ctl_2)
        (@RHS=
                (Reset   (Ctrl_2.Opt_p1))
                (Do      ("CrvDev_Ctl_1")          (CrvDev_Call))
        )
)

(@RULE=         Phase5_Store_Curve
        (@LHS=
                (=       (Phase)  (5))
        )
        (@HYPO=          CrvDev_Ctl_3)
        (@RHS=
                (Do      (TRUE)            (T42))
                (Do      (TRUE)            (T41))
                (Do      (TRUE)            (DS_Remove))
                (Do      (TRUE)            (ML_Remove_All))
                (Do      ("EndSession")    (System_Call))
                (Do      ("OFF")           (Lock_CrvDev))
        )
)

(@RULE=         Phase3_Store_Curve
        (@LHS=
                (=       (Phase)  (3))
                (Is      (Ctrl_3.Opt_p3)  ("Store Model"))
        )
        (@HYPO=          CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)          (T41))
                (Do     (TRUE)          (DS_Remove))
                (Do     (TRUE)          (ML_Remove_All))
                (Do     ("EndSession")  (System_Call))
                (Do     ("OFF")         (Lock_CrvDev))
        )
)

(@RULE=         Phase3_Exit_Abort_System
        @COMMENTS="This rule aborts the Phase 3 modelling procedure and returns to the main
system.";
        (@LHS=
                (=      (Phase)  (3))
                (Is     (Ctrl_3.Opt_p3)  ("Abort Modelling"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)          (DS_Remove))
                (Do     (TRUE)          (ML_Remove_All))
                (Do     ("EndSession")  (System_Call))
                (Do     ("OFF")         (Lock_CrvDev))
        )
)

(@RULE=         Phase2_Store_Curve
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_3.Opt_p2)  ("Store Model"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)          (T41))
                (Do     (TRUE)          (DS_Remove))
                (Do     (TRUE)          (ML_Remove_All))
                (Do     ("EndSession")  (System_Call))
                (Do     ("OFF")         (Lock_CrvDev))
        )
)

(@RULE=         Phase2_Repeat_Phase_2
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_3.Opt_p2)  ("Select a Different Dataset"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)          (DS_Remove))
                (Do     (TRUE)          (ML_Remove_All))
                (Do     ("CrvDev_Ctl_SW2")      (CrvDev_Call))
        )
)
```

```
(@RULE=        Phase2_Goto_Phase4
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_3.Opt_p2)  ("Segment Curve  *"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)            (DS_Remove))
                (Do     (TRUE)            (ML_Remove_All))
                (Do     ("EndSession")    (System_Call))
                (Do     ("OFF")           (Lock_CrvDev))
        )
)

(@RULE=        Phase2_Goto_Phase3
        (@LHS=
                (=      (Phase)  (2))
                (Is     (Ctrl_3.Opt_p2)  ("Identify Outliers "))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     ("CrvDev_Ctl_OL_Load")            (CrvDev_Call))
        )
)

(@RULE=        Phase1_Store_Curve
        (@LHS=
                (=      (Phase)  (1))
                (Is     (Ctrl_3.Opt_p1)  ("Store Model"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)            (T41))
                (Do     (TRUE)            (DS_Remove))
                (Do     (TRUE)            (ML_Remove_All))
                (Do     ("EndSession")    (System_Call))
                (Do     ("OFF")           (Lock_CrvDev))
        )
)

(@RULE=        Phase1_Goto_Phase_2
        (@LHS=
                (=      (Phase)  (1))
                (Is     (Ctrl_3.Opt_p1)  ("Re-select Specific Measurements"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)            (DS_Remove))
                (Do     (TRUE)            (ML_Remove_All))
                (Do     ("CrvDev_Ctl_SW2")    (CrvDev_Call))
        )
```

```
)

(@RULE=        Phase1_Goto_Phase4
        (@LHS=
                (=      (Phase)  (1))
                (Is     (Ctrl_3.Opt_p1)  ("Segment Curve  *"))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     (TRUE)           (DS_Remove))
                (Do     (TRUE)           (ML_Remove_All))
                (Do     ("EndSession")   (System_Call))
                (Do     ("OFF")          (Lock_CrvDev))
        )
)

(@RULE=        Phase1_Goto_Phase3
        (@LHS=
                (=      (Phase)  (1))
                (Is     (Ctrl_3.Opt_p1)  ("Identify Outliers "))
        )
        (@HYPO=         CrvDev_Ctl_3)
        (@RHS=
                (Do     ("CrvDev_Ctl_OL_Load")          (CrvDev_Call))
        )
)

(@RULE=        Execute_Modeling_Tasks
        (@LHS=
                (=      (1)      (1))
        )
        (@HYPO=         CrvDev_Ctl_M)
        (@RHS=
                (Do     (TRUE)           (T20))
                (Do     (TRUE)           (T21))
                (Do     (TRUE)           (T22))
                (Do     (TRUE)           (T23))
                (Do     (TRUE)           (T24))
                (Do     ("CrvDev_Ctl_2")          (CrvDev_Call))
        )
)

(@RULE=        R57
        (@LHS=
                (=      (1)      (1))
        )
        (@HYPO=         CrvDev_Ctl_OL_Load)
        (@RHS=
                (Do     ("OutlierAnalysis_Start")  (System_Call))
                (Do     ("OFF")          (Lock_CrvDev))
                (Show   ("msg_30.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
        )
)

(@RULE=         Execute_Data_Selection_Tasks
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvDev_Ctl_R)
        (@RHS=
                (Do     (TRUE)          (T10))
                (Do     (TRUE)          (T11_A))
                (Do     (TRUE)          (T12))
                (Do     ("CrvDev_Ctl_1")        (CrvDev_Call))
        )
)

(@RULE=         Select_model_with_MIN_STDEE
        @COMMENTS="Select Best Model using SSE and copy to BEST (model) object.";
        (@LHS=
                (Is     (Best_Model_Crit)       ("Standard Error of Estimate"))
                (Name   (MIN(<|Models|>.StdEE))         (Best_StdEE))
                (=      (Best_StdEE-<|Models|>.StdEE)   (0))
        )
        (@HYPO=         CrvDev_Ctl_SBM)
        (@RHS=
                (Reset  (ML_Name))
                (Execute        ("AtomNameValue")
(@ATOMID=<|Models|>;@STRING="@RETURN=ML_Name.value,\
@NAMES";))
                (Execute        ("CopyFrame")
(@ATOMID=<|Models|>,BEST_Model;@STRING="@STRAT=SET";\
))
                (Do     (TRUE)          (Best_Xtras))
                (Do     (TRUE)          (BEST_Model.Plot))
                (Do     (TRUE)          (T40))
                (Do     ("CrvDev_Ctl_3")        (CrvDev_Call))
        )
)

(@RULE=         Select_model_with_MIN_OUTLIERS
        @COMMENTS="Select Best Model using OUTLIERS and copy to BEST (model) object.";
        (@LHS=
                (Is     (Best_Model_Crit)       ("Minimum Number of Outliers"))
                (Name   (MIN(<|Models|>.Num_OL))        (Best_NOL))
                (=      (Best_NOL-<|Models|>.Num_OL)  (0))
        )
        (@HYPO=         CrvDev_Ctl_SBM)
        (@RHS=
                (Reset  (ML_Name))
                (Execute        ("AtomNameValue")
(@ATOMID=<|Models|>;@STRING="@RETURN=ML_Name.Value,\
@NAMES";))
                    (Execute          ("CopyFrame")
(@ATOMID=<|Models|>,BEST_Model;@STRING="@STRAT=SET";\
))
                    (Do      (TRUE)           (Best_Xtras))
                    (Do      (TRUE)           (BEST_Model.Plot))
                    (Do      (TRUE)           (T40))
                    (Do      ("CrvDev_Ctl_3")          (CrvDev_Call))
          )
)

(@RULE=        Select_model_with_Max_R2
          @COMMENTS="Select Best Model using OUTLIERS and copy to BEST (model) object.";
          (@LHS=
                    (Is      (Best_Model_Crit)        ("Maximum R2 value"))
                    (Name    (MIN(<|Models|>.R2))     (Best_R2))
                    (=       (Best_R2-<|Models|>.R2) (0))
          )
          (@HYPO=        CrvDev_Ctl_SBM)
          (@RHS=
                    (Reset   (ML_Name))
                    (Execute          ("AtomNameValue")
(@ATOMID=<|Models|>;@STRING="@RETURN=ML_Name.value,\
@NAMES";))
                    (Execute          ("CopyFrame")
(@ATOMID=<|Models|>,BEST_Model;@STRING="@STRAT=SET";\
))
                    (Do      (TRUE)           (Best_Xtras))
                    (Do      (TRUE)           (BEST_Model.Plot))
                    (Do      (TRUE)           (T40))
                    (Do      ("CrvDev_Ctl_3")          (CrvDev_Call))
          )
)

(@RULE=        Phase2_Repeat_Phase_2
          (@LHS=
                    (=       (Phase)  (2))
          )
          (@HYPO=        CrvDev_Ctl_SW2)
          (@RHS=
                    (Show    ("msg_24.txt")   (@KEEP=FALSE;@WAIT=FALSE;))
                    (Reset   (Phase.Type))
                    (Reset   (Query_String))
                    (Reset   (Set_Qry_Suffix))
                    (Do      (TRUE)           (Init_Ctrls))
                    (Do      ("Select")       (Ctrl_0.Opt_p2))
                    (Do      ("CrvDev_Ctl_0")          (CrvDev_Call))
          )
)

(@RULE=        Phase1_Change_to_Phase_2
        (@LHS=
                (=      (Phase)  (1))
        )
        (@HYPO=         CrvDev_Ctl_SW2)
        (@RHS=
                (Show   ("msg_24.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset  (Phase.Type))
                (Reset  (Set_Qry_Suffix))
                (Do     (TRUE)            (Init_Ctrls))
                (Do     ("Select")        (Ctrl_0.Opt_p2))
                (Do     (2)      (Phase))
                (Do     ("CrvDev_Ctl_0")          (CrvDev_Call))
        )
)

(@RULE=         R65
        (@LHS=
                (Is     (Another_model)           ("Yes"))
        )
        (@HYPO=         Gen_Models)
        (@RHS=
                (Do     (Model_Count+1)           (Model_Count))
                (Do     (TRUE)            (T20))
                (Do     (TRUE)            (T21))
                (Do     (TRUE)            (T22))
                (Do     (TRUE)            (T23))
                (Do     (TRUE)            (T24))
                (Reset  (Another_model))
                (Reset  (Gen_Models))
        )
)

(@RULE=         R64
        (@LHS=
                (Is     (Another_model)           ("No"))
        )
        (@HYPO=         Gen_Models)
)

(@RULE=         Get_ModelData
        (@LHS=
                (>=     (Cursor)          (0))
        )
        (@HYPO=         Get_Seq_Data)
        (@RHS=
                (Retrieve          ("@V(db_access)")
(@TYPE=ORACLE;@SLOTS=New_msr.Stage,New_msr.Discharge,\
New_msr.Msr_Id,New_msr.Msr_Date;@FIELDS="stage",\
"discharge","msr_id","date_of_msr";@QUERY="@V(Query_String)";\
@CURSOR=Cursor;))
                (Reset  (Misc_duties))
```

```
                    (Do       (Misc_duties)     (Misc_duties))
                    (Reset    (Get_Seq_Data))
          )
)

(@RULE=         Get_Zero_FLow_Stage_A
          @COMMENTS="Retreives the ZERO FLOW STAGE for the data retrieved according to
QUERY_STRING.";
          (@LHS=
                    (=        (Phase)  (1))
          )
          (@HYPO=         Get_Zero_Flow)
          (@RHS=
                    (Reset    (Cursor))
                    (Retrieve          ("@V(db_access)")
(@TYPE=ORACLE;@SLOTS=\DS_Name\No_Flow_Stg;
@FIELDS="MAX(STAGE)";@QUERY="@V(Query_String) WHERE DISCHARGE = 0";
@CURSOR=Cursor;@END="RELEASE";))
                    (Do       (0)       (Zero_count))
          )
)

(@RULE=         Get_Zero_FLow_Stage_B
          @COMMENTS="Retreives the ZERO FLOW STAGE for the data retrieved according to
QUERY_STRING.";
          (@LHS=
                    (>        (Phase)  (1))
          )
          (@HYPO=         Get_Zero_Flow)
          (@RHS=
                    (Reset    (Cursor))
                    (Retrieve          ("@V(db_access)")
(@TYPE=ORACLE;@SLOTS=\DS_Name\No_Flow_Stg;
@FIELDS="MAX(STAGE)";@QUERY="@V(Query_String) AND DISCHARGE = 0";
@CURSOR=Cursor;@END="RELEASE";))
                    (Do       (0)       (Zero_count))
          )
)



(@RULE=         R67
          (@LHS=
                    (Is       (<|Models|>.Plot)          (TRUE))
          )
          (@HYPO=         List_curve_A)
          (@RHS=
                    (Execute           ("AtomNameValue")
(@ATOMID=<|Models|>;@STRING="@RETURN=Plot_Info.Obj_names,\
@NAMES";))
                    (Execute           ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names;@STRING="@RETURN=Plot_Info.Num_obj,\
@LENGTH";))
                (Do     (1)       (Plot_Info.Index))
                (Do     (List_curve_B)  (List_curve_B))
                (Reset  (List_curve_B))
        )
)

(@RULE=         R68
        (@LHS=
                (>=     (Plot_Info.Num_obj-Plot_Info.Index)        (0))
        )
        (@HYPO=         List_curve_B)
        (@RHS=
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names,dummy_obj.Value;\
@STRING="@INDEX=@V(Plot_Info.Index)";))
                (Do     (\dummy_obj\.FN_Curve)          (dummy_file))
                (Do     (STRCAT("\\"",\dummy_obj\.Title))          (dummy_title))
                (Execute        ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute        ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     ("-1 -1")         (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename")
(@TYPE=EXE;))
                (Do     ("")      (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     (Plot_Info.Index+1)       (Plot_Info.Index))
                (Reset  (List_curve_B))
        )
)

(@RULE=         R69
        (@LHS=
                (Is     (<|Datasets|>.Plot)         (TRUE))
        )
        (@HYPO=         List_data_A)
        (@RHS=
                (Execute        ("AtomNameValue")
(@ATOMID=<|Datasets|>;@STRING="@RETURN=Plot_Info.Obj_names,\
@NAMES";))
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names;@STRING="@RETURN=Plot_Info.Num_obj,\
@LENGTH";))
                (Do     (1)       (Plot_Info.Index))
                (Execute        ("rm -f @V(Plot_Info.Filename)") (@TYPE=EXE;))
                (Reset  (List_data_B))
                (Do     (List_data_B)    (List_data_B))
        )
```

```
)

(@RULE=        R70
        (@LHS=
                (>=     (Plot_Info.Num_obj-Plot_Info.Index)         (0))
        )
        (@HYPO=         List_data_B)
        (@RHS=
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names,dummy_obj.Value; @STRING="@INDEX=@V(Plot_Info.Index)";))
                (Do     (\dummy_obj\.FN_Data)   (dummy_file))
                (Do     (STRCAT("\\"",\dummy_obj\.Title))          (dummy_title))
                (Execute        ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute        ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     ("-1 -1")           (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.filename)")
(@TYPE=EXE;))
                (Do     ("")    (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     (Plot_Info.Index+1)         (Plot_Info.Index))
                (Reset  (List_data_B))
        )
)

(@RULE=        R71
        (@LHS=
                (<=     (Phase) (2))
                (>=     (Phase) (1))
        )
        (@HYPO=         Misc_duties)
        (@RHS=
                (Do     (Print_Record)  (Print_Record))
                (Reset  (Print_Record))
        )
)

(@RULE=        R72
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         Next_CrvId)
        (@RHS=
                (Reset  (Cursor))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="release";@SLOTS=dummy_integer.Value;@FIELDS="max(crv_id)+1";@
QUERY="lr_lj047";@CURSOR=Cursor;))
                (Do     (dummy_integer)         (\ML_Name\.Crv_Id))
        )
```

```
)

(@RULE=      Plot_TRUE_Current_Phase_Data
        (@LHS=
                (<=      (count)  (5))
                (=       (Phase-count)     (0))
        )
        (@HYPO=      Plot_Settings)
        (@RHS=
                (Do      (STRCAT("Phs",INT2STR(count)))        (dummy_string))
                (Do      (TRUE)          (\dummy_string\Plot))
                (Do      (count+1)       (count))
                (Reset   (Plot_Settings))
        )
)

(@RULE=      Plot_FALSE_Non_Current_Phase_Data
        (@LHS=
                (<=      (count)  (5))
                (<>      (Phase-count)     (0))
        )
        (@HYPO=      Plot_Settings)
        (@RHS=
                (Do      (STRCAT("Phs",INT2STR(count)))        (dummy_string))
                (Do      (FALSE)         (\dummy_string\Plot))
                (Do      (count+1)       (count))
                (Reset   (Plot_Settings))
        )
)

(@RULE=      Print_S_D_to_Unix_A
        @INFCAT=20;
        @COMMENTS="Filter out measurements with zero flow stages less than NO_FLOW_STG.  Count
# of zero flow msrs and subtract from total number of msrs for dataset.";
        (@LHS=
                (>=      (Cursor)          (0))
                (=       (New_msr.Discharge)      (0.000))
                (NotEqual          (\DS_Name\No_Flow_Stg)         (New_msr.Stage))
        )
        (@HYPO=      Print_Record)
        (@RHS=
                (Do      (Zero_count+1)  (Zero_count))
                (Do      (\DS_Name\Num_Msrs-1)          (\DS_Name\Num_Msrs))
        )
)

(@RULE=      Print_S_D_to_Unix_B
        @INFCAT=5;
        @COMMENTS="Print S/D measurement to UNIX file";
        (@LHS=
		(>=	(Cursor)	(0))
		(<>	(New_msr.Discharge)	(0))
	)
	(@HYPO=	Print_Record)
	(@RHS=
		(Do	(STRCAT(DS_Name,".dat"))	(dummy_file))
		(Do	(STRCAT(FLOAT2STR(New_msr.Discharge,"u.000"),STRCAT("
",FLOAT2STR(New_msr.Stage,"u.000")))) (New_msr.Set))
		(Execute	("echo @V(New_msr.Set) >> @V(dummy_file)")	(@TYPE=EXE;))
	)
)


(@RULE=	Print_S_D_to_Unix_C
	@INFCAT=5;
	@COMMENTS="Print S/D measurement to UNIX file";
	(@LHS=
		(>=	(Cursor)	(0))
		(=	(New_msr.Discharge)	(0))
		(Equal	(\DS_Name\No_Flow_Stg)	(New_msr.Stage))
	)
	(@HYPO=	Print_Record)
	(@RHS=
		(Do	(STRCAT(DS_Name,".dat"))	(dummy_file))
		(Do	(STRCAT(FLOAT2STR(New_msr.Discharge,"u.000"),STRCAT("
",FLOAT2STR(New_msr.Stage,"u.000")))) (New_msr.Set))
		(Execute	("echo @V(New_msr.Set) >> @V(dummy_file)")	(@TYPE=EXE;))
	)
)


(@RULE=	R77
	@INFCAT=5;
	@COMMENTS="Remove each model from MODELS CLASS and all related system files.";
	(@LHS=
		(No	(Bool_Answer))
	)
	(@HYPO=	Seq_ML_Remove)
)

(@RULE=	R76
	@INFCAT=10;
	@COMMENTS="Remove each model from MODELS CLASS and all related system files.";
	(@LHS=
		(Name	(STRCAT("ML",INT2STR(Counter)))	(ML_Name))
		(Execute	("AtomExist")
(@STRING="@NAME=@V(ML_Name),@RETURN=Bool_Answer.Value";\
))
		(Yes	(Bool_Answer))
	)
	(@HYPO=	Seq_ML_Remove)
```

```
          (@RHS=
                    (Do       (TRUE)              (ML_Remove))
                    (Do       (Counter+1)         (Counter))
                    (Reset    (Seq_ML_Remove))
          )
)

(@RULE=             R78
          @INFCAT=100;
          (@LHS=
                    (Yes      (<|Datasets|>.Plot))
          )
          (@HYPO=             Set_Axis)
          (@RHS=
                    (Do       (MAX(<|Datasets|>.Max_S))        (Plot_Info.Max_S))
                    (Do       (MAX(<|Datasets|>.Max_D))        (Plot_Info.Max_D))
          )
)

(@RULE=             Query_Time_Range_Case4
          @COMMENTS="Same First and Last Dates";
          (@LHS=
                    (Is       (Phase.Type)        ("TIME"))
                    (=        (P2_Time.LM-P2_Time.FM)          (0))
                    (=        (P2_Time.LD-P2_Time.FD)          (0))
          )
          (@HYPO=             Set_Qry_Suffix)
          (@RHS=
                    (Reset    (Set_Qry_Suffix))
          )
)

(@RULE=             Query_Time_Range_Case3_B
          @COMMENTS="Same Month, Range crosses years";
          (@LHS=
                    (Is       (Phase.Type)        ("TIME"))
                    (=        (P2_Time.FM-P2_Time.LM)          (0))
                    (>        (P2_Time.FD-P2_Time.LD)          (0))
          )
          (@HYPO=             Set_Qry_Suffix)
          (@RHS=
                    (Do       (Cnst_time2)        (Qry_Str_Suffix))
          )
)

(@RULE=             Query_Time_Range_Case3_A
          @COMMENTS="Same Month, Range within single year";
          (@LHS=
                    (Is       (Phase.Type)        ("TIME"))
                    (=        (P2_Time.LM-P2_Time.FM)          (0))
                    (>        (P2_Time.LD-P2_Time.FD)          (0))
```

```
        )
        (@HYPO=       Set_Qry_Suffix)
        (@RHS=
                (Do      (Cnst_time1)     (Qry_Str_Suffix))
        )
)

(@RULE=       Query_Time_Range_Case2
        @COMMENTS="First Month is later in year than Last Month";
        (@LHS=
                (Is      (Phase.Type)     ("TIME"))
                (>       (P2_Time.FM-P2_Time.LM)        (0))
        )
        (@HYPO=       Set_Qry_Suffix)
        (@RHS=
                (Do      (Cnst_time2)     (Qry_Str_Suffix))
        )
)

(@RULE=       Query_Time_Range_Case1
        @INFCAT=5;
        @COMMENTS="First Month is earlier in year than Last Month";
        (@LHS=
                (Is      (Phase.Type)     ("TIME"))
                (Name    (TRUE)           (Input_TimeRange))
                (>       (P2_Time.LM-P2_Time.FM)        (0))
        )
        (@HYPO=       Set_Qry_Suffix)
        (@RHS=
                (Do      (Cnst_time1)     (Qry_Str_Suffix))
        )
)

(@RULE=       Query_Rapid_Rising_Stage_Conditions
        (@LHS=
                (Is      (Phase.Type)     ("STAGE VARIATION"))
                (Is      (P2_Stage)       ("RAPIDLY RISING STAGE"))
        )
        (@HYPO=       Set_Qry_Suffix)
        (@RHS=
                (Do      (STRCAT(Qry_Str_Prefix," and REMARK_A = '8'"))         (Query_String))
                (Do      ("STG_RR")       (Phase.Type))
        )
)

(@RULE=       Query_Open_Flow
        @COMMENTS="Query Open Flow Conditions";
        (@LHS=
                (Is      (Phase.Type)     ("OPEN FLOW"))
        )
        (@HYPO=       Set_Qry_Suffix)
```

```
          (@RHS=
                    (Do       ("OPN_FLW")    (\DS_Name\Type))
                    (Do       (\DS_Name\Type)         (dummy_string))
                    (Reset    (Cursor))
                    (Retrieve           ("@V(db_access)")
(@TYPE=ORACLE;@END="release";@SLOTS=Qry_Str_Suffix.Value;\
@FIELDS="qry_string";@QUERY="qry_types where  type = '@V(dummy_string)'";\
@CURSOR=Cursor;))
          )
)

(@RULE=         Query_Ice_Open_Flow
          @COMMENTS="Query Open Flow Conditions";
          (@LHS=
                    (Is       (Phase.Type)       ("ICE CONDITIONS"))
                    (Is       (P2_Ice)           ("COMPLETE COVER"))
          )
          (@HYPO=        Set_Qry_Suffix)
          (@RHS=
                    (Do       (STRCAT(Query_String," and NOT( REMARK_B IN ('9',\
 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'J' ) "))   (Query_String))
          )
)

(@RULE=         R89
          (@LHS=
                    (=        (Level_1)          (3))
          )
          (@HYPO=        Show_Phase_Title)
          (@RHS=
                    (Show     ("Txt/P3t.txt")    (@KEEP=FALSE;@WAIT=TRUE;@RECT=5,380,527,\
300;))
          )
)

(@RULE=         R88
          (@LHS=
                    (=        (Level_1)          (2))
          )
          (@HYPO=        Show_Phase_Title)
          (@RHS=
                    (Show     ("Txt/P2t.txt")    (@KEEP=FALSE;@WAIT=TRUE;@RECT=5,380,527,\
300;))
          )
)

(@RULE=         R87
          (@LHS=
                    (=        (Level_1)          (1))
          )
          (@HYPO=        Show_Phase_Title)
          (@RHS=
                    (Show   ("Txt/P1t.txt")    (@KEEP=FALSE;@WAIT=TRUE;@RECT=5,380,527,\
300;))
          )
)


#*****************************************************************
#*  End of KB Code:  A3_main.tkb                                  *
#*****************************************************************
```

```
#****************************************************************
#*  Nexpert Object KB Code:  A4_main.tkb                         *
#****************************************************************
#*  Outlier Analysis Module                                      *
#****************************************************************


(@VERSION=    020)
(@PROPERTY=          BV       @TYPE=Boolean;)
(@PROPERTY=          CO       @TYPE=Boolean;)
(@PROPERTY=          DB       @TYPE=Boolean;)
(@PROPERTY=          Id       @TYPE=String;)
(@PROPERTY=          MU       @TYPE=Boolean;)
(@PROPERTY=          OK       @TYPE=Boolean;)
(@PROPERTY=          Rem_A    @TYPE=Integer;)
(@PROPERTY=          Rem_B    @TYPE=Integer;)
(@PROPERTY=          Rem_C    @TYPE=Integer;)
(@PROPERTY=          RF       @TYPE=Boolean;)
(@PROPERTY=          RR       @TYPE=Boolean;)
(@PROPERTY=          TR       @TYPE=Boolean;)
(@PROPERTY=          Trans_Exp          @TYPE=Float;)
(@PROPERTY=          Type_Set           @TYPE=String;)
(@PROPERTY=          UX       @TYPE=Boolean;)
(@PROPERTY=          WE       @TYPE=Boolean;)
(@PROPERTY=          WN       @TYPE=Boolean;)


(@OBJECT=      Attrib_Count
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=      Beaver
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=      Begin_Analysis
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=      Calc_limits
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
```

```
        )
)

(@OBJECT=     Cat_outlier
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Check_Outlier
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)


(@OBJECT=     Current_DS
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=     Ctrl_1_Next
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=     Current_MD
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=     Cursor1
        (@PROPERTIES=
                Value   @TYPE=Integer;
        )
)

(@OBJECT=     Cutoff
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=     Debris
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)
```

```
(@OBJECT=     DS_name
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     DS_num
          (@PROPERTIES=
                    Value    @TYPE=Integer;
          )
)

(@OBJECT=     dummy_file
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     dummy_obj
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     dummy_title
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Falling
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     FN_Data
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     List_curve_A
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     List_curve_B
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     List_data_A
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     List_data_B
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Make_OL_DS
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     ML_Name
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Model_response
          (@PROPERTIES=
                    Value    @TYPE=Float;
          )
)

(@OBJECT=     Msr_Attribs
          (@PROPERTIES=
                    BV
                    CO
                    DB
                    MU
                    OK
                    RF
                    RR
                    TR
                    UX
                    WE
                    WN
          )
)

(@OBJECT=     neg_limit
          (@PROPERTIES=
		Value	@TYPE=Float;
	)
)

(@OBJECT=	Out_side
	(@PROPERTIES=
		Value	@TYPE=String;
	)
)

(@OBJECT=	OutAnal_Ctl_0
	(@PROPERTIES=
		Value	@TYPE=Boolean;
	)
)

(@OBJECT=	OutAnal_Ctl_1
	(@PROPERTIES=
		Value	@TYPE=Boolean;
	)
)

(@OBJECT=	OutAnal_Ctl_Resets
	(@PROPERTIES=
		Value	@TYPE=Boolean;
	)
)

(@OBJECT=	Outlier_count
	(@PROPERTIES=
		Value	@TYPE=Integer;
	)
)

(@OBJECT=	Perc_err
	(@PROPERTIES=
		Value	@TYPE=Float;
	)
)



(@OBJECT=	Plot_Elements
	(@PROPERTIES=
		Value	@TYPE=Boolean;
	)
)

(@OBJECT=	Plot_What
	(@PROPERTIES=
		Value	@TYPE=String;
```

```
        )
)

(@OBJECT=     pos_limit
        (@PROPERTIES=
                Value    @TYPE=Float;
        )
)

(@OBJECT=     Print_Record
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Re_set_physical
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Record_Msr
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Rising
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Run_Xgraph
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Seq_Ret_Outliers
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Show_Analysis_Results
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)
```

```
(@OBJECT=     Temp_Msr
        (@PROPERTIES=
                Discharge
                Msr_Date
                Msr_Id
                Rem_A
                Rem_B
                Rem_C
                Stage
                Type_Set
        )
)

(@OBJECT=     Trib
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Weed
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@OBJECT=     Wind
        (@PROPERTIES=
                Value    @TYPE=String;
        )
)

(@SLOT=       Beaver
        @PROMPT="WAS THERE Beaver Activity NEAR THE SITE DURING THE MEASUREMENT
?";
)

(@SLOT=       Begin_Analysis
        @COMMENTS="Initiates the Outlier Analysis loop";
        (@CACTIONS=
                (Do       (0)       (Outlier_count))
                (Do       (0)       (Cursor))
                (Execute            ("rm -f atrrec.IN")          (@TYPE=EXE;))
                (Reset    (Seq_Ret_Outliers))
                (Do       (Seq_Ret_Outliers)          (Seq_Ret_Outliers))
                (Execute            ("cat eof_mark1.txt >> atrrec.IN") (@TYPE=EXE;))
                (Execute            ("atrrec")          (@TYPE=EXE;))
                (Execute            ("cat atrrec.OUT3 > Phs3.dat")     (@TYPE=EXE;))
                (Reset    (Begin_Analysis))
        )
)
```

```
(@SLOT=         Calc_limits
        @INFCAT=400;
        (@CACTIONS=
                (Reset  (Model_response))
                (Reset  (neg_limit))
                (Reset  (pos_limit))
                (Do     ((BEST_Model.B0)+(BEST_Model.B1*POW(Temp_Msr.Stage,\
BEST_Model.TX)))        (Model_response))
                (Do     ((Model_response-Temp_Msr.Discharge)/Model_response
*100)   (Perc_err))
                (Do     (Model_response*(1-Allow_err/100))       (neg_limit))
                (Do     (Model_response*(1+Allow_err/100))       (pos_limit))
                (Reset  (Calc_limits))
        )
)

(@SLOT=         Ctrl_1_Next
        @PROMPT="        What Next ?         ";
)



(@SLOT=         Cutoff
        @PROMPT="HAVE THERE BEEN ANY RECENT  Meandering Cut-Offs  NEAR THE SITE
?";
)

(@SLOT=         Debris
        @PROMPT="WAS THERE ANY TRAPPED  Debris  AFFECTING THE MEASUREMENT ?";
)

(@SLOT=         Falling
        @PROMPT="WAS  THE  MEASUREMENT  TAKEN  DURING  Rapidly  Falling  Stage
CONDITIONS ?";
)

(@SLOT=         Model_response
        @INFCAT=110;
        @FORMAT="u.000";
)

(@SLOT=         neg_limit
        @INFCAT=100;
        @FORMAT="u.000";
)

(@SLOT=         OutAnal_Ctl_Resets
        (@CACTIONS=
                (Reset  (OutAnal_Ctl_0))
                (Reset  (OutAnal_Ctl_1))
                (Reset  (OutAnal_Ctl_Resets))
        )
)

(@SLOT=       Perc_err
        @INFCAT=100;
        @FORMAT="u.000";
)



(@SLOT=       Plot_What
        @PROMPT="Would you like the stratified outlier data ?";
)

(@SLOT=       pos_limit
        @INFCAT=100;
        @FORMAT="u.000";
)

(@SLOT=       Re_set_physical
        (@CACTIONS=
                (Reset   (Rising))
                (Reset   (Falling))
                (Reset   (Beaver))
                (Reset   (Debris))
                (Reset   (Trib))
                (Reset   (Weed))
                (Reset   (Wind))
                (Reset   (Re_set_physical))
        )
)

(@SLOT=       Rising
        @PROMPT="WAS THE MEASUREMENT TAKEN DURING Rapidly Rising Stage
CONDITIONS ?";
)

(@SLOT=       Run_Xgraph
        (@CACTIONS=
                (Do      (C_Stn.Stn_Id)   (Plot_Info.Title))
                (Do      (STRCAT(FLOAT2STR(Plot_Info.Max_S*1.25),STRCAT(" ",\
FLOAT2STR(Plot_Info.Max_D*1.25))))   (dummy_string))
                (Execute        ("XGR   @V(Plot_Info.Title)   @v(Plot_Info.Filename)
@V(dummy_string)")   (@TYPE=EXE;))
                (Reset   (Run_Xgraph))
        )
)

(@SLOT=       Show_Analysis_Results
        @COMMENTS="Displays Outlier Report and calls plotting routine.";
        (@CACTIONS=
                (Execute        ("mv atrrec.OUT2 atrrec.txt")      (@TYPE=EXE;))
                (Show   ("atrrec.txt")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Reset  (Plot_Elements))
                (Do     (Plot_Elements)  (Plot_Elements))
                (Execute        ("ControlSession")        (@STRING="@STOP";))
                (Reset  (Show_Analysis_Results))
        )
)

(@SLOT=         Trib
        @PROMPT="DURING THE MEASUREMENT, WERE THERE ANY NEARBY TRIBUTARIES
WITH ...";
)

(@SLOT=         Weed
        @PROMPT="WAS THERE SUBSTATIAL  Weed Growth  IN THE MEASUREMENT
CROSS-SECTION ?";
)

(@SLOT=         Wind
        @PROMPT="WAS THERE ANY  Wind  AT THE SITE DURING MEASUREMENT ?";
)

(@RULE=         R200
        @INFCAT=-25;
        (@LHS=
                (Is     (Out_side)        ("BELOW"))
                (No     (Msr_Attribs.BV))
                (No     (Msr_Attribs.CO))
                (No     (Msr_Attribs.DB))
                (No     (Msr_Attribs.RF))
                (No     (Msr_Attribs.RR))
                (No     (Msr_Attribs.TR))
                (No     (Msr_Attribs.WE))
                (No     (Msr_Attribs.WN))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)            (Msr_Attribs.UX))
                (Do     ("UX")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)          (Attrib_Count))
        )
)

(@RULE=         R199
        @INFCAT=-25;
        (@LHS=
                (Is     (Out_side)        ("ABOVE"))
                (No     (Msr_Attribs.BV))
                (No     (Msr_Attribs.CO))
                (No     (Msr_Attribs.DB))
                (No     (Msr_Attribs.RF))
                (No     (Msr_Attribs.RR))
                (No     (Msr_Attribs.TR))
                (No     (Msr_Attribs.WE))
                (No     (Msr_Attribs.WN))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.UX))
                (Do     ("UX")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=         Impossible_Negative_Responses
        @INFCAT=-50;
        @COMMENTS="This rule is used ONLY to present Negativeresponse choices for the Outlier
Questions.";
        (@LHS=
                (>      (0)     (1))
                (Is     (Rising)        ("NO"))
                (Is     (Falling)       ("NO"))
                (Is     (Beaver)        ("Nil"))
                (Is     (Wind)  ("Nil"))
                (Is     (Debris)        ("None"))
                (Is     (Weed)  ("None"))
                (Is     (Trib)  ("All Tributaries: Normal Flow"))
                (Is     (Cutoff)        ("None"))
        )
        (@HYPO=         Cat_outlier)
)

(@RULE=         R197
        @INFCAT=-100;
        (@LHS=
                (>      (Attrib_Count)  (1))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.MU))
                (Do     ("MU")  (Temp_Msr.Type_Set))
        )
)

(@RULE=         R196
        @INFCAT=50;
        (@LHS=
                (Is     (Out_side)      ("ABOVE"))
                (Is     (Falling)       ("YES"))
        )
        (@HYPO=         Cat_outlier)
```

```
          (@RHS=
                    (Do       (TRUE)          (Msr_Attribs.RF))
                    (Do       ("RF")  (Temp_Msr.Type_Set))
                    (Do       (Attrib_Count+1)          (Attrib_Count))
          )
)

(@RULE=        R195
          @INFCAT=40;
          (@LHS=
                    (Is       (Out_side)         ("BELOW"))
                    (Is       (Beaver)           ("Up Stream"))
          )
          (@HYPO=        Cat_outlier)
          (@RHS=
                    (Do       (TRUE)          (Msr_Attribs.BV))
                    (Do       ("BV")  (Temp_Msr.Type_Set))
                    (Do       (Attrib_Count+1)          (Attrib_Count))
          )
)

(@RULE=        R194
          @INFCAT=25;
          (@LHS=
                    (Is       (Out_side)         ("BELOW"))
                    (Is       (Wind)  ("DOWN STREAM"))
          )
          (@HYPO=        Cat_outlier)
          (@RHS=
                    (Do       (TRUE)          (Msr_Attribs.WN))
                    (Do       ("WN")  (Temp_Msr.Type_Set))
                    (Do       (Attrib_Count+1)          (Attrib_Count))
          )
)

(@RULE=        R193
          @INFCAT=30;
          (@LHS=
                    (Is       (Out_side)         ("ABOVE"))
                    (Is       (Wind)  ("UP STREAM"))
          )
          (@HYPO=        Cat_outlier)
          (@RHS=
                    (Do       (TRUE)          (Msr_Attribs.WN))
                    (Do       ("WN")  (Temp_Msr.Type_Set))
                    (Do       (Attrib_Count+1)          (Attrib_Count))
          )
)

(@RULE=        R192
          @INFCAT=40;
```

```
        (@LHS=
                (Is     (Out_side)      ("ABOVE"))
                (Is     (Beaver)        ("Down Stream"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.BV))
                (Do     ("BV")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=         R191
        @INFCAT=35;
        (@LHS=
                (Is     (Out_side)      ("ABOVE"))
                (Is     (Debris)        ("Down Stream of Gauge"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.DB))
                (Do     ("DB")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=         R190
        @INFCAT=35;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Debris)        ("Up Stream of Gauge"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.DB))
                (Do     ("DB")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=         R189
        @INFCAT=15;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Weed)  ("Yes"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.WE))
                (Do     ("WE")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=     R188
        @INFCAT=15;
        (@LHS=
                (Is     (Out_side)      ("ABOVE"))
                (Is     (Weed)  ("YES"))
        )
        (@HYPO=     Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.WE))
                (Do     ("WE")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=     R187
        @INFCAT=50;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Rising)        ("YES"))
        )
        (@HYPO=     Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.RR))
                (Do     ("RR")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=     R186
        @INFCAT=35;
        (@LHS=
                (Is     (Out_side)      ("ABOVE"))
                (Is     (Cutoff)        ("Up Stream of Gauge"))
        )
        (@HYPO=     Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.CO))
                (Do     ("CO")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=     R185
        @INFCAT=35;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Cutoff)        ("Down Stream of Gauge"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.CO))
                (Do     ("CO")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=         Below_Trib_No2_B
        @INFCAT=30;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Trib)  ("U/S: Higher than Normal Flow"))
                (Yes    (Msr_Attribs.TR))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.TR))
                (Do     ("TR")  (Temp_Msr.Type_Set))
        )
)

(@RULE=         Below_Trib_No2_A
        @INFCAT=31;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Trib)  ("U/S: Higher than Normal Flow"))
                (No     (Msr_Attribs.TR))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.TR))
                (Do     ("TR")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)

(@RULE=         Below_Trib_No1
        @INFCAT=32;
        (@LHS=
                (Is     (Out_side)      ("BELOW"))
                (Is     (Trib)  ("D/S: Lower than Normal Flow"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.TR))
                (Do     ("TR")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)        (Attrib_Count))
        )
)
```

```
(@RULE=      Above_Trib_No2_B
        @INFCAT=30;
        (@LHS=
                (Is     (Out_side)       ("ABOVE"))
                (Is     (Trib)   ("U/S: Lower than Normal Flow"))
                (Yes    (Msr_Attribs.TR))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)           (Msr_Attribs.TR))
                (Do     ("TR")  (Temp_Msr.Type_Set))
        )
)

(@RULE=      Above_Trib_No2_A
        @INFCAT=31;
        (@LHS=
                (Is     (Out_side)       ("ABOVE"))
                (Is     (Trib)   ("U/S: Lower than Normal Flow"))
                (No     (Msr_Attribs.TR))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)           (Msr_Attribs.TR))
                (Do     ("TR")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)         (Attrib_Count))
        )
)

(@RULE=      Above_Trib_No1
        @INFCAT=32;
        (@LHS=
                (Is     (Out_side)       ("ABOVE"))
                (Is     (Trib)   ("D/S: Higher than Normal Flow"))
        )
        (@HYPO=         Cat_outlier)
        (@RHS=
                (Do     (TRUE)           (Msr_Attribs.TR))
                (Do     ("TR")  (Temp_Msr.Type_Set))
                (Do     (Attrib_Count+1)         (Attrib_Count))
        )
)

(@RULE=      Reject_below_curve
        @INFCAT=45;
        (@LHS=
                (>=     (Cursor)         (0))
                (<      (Perc_err)       (0.000))
                (>      (ABS(Perc_err)-Allow_err)        (0.000))
        )
        (@HYPO=         Check_Outlier)
        (@RHS=
                (Do     (Outlier_count+1)         (Outlier_count))
                (Do     (STRCAT(INT2STR(Outlier_count),STRCAT(" ",\
STRCAT(INT2STR(BEST_Model.Num_OL),STRCAT(" ",\
FLOAT2STR(Model_response))))))          (dummy_string))
                (Do     ("BELOW")       (Out_side))
                (Execute        ("shw_1  @V(Temp_Msr.Msr_Date)   @V(Temp_Msr.Stage)
@V(Temp_Msr.Discharge) @V(dummy_\
string) @V(Out_side)")   (@TYPE=EXE;))
                (Show   ("ol_set.txt")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Strategy       (@EXHBWRD=TRUE;))
                (Do     (0)     (Attrib_Count))
                (Reset  (Cat_outlier))
                (Do     (Cat_outlier)       (Cat_outlier))
                (Strategy       (@EXHBWRD=FALSE;))
                (Do     (TRUE)          (Re_set_physical))
        )
)

(@RULE=         Reject_above_curve
        @INFCAT=45;
        (@LHS=
                (>=     (Cursor)        (0))
                (>      (Perc_err)      (0.000))
                (>      (ABS(Perc_err)-Allow_err)       (0.000))
        )
        (@HYPO=         Check_Outlier)
        (@RHS=
                (Do     (Outlier_count+1)         (Outlier_count))
                (Do     (STRCAT(INT2STR(Outlier_count),STRCAT(" ",\
STRCAT(INT2STR(BEST_Model.Num_OL),STRCAT(" ",\
FLOAT2STR(Model_response))))))          (dummy_string))
                (Do     ("ABOVE")       (Out_side))
                (Execute        ("shw_1  @V(Temp_Msr.Msr_Date)   @V(Temp_Msr.Stage)
@V(Temp_Msr.Discharge) @V(dummy_\
string) @V(Out_side)")   (@TYPE=EXE;))
                (Show   ("ol_set.txt")      (@KEEP=FALSE;@WAIT=FALSE;))
                (Strategy       (@EXHBWRD=TRUE;))
                (Do     (0)     (Attrib_Count))
                (Reset  (Cat_outlier))
                (Do     (Cat_outlier)       (Cat_outlier))
                (Strategy       (@EXHBWRD=FALSE;))
                (Do     (TRUE)          (Re_set_physical))
        )
)

(@RULE=         Accept_SD_set
        @INFCAT=50;
        @COMMENTS="Accept S-D measure, assign OK attribute";
        (@LHS=
                (>=     (Cursor)        (0))
                (<=     (ABS(Perc_err)-Allow_err)       (0.000))
        )
        (@HYPO=         Check_Outlier)
        (@RHS=
                (Do     (TRUE)          (Msr_Attribs.OK))
                (Do     ("OK")  (Temp_Msr.Type_Set))
                (Do     (TRUE)          (Re_set_physical))
        )
)


(@RULE=         R205
        (@LHS=
                (Is     (<|Models|>.Plot)               (TRUE))
        )
        (@HYPO=         List_curve_A)
        (@RHS=
                (Execute        ("AtomNameValue")
(@ATOMID=<|Models|>;@STRING="@RETURN=Plot_Info.Obj_names,\
@NAMES";))
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names;@STRING="@RETURN=Plot_Info.Num_obj,\
@LENGTH";))
                (Do     (1)     (Plot_Info.Index))
                (Do     (List_curve_B)  (List_curve_B))
                (Reset  (List_curve_B))
        )
)

(@RULE=         R206
        (@LHS=
                (>=     (Plot_Info.Num_obj-Plot_Info.Index)     (0))
        )
        (@HYPO=         List_curve_B)
        (@RHS=
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names,dummy_obj.Value;\
@STRING="@INDEX=@V(Plot_Info.Index)";))
                (Do     (\dummy_obj\.FN_Curve)          (dummy_file))
                (Do     (STRCAT("\\"",\dummy_obj\.Title))       (dummy_title))
                (Execute        ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute        ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     ("-1 -1")       (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     ("")    (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     (Plot_Info.Index+1)     (Plot_Info.Index))
                (Reset    (List_curve_B))
        )
)

(@RULE=         R207
        (@LHS=
                (Is       (<|Datasets|>.Plot)          (TRUE))
        )
        (@HYPO=         List_data_A)
        (@RHS=
                (Execute            ("AtomNameValue")
(@ATOMID=<|Datasets|>;@STRING="@RETURN=Plot_Info.Obj_names,\
@NAMES";))
                (Execute            ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names;@STRING="@RETURN=Plot_Info.Num_obj,\
@LENGTH";))
                (Do       (1)        (Plot_Info.Index))
                (Do       (List_data_B)      (List_data_B))
                (Reset    (List_data_B))
        )
)

(@RULE=         R208
        (@LHS=
                (>=       (Plot_Info.Num_obj-Plot_Info.Index)          (0))
        )
        (@HYPO=         List_data_B)
        (@RHS=
                (Execute            ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names,dummy_obj.Value;\
@STRING="@INDEX=@V(Plot_Info.Index)";))
                (Do       (\dummy_obj\.FN_Data)   (dummy_file))
                (Do       (STRCAT("\\"",\dummy_obj\.Title))          (dummy_title))
                (Execute            ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute            ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do       ("-1 -1")           (dummy_string))
                (Execute            ("echo @V(dummy_string) >> @V(Plot_Info.filename)")
(@TYPE=EXE;))
                (Do       ("")       (dummy_string))
                (Execute            ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do       (Plot_Info.Index+1)         (Plot_Info.Index))
                (Reset    (List_data_B))
        )
)

(@RULE=         R209
        (@LHS=
                (<=       (DS_num)          (5))
        )
        (@HYPO=     Make_OL_DS)
        (@RHS=
                (Do     (STRCAT("DS_",INT2STR(DS_num)))     (Current_DS))
                (DeleteObject     (\Current_DS\))
                (CreateObject     (\Current_DS\)   (|Datasets|))
                (Do     (STRCAT(Current_DS,".dat"))     (\Current_DS\FN_Data))
                (Do     (TRUE)          (\Current_DS\Plot))
                (Do     (STRCAT("OL_type",INT2STR(DS_num)))          (\Current_DS\Title))
                (Do     (DS_num+1)     (DS_num))
                (Reset  (Make_OL_DS))
        )
)

(@RULE=     Start_System_Controls
        (@LHS=
                (Is     (Lock_OutAnal)  ("ON"))
        )
        (@HYPO=     OutAnal_Control)
        (@RHS=
                (Do     (\OutAnal_Call\Value)   (Control_Return))
                (Do     (TRUE)          (OutAnal_Ctl_Resets))
                (Reset  (OutAnal_Control))
        )
)

(@RULE=     Return_to_Main_System
        (@LHS=
                (Execute          ("rm -f bML.*")  (@TYPE=EXE;))
                (Is     (Lock_OutAnal)  ("OFF"))
        )
        (@HYPO=     OutAnal_Control)
)

(@RULE=     Begin_Outlier_Analysis
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=     OutAnal_Ctl_0)
        (@RHS=
                (Do     (0)     (Cursor))
                (Do     (TRUE)          (Begin_Analysis))
                (Do     (TRUE)          (Show_Analysis_Results))
                (Do     ("OutAnal_Ctl_1")       (OutAnal_Call))
        )
)


(@RULE=     Review_Plot_Display
        @COMMENTS=".";
        (@LHS=
                (Is         (Ctrl_1_Next)      ("Re-View Plot"))
        )
        (@HYPO=             OutAnal_Ctl_1)
        (@RHS=
                (Do         (TRUE)             (Run_Xgraph))
                (Reset      (Ctrl_1_Next))
                (Reset      (OutAnal_Ctl_1))
        )
)

(@RULE=         Continue_To_Model_as_Phase3
        @COMMENTS=".";
        (@LHS=
                (Is         (Ctrl_1_Next)      ("Continue...Model ACCEPTED Data"))
        )
        (@HYPO=             OutAnal_Ctl_1)
        (@RHS=
                (Execute            ("CopyFrame")
(@ATOMID=Phs2,Phs3;@STRING="@STRAT=SET";))
                (Execute            ("rm -f Phs2.dat")          (@TYPE=EXE;))
                (Do         (Phs2.Num_Msrs-BEST_Model.Num_OL)  (Phs3.Num_Msrs))
                (Do         ("Phs3.dat")       (Phs3.FN_Data))
                (Do         ("ACCEPTED")  (Phs3.Title))
                (Do         ("OutlierAnalysis_Return")          (System_Call))
                (Execute            ("rm -f Phs1.dat")          (@TYPE=EXE;))
                (Execute            ("rm -f Phs2.dat")          (@TYPE=EXE;))
                (Do         ("OFF")            (Lock_OutAnal))
        )
)

(@RULE=         Abort_Curve_Modification
        @COMMENTS=".";
        (@LHS=
                (Is         (Ctrl_1_Next)      ("Abort -> Exit System"))
        )
        (@HYPO=             OutAnal_Ctl_1)
        (@RHS=
                (Do         ("EndSession")   (System_Call))
                (Do         ("OFF")            (Lock_OutAnal))
                (Execute            ("rm -f Phs1.dat")          (@TYPE=EXE;))
                (Execute            ("rm -f Phs2.dat")          (@TYPE=EXE;))
                (Execute            ("rm -f Phs3.dat")          (@TYPE=EXE;))
        )
)


(@RULE=         R215
        (@LHS=
                (Is         (Plot_What)        ("Data, Outliers with Curve"))
        )
        (@HYPO=         Plot_Elements)
        (@RHS=
                (Execute          ("rm -f xgdat")   (@TYPE=EXE;))
                (Do      ("xgdat")          (Plot_Info.Filename))
                (Execute          ("cat atrrec.OUT1 > @V(Plot_Info.Filename)")       (@TYPE=EXE;))
                (Do      ("\"Model")        (dummy_string))
                (Execute          ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute          ("cat  @V(BEST_Model.FN_Curve)  >>
@V(Plot_Info.Filename)")          (@TYPE=EXE;))
                (Do      (TRUE)             (Run_Xgraph))
        )
)

(@RULE=         R215
        (@LHS=
                (Is      (Plot_What)        ("Data and Outliers ONLY"))
        )
        (@HYPO=         Plot_Elements)
        (@RHS=
                (Execute          ("rm -f xgdat")   (@TYPE=EXE;))
                (Do      ("xgdat")          (Plot_Info.Filename))
                (Execute          ("cat atrrec.OUT1 > @V(Plot_Info.Filename)")       (@TYPE=EXE;))
                (Do      (TRUE)             (Run_Xgraph))
        )
)

(@RULE=         R214
        (@LHS=
                (Is      (Plot_What)        ("Exit Plotting"))
        )
        (@HYPO=         Plot_Elements)
)

(@RULE=         Output_Data_to_OLORG_Input_File
        @COMMENTS="Write ATTRIBUTE CODE, DISCHARGE, and STAGE to the atrrec.IN file and
then reset the Msr_Attribute properties and Type_Set";
        (@LHS=
                (>=      (Cursor)           (0))
        )
        (@HYPO=         Record_Msr)
        (@RHS=
                (Do      (STRCAT(Temp_Msr.Type_Set,STRCAT("
",STRCAT(FLOAT2STR(Temp_Msr.Discharge),\
STRCAT(" ",FLOAT2STR(Temp_Msr.Stage)))))) (Temp_Msr.Type_Set))
                (Execute          ("echo @V(Temp_Msr.Type_Set) >> atrrec.IN")    (@TYPE=EXE;))
                (Execute          ("SetValue")
(@ATOMID=Msr_Attribs;@STRING="@VALUE=False,\
@STRAT=SET";))
                (Reset   (Temp_Msr.Type_Set))
        )
```

```
)

(@RULE=         Retrv_Each_Msr_in_Sequence
        (@LHS=
                (>=     (Cursor)        (0))
        )
        (@HYPO=         Seq_Ret_Outliers)
        (@RHS=
                (Show   ("msg_33.txt")  (@KEEP=FALSE;@WAIT=FALSE;@RECT=5,400,650,\
400;))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@SLOTS=Temp_Msr.Stage,Temp_Msr.Discharge,\
Temp_Msr.Msr_Id,Temp_Msr.Msr_Date,Temp_Msr.Rem_A,\
Temp_Msr.Rem_B,Temp_Msr.Rem_C;@FIELDS="STAGE",\
"DISCHARGE","MSR_ID","DATE_OF_MSR","REM_A",\
"REM_B","REM_C";@QUERY="@V(Query_String)";\
@CURSOR=Cursor;))
                (Do     (TRUE)          (Calc_limits))
                (Reset  (Check_Outlier))
                (Do     (Check_Outlier) (Check_Outlier))
                (Reset  (Record_Msr))
                (Do     (Record_Msr)    (Record_Msr))
                (Reset  (Seq_Ret_Outliers))
        )
)


#*******************************************************************
#*  End of KB Code:  A4_main.tkb                                    *
#*******************************************************************
```

```
#*******************************************************************
#*  Nexpert Object KB Code:  A5_main.tkb                           *
#*******************************************************************
#*  Curve Modification Module                                      *
#*******************************************************************


(@VERSION=   020)


(@OBJECT=     Build_Query
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Chg_Max_D
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Chg_Max_S
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Chg_Min_D
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Chg_Min_S
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     Crv_Generate
        (@PROPERTIES=
                Value    @TYPE=Boolean;
        )
)

(@OBJECT=     CrvMod_Ctl_2
        (@PROPERTIES=
                Value    @TYPE=Boolean;
```

```
          )
)

(@OBJECT=     CrvMod_Ctl_3
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_4
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_5
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_6
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_7
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_8
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_9
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     CrvMod_Ctl_Resets
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)
```

```
(@OBJECT=    Ctrl_8_Next
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=    CV_NEW
        (@CLASSES=
                Curves
        )
        (@PROPERTIES=
                B0
                B1
                Crv_Type
                Datemade
                F_Day
                FN_Curve
                L_Day
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                TX
                Value   @TYPE=Integer;
        )
)

(@OBJECT=    CV_OLD
        (@CLASSES=
                Curves
        )
        (@PROPERTIES=
                B0
                B1
                Crv_Type
                Datemade
                F_Day
                FN_Curve
                L_Day
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                TX
                Value   @TYPE=Integer;
        )
)

(@OBJECT=    Data_Xtras
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=    DS_Name
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=    DS_NEW
        (@CLASSES=
                Datasets
        )
        (@PROPERTIES=
                Current
                Description
                FN_Data
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                No_Flow_Stg
                Num_Msrs
                Plot
                Query
                Title
                Type
                ZF_Count
        )
)

(@OBJECT=    DS_OLD
        (@CLASSES=
                Datasets
        )
        (@PROPERTIES=
                B0
                B1
                Crv_Type
                Current
                Description
                F_Day
                FN_Data
                L_Day
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                No_Flow_Stg
                Num_Msrs
                Plot
                Query
                Title
                Type
                ZF_Count
        )
)

(@OBJECT=       DS_OLD_NEW
        (@CLASSES=
                Datasets
        )
        (@PROPERTIES=
                Current
                Description
                FN_Data
                Last_Msr
                Max_D
                Max_S
                Min_D
                Min_S
                No_Flow_Stg
                Num_Msrs
                Plot
                Query
                Title
                Type
                ZF_Count
        )
)

(@OBJECT=       dummy_file
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=       dummy_obj
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)

(@OBJECT=       dummy_title
        (@PROPERTIES=
                Value   @TYPE=String;
        )
)
```

```
(@OBJECT=     Get_Zero_Flow
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Label_Msr
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     List_curve
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     List_data_A
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     List_data_B
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Put_Msr_NEW
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Put_Msr_OLD
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
          )
)

(@OBJECT=     Qry_Str_Suffix
          (@PROPERTIES=
                    Value    @TYPE=String;
          )
)

(@OBJECT=     Run_Xgraph
          (@PROPERTIES=
                    Value    @TYPE=Boolean;
```

```
        )
)

(@OBJECT=     Seq_Ret_Data
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Setup_Plot
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@OBJECT=     Store_Msr
        (@PROPERTIES=
                Value   @TYPE=Boolean;
        )
)

(@SLOT=       Crv_Generate
        @COMMENTS="Generate Curve Points for Plotting, based on Model Parameters Generated.";
        (@CACTIONS=
                (Do     (STRCAT(FLOAT2STR(C_Crv.B0),STRCAT("
",STRCAT(FLOAT2STR(C_Crv.B1),\
" ")))) (dummy_params))
                (Do     (STRCAT(dummy_params,STRCAT(FLOAT2STR(C_Crv.TX),\
STRCAT(" ",STRCAT(FLOAT2STR(C_Crv.Max_S),\
" Old_Crv"))))) (dummy_params))
                (Execute        ("GC @V(dummy_params)")      (@TYPE=EXE;))
                (Do     ("Old_Crv.mdl") (C_Crv.FN_Curve))
                (Reset  (Crv_Generate))
        )
)

(@SLOT=       CrvMod_Ctl_Resets
        (@CACTIONS=
                (Reset  (CrvMod_Ctl_2))
                (Reset  (CrvMod_Ctl_3))
                (Reset  (CrvMod_Ctl_4))
                (Reset  (CrvMod_Ctl_5))
                (Reset  (CrvMod_Ctl_6))
                (Reset  (CrvMod_Ctl_7))
                (Reset  (CrvMod_Ctl_8))
                (Reset  (CrvMod_Ctl_9))
                (Reset  (CrvMod_Ctl_Resets))
        )
)

(@SLOT=       DS_NEW.FN_Data
        (@INITVAL=    "ds.NEW")
)

(@SLOT=         DS_OLD.FN_Data
        (@INITVAL=    "ds.OLD")
)

(@SLOT=         DS_OLD_NEW.FN_Data
        (@INITVAL=    "ds.OLD_NEW")
)

(@SLOT=         Run_Xgraph
        (@CACTIONS=
                (Do     (C_Stn.Stn_Id)   (Plot_Info.Title))
                (Do     (STRCAT(FLOAT2STR(Plot_Info.Max_S*1.25),STRCAT(" ",\
FLOAT2STR(Plot_Info.Max_D*1.25))))    (dummy_string))
                (Execute        ("XGR   @V(Plot_Info.Title)   @V(Plot_Info.Filename)
@V(dummy_string)")      (@TYPE=EXE;))
                (Reset  (Run_Xgraph))
        )
)

(@SLOT=         Setup_Plot
        (@CACTIONS=
                (Do     (FALSE)         (<|Datasets|>.Plot))
                (Do     (TRUE)          (DS_OLD.Plot))
                (Do     (TRUE)          (DS_NEW.Plot))
                (Reset  (Setup_Plot))
        )
)

(@RULE=         Construct_Time_B_Query
        @COMMENTS="First Day in time range is later in year than Last Day";
        (@LHS=
                (Is     (C_Crv.Crv_Type)        ("TIME"))
                (>      (C_Crv.F_Day-C_Crv.L_Day)       (0))
        )
        (@HYPO=         Build_Query)
        (@RHS=
                (Do     (" WHERE ")     (Qry_Str_Suffix))
                (Do     (STRCAT(Qry_Str_Suffix,Cnst_time2))     (Qry_Str_Suffix))
                (Do     (C_Crv.F_Day)   (Start_day))
                (Do     (C_Crv.L_Day)   (Stop_day))
        )
)

(@RULE=         Construct_Time_A_Query
        @COMMENTS="First Day in time range is before Last Day in time range";
        (@LHS=
                (Is     (C_Crv.Crv_Type)        ("TIME"))
                (>      (C_Crv.L_Day-C_Crv.F_Day)       (0))
        )
        (@HYPO=       Build_Query)
        (@RHS=
                (Do     (" WHERE ")    (Qry_Str_Suffix))
                (Do     (STRCAT(Qry_Str_Suffix,Cnst_time1))     (Qry_Str_Suffix))
                (Do     (C_Crv.F_Day)  (Start_day))
                (Do     (C_Crv.L_Day)  (Stop_day))
        )
)

(@RULE=       Construct_ALL_Query
        @COMMENTS="Query String selects ALL data, no restrictions.";
        (@LHS=
                (Is     (C_Crv.Crv_Type)        ("ALL"))
        )
        (@HYPO=       Build_Query)
        (@RHS=
                (Do     (" ")     (Qry_Str_Suffix))
        )
)

(@RULE=       Check_Max_D
        (@LHS=
                (>      (New_msr.Discharge-\DS_Name\.Max_D)  (0))
        )
        (@HYPO=       Chg_Max_D)
        (@RHS=
                (Do     (New_msr.Discharge)     (\DS_Name\.Max_D))
        )
)

(@RULE=       Check_Max_S
        (@LHS=
                (>      (New_msr.Stage-\DS_Name\.Max_S)        (0))
        )
        (@HYPO=       Chg_Max_S)
        (@RHS=
                (Do     (New_msr.Stage)         (\DS_Name\.Max_S))
        )
)

(@RULE=       Check_Min_D
        (@LHS=
                (<      (New_msr.Discharge-\DS_Name\.Min_D)  (0))
        )
        (@HYPO=       Chg_Min_D)
        (@RHS=
                (Do     (New_msr.Discharge)     (\DS_Name\.Min_D))
        )
)
```

```
(@RULE=         Check_Min_S
        (@LHS=
                (<      (New_msr.Stage-\DS_Name\.Min_S)         (0))
        )
        (@HYPO=         Chg_Min_S)
        (@RHS=
                (Do     (New_msr.Stage)         (\DS_Name\.Min_S))
        )
)

(@RULE=         CrvMod_System_Control
        @COMMENTS="This rule transfers returns control back to the main system.";
        (@LHS=
                (Is     (Lock_CrvMod)  ("OFF"))
        )
        (@HYPO=         CrvMod_Control)
)

(@RULE=         CrvMod_System_Control
        @COMMENTS="This rule controls the next control call.";
        (@LHS=
                (Is     (Lock_CrvMod)  ("ON"))
        )
        (@HYPO=         CrvMod_Control)
        (@RHS=
                (Do     (\CrvMod_Call\.Value)   (Control_Return))
                (Do     (TRUE)          (CrvMod_Ctl_Resets))
                (Reset  (CrvMod_Control))
        )
)

(@RULE=         Make_Old_New_Objects
        @COMMENTS="make dynamic ovjects for old and new curves.";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_2)
        (@RHS=
                (Do     ("CrvMod_Ctl_3")        (CrvMod_Call))
        )
)

(@RULE=         Retrieve_OLD_Curve
        @COMMENTS="Retrieve OLD curve from database, according to Curve Entry selected by user.";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_3)
        (@RHS=
                (Reset  (Cursor))
                (Retrieve       ("db_access")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=DS_OLD.Crv_Type,\
DS_OLD.B0,DS_OLD.B1,DS_OLD.F_Day,DS_OLD.L_Day,\
DS_OLD.Last_Msr,DS_OLD.Max_D,DS_OLD.Max_S,\
DS_OLD.Min_D,DS_OLD.Min_S,DS_OLD.No_Flow_Stg;\
@FIELDS="TYPE","B0","B1","F_DAY","L_DAY",\
"LAST_MSR","MAX_D","MAX_S","MIN_D","MIN_S",\
"NO_FLOW_STG";@QUERY="@V(C_Stn.Crv_Tbl) where CRV_ID = @V(Curve_Entry)";\
@CURSOR=Cursor;))
                (Do     (TRUE)          (Setup_Plot))
                (Do     ("CrvMod_Ctl_4")        (CrvMod_Call))
        )
)

(@RULE=         Make_Query_String
        @COMMENTS="Construct the Query String based on the OLD curve.";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_4)
        (@RHS=
                (Do     (C_Stn.Dat_Tbl) (Qry_Str_Prefix))
                (Do     (" ")   (Qry_Str_Suffix))
                (Reset  (Build_Query))
                (Do     (Build_Query)   (Build_Query))
                (Do     (STRCAT(Qry_Str_Prefix,Qry_Str_Suffix))         (Query_String))
                (Do     ("CrvMod_Ctl_5")        (CrvMod_Call))
        )
)

(@RULE=         Make_Datasets
        @COMMENTS="Initiates the sequential retrieve of measurements according to Query String.  As
each measurements is retrieved, it is copied to the appropriate UNIX file, depending on the date_of_msr
and Last_Msr dates.";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_5)
        (@RHS=
                (Execute        ("rm -f ds.NEW")        (@TYPE=EXE;))
                (Execute        ("rm -f ds.OLD")        (@TYPE=EXE;))
                (Execute        ("rm -f ds.OLD_NEW")    (@TYPE=EXE;))
                (Execute        ("rm -f Old_Crv.*")     (@TYPE=EXE;))
                (Execute        ("rm -f xgdat")  (@TYPE=EXE;))
                (Show   ("msg_42.txt")  (@KEEP=FALSE;@WAIT=FALSE;))
                (Do     ("xgdat")       (Plot_Info.Filename))
                (Do     ("ds.NEW")      (DS_NEW.FN_Data))
                (Do     ("ds.OLD")      (DS_OLD.FN_Data))
                (Do     ("ds.OLD_NEW")          (DS_OLD_NEW.FN_Data))
                (Do     (0)     (DS_OLD_NEW.No_Flow_Stg))
                (Reset  (Get_Zero_Flow))
                (Do     (Get_Zero_Flow)         (Get_Zero_Flow))
                (Do     (0)     (Cursor))
                (Reset  (Seq_Ret_Data))
                (Do     (Seq_Ret_Data)  (Seq_Ret_Data))
                (Reset  (Cursor))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=DS_OLD_NEW.Last_Msr;
@FIELDS="MAX(DATE_OF_MSR)"; @QUERY="@V(Query_String)";@CURSOR=Cursor;))
                (Do     ("CrvMod_Ctl_6")        (CrvMod_Call))
        )
)

(@RULE=         Make_Datasets_Xtras
        @COMMENTS=".";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_6)
        (@RHS=
                (Do     (STRCAT(DATE2STR(C_Crv.Last_Msr,"d-MMM-yyyy"),   STRCAT("
",STRCAT(INT2STR(DS_OLD.Num_Msrs),STRCAT(" ",
INT2STR(DS_NEW.Num_Msrs))))))   (dummy_params))
                (Do     (STRCAT(dummy_params,STRCAT("
",STRCAT(FLOAT2STR(DS_OLD_NEW.No_Flow_Stg),
STRCAT(" ",STRCAT(INT2STR(DS_OLD.ZF_Count),
STRCAT(" ",INT2STR(DS_NEW.ZF_Count))))))))  (dummy_params))
                (Execute        ("mg3 @V(dummy_params)")        (@TYPE=EXE;))
                (Show   ("mg_3.txt")    (@KEEP=FALSE;@WAIT=FALSE;))
                (Do     ("OLD")         (DS_OLD.Title))
                (Do     ("NEW")         (DS_NEW.Title))
                (Do     ("OLD/NEW")     (DS_OLD_NEW.Title))
                (Do     (DS_OLD_NEW.Max_D*1.25)     (Plot_Info.Max_D))
                (Do     (DS_OLD_NEW.Max_S*1.25)     (Plot_Info.Max_S))
                (Reset  (List_data_A))
                (Do     (List_data_A)   (List_data_A))
                (Reset  (List_curve))
                (Do     (List_curve)    (List_curve))
                (Do     ("CrvMod_Ctl_7")        (CrvMod_Call))
        )
)

(@RULE=         Display_Data_and_Curve
        @COMMENTS=".";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_7)
        (@RHS=
                (Do     (TRUE)          (Run_Xgraph))
                (Reset  (Ctrl_8_Next))
                (Do     ("CrvMod_Ctl_8")        (CrvMod_Call))
        )
```

```
)

(@RULE=        Review_Plot_Display
        @COMMENTS=".";
        (@LHS=
                (Is        (Ctrl_8_Next)     ("Re-View Plot"))
        )
        (@HYPO=        CrvMod_Ctl_8)
        (@RHS=
                (Do        ("CrvMod_Ctl_7")          (CrvMod_Call))
        )
)

(@RULE=        Continue_To_Model
        @COMMENTS=".";
        (@LHS=
                (Is        (Ctrl_8_Next)     ("Continue...Model New Data"))
        )
        (@HYPO=        CrvMod_Ctl_8)
        (@RHS=
                (Reset    (Cursor))
                (Retrieve            ("db_access")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=DS_OLD_NEW.Type;\
@FIELDS="TYPE";@QUERY="@V(C_Stn.Crv_Tbl) where CRV_ID = @V(Crv_Entry)";\
@CURSOR=Cursor;))
                (Reset    (Cursor))
                (Retrieve            ("db_access")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=DS_OLD_NEW.Last_Msr;\
@FIELDS="MAX(DATE_OF_MSR)";@QUERY="@V(C_Stn.Crv_Tbl) where CRV_ID = @V(Crv_Entry\
)";@CURSOR=Cursor;))
                (Execute             ("CopyFrame")
(@ATOMID=DS_OLD_NEW,Phs5;@STRING="@STRAT=SET";\
))
                (Execute             ("cp ds.OLD_NEW Phs5.dat")      (@TYPE=EXE;))
                (Do        (C_Crv.Crv_Type)          (Phs5.Type))
                (Do        ("Phs5.dat")        (Phs5.FN_Data))
                (Do        ("CurveModify_Return")  (System_Call))
                (Do        ("CrvMod_Ctl_9")          (CrvMod_Call))
        )
)

(@RULE=        Abort_Curve_Modification
        @COMMENTS=".";
        (@LHS=
                (Is        (Ctrl_8_Next)     ("Abort -> Main Menu"))
        )
        (@HYPO=        CrvMod_Ctl_8)
        (@RHS=
                (Do        ("MainMenu_Start")        (System_Call))
                (Do        ("CrvMod_Ctl_9")          (CrvMod_Call))
        )
```

```
)

(@RULE=         Prepare_to_Exit_CrvMod
        @COMMENTS="This rule removes any Unix Files from the Operating System that were created
during the CrvMod Procedure...Before exitting.";
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         CrvMod_Ctl_9)
        (@RHS=
                (Execute        ("rm -f ds.NEW")        (@TYPE=EXE;))
                (Execute        ("rm -f ds.OLD")        (@TYPE=EXE;))
                (Execute        ("rm -f ds.OLD_NEW")    (@TYPE=EXE;))
                (Execute        ("rm -f Old_Crv.*")     (@TYPE=EXE;))
                (Execute        ("rm -f xg.dat")  (@TYPE=EXE;))
                (Do     ("OFF")         (Lock_CrvMod))
        )
)

(@RULE=         Check_Max_Min_Values
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         Data_Xtras)
        (@RHS=
                (Do     (\DS_Name\FN_Data)      (dummy_string))
                (Execute        ("echo @V(New_msr.Set) >> @V(dummy_string)")
(@TYPE=EXE;))
                (Do     (\DS_Name\Num_Msrs+1)           (\DS_Name\Num_Msrs))
                (Reset  (Chg_Max_D))
                (Do     (Chg_Max_D)     (Chg_Max_D))
                (Reset  (Chg_Max_S))
                (Do     (Chg_Max_S)     (Chg_Max_S))
                (Reset  (Chg_Min_D))
                (Do     (Chg_Min_D)     (Chg_Min_D))
                (Reset  (Chg_Min_S))
                (Do     (Chg_Min_S)     (Chg_Min_D))
        )
)

(@RULE=         Get_Zero_FLow_Stage_B
        @COMMENTS="This rule is used fro retreiving the ZERO FLOW STAGE for the data retrieved
according to QUERY_STRING, for ALL data type. If no ZERO FLOWs are found, the NO_FLOW_STG
has been preset to zero in the CrvMod_Ctl_5 rule. Store in OLD_NEW dataset.";
        (@LHS=
                (Is     (C_Crv.Crv_Type)        ("ALL"))
        )
        (@HYPO=         Get_Zero_Flow)
        (@RHS=
                (Reset  (Cursor))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=DS_OLD_NEW.No_Flow_Stg;
@FIELDS="MAX(STAGE)";@QUERY="@V(Query_String) WHERE DISCHARGE = 0";
@CURSOR=Cursor;))
                (Do     (0)     (<|Datasets|>.ZF_Count))
        )
)

(@RULE=         Get_Zero_FLow_Stage_A
        @COMMENTS="This rule is used fro retreiving the ZERO FLOW STAGE for the data retrieved
according to QUERY_STRING, for TIME data type. If no ZERO FLOWs are found, the NO_FLOW_STG
has been preset to zero in the CrvMod_Ctl_5 rule. Store in OLD_NEW dataset.";
        (@LHS=
                (Is     (C_Crv.Crv_Type)        ("TIME"))
        )
        (@HYPO=         Get_Zero_Flow)
        (@RHS=
                (Reset  (Cursor))
                (Retrieve       ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=DS_OLD_NEW.No_Flow_Stg;
@FIELDS="MAX(STAGE)";@QUERY="@V(Query_String) AND DISCHARGE = 0";
@CURSOR=Cursor;))
                (Do     (0)     (<|Datasets|>.ZF_Count))
        )
)

(@RULE=         Label_Type_OLD
        @COMMENTS="The msr is identified as OLD.";
        (@LHS=
                (>=     (Cursor)        (0))
                (>=     ((DATE2FLOAT(C_Crv.Last_Msr))-(DATE2FLOAT(New_msr.Msr_Date)))   (0))
        )
        (@HYPO=         Label_Msr)
        (@RHS=
                (Do     ("DS_OLD")      (DS_Name))
                (Reset  (Store_Msr))
                (Do     (Store_Msr)     (Store_Msr))
        )
)

(@RULE=         Label_Type_NEW
        @COMMENTS="The msr is identified as NEW.";
        (@LHS=
                (>=     (Cursor)        (0))
                (>      ((DATE2FLOAT(New_msr.Msr_Date))-(DATE2FLOAT(C_Crv.Last_Msr)))   (0))
        )
        (@HYPO=         Label_Msr)
        (@RHS=
                (Do     ("DS_NEW")      (DS_Name))
                (Reset  (Store_Msr))
                (Do     (Store_Msr)     (Store_Msr))
        )
```

```
)

(@RULE=     Plot_Prep_Curve
        (@LHS=
                (=      (1)     (1))
        )
        (@HYPO=         List_curve)
        (@RHS=
                (Do     (TRUE)          (Crv_Generate))
                (Do     (C_Crv.FN_Curve)        (dummy_file))
                (Do     (STRCAT("\\"","OLD_CURVE")) (dummy_title))
                (Execute        ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute        ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do     ("-1 -1")       (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename")
(@TYPE=EXE;))
                (Do     ("")    (dummy_string))
                (Execute        ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
        )
)

(@RULE=     Plot_Prep_Data_Part1
        (@LHS=
                (Is     (<|Datasets|>.Plot)     (TRUE))
        )
        (@HYPO=         List_data_A)
        (@RHS=
                (Execute        ("AtomNameValue")
(@ATOMID=<|Datasets|>;@STRING="@RETURN=Plot_Info.Obj_names,
@NAMES";))
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names;@STRING="@RETURN=Plot_Info.Num_obj,
@LENGTH";))
                (Do     (1)     (Plot_Info.Index))
                (Do     (List_data_B)   (List_data_B))
                (Reset  (List_data_B))
        )
)

(@RULE=     Plot_Prep_Data_Part2
        (@LHS=
                (>=     (Plot_Info.Num_obj-Plot_Info.Index)     (0))
        )
        (@HYPO=         List_data_B)
        (@RHS=
                (Execute        ("GetMultiValue")
(@ATOMID=Plot_Info.Obj_names,dummy_obj.Value;
@STRING="@INDEX=@V(Plot_Info.Index)";))
```

```
                (Do      (\dummy_obj\FN_Data)  (dummy_file))
                (Do      (STRCAT("\\"",\dummy_obj\Title))        (dummy_title))
                (Execute         ("echo @V(dummy_title) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Execute         ("cat @V(dummy_file) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do      ("-1 -1")         (dummy_string))
                (Execute         ("echo @V(dummy_string) >> @V(Plot_Info.filename)")
(@TYPE=EXE;))
                (Do      ("")     (dummy_string))
                (Execute         ("echo @V(dummy_string) >> @V(Plot_Info.Filename)")
(@TYPE=EXE;))
                (Do      (Plot_Info.Index+1)       (Plot_Info.Index))
                (Reset   (List_data_B))
        )
)

(@RULE=         Seq_Ret_of_Measurements
        (@LHS=
                (>=      (Cursor)          (0))
        )
        (@HYPO=         Seq_Ret_Data)
        (@RHS=
                (Retrieve          ("@V(db_access)")
(@TYPE=ORACLE;@END="RELEASE";@SLOTS=New_msr.Stage,
New_msr.Discharge,New_msr.Msr_Id,New_msr.Msr_Date;
@FIELDS="stage","discharge","msr_id","date_of_msr";
@QUERY="@V(Query_String)";@CURSOR=Cursor;))
                (Do      (STRCAT(FLOAT2STR(New_msr.Discharge),STRCAT("
",FLOAT2STR(New_msr.Stage))))          (New_msr.Set))
                (Reset   (Label_Msr))
                (Do      (Label_Msr)      (Label_Msr))
                (Reset   (Seq_Ret_Data))
        )
)

(@RULE=         Print_S_D_to_Unix_A
        @INFCAT=20;
        @COMMENTS="DO NOT Print measurements with zero flow stages less than NO_FLOW_STG.
Count # of zero flow msrs and subtract from total number of msrs for dataset.";
        (@LHS=
                (>=      (Cursor)          (0))
                (=       (New_msr.Discharge)      (0.0))
                (NotEqual         (DS_OLD_NEW.No_Flow_Stg)    (New_msr.Stage))
        )
        (@HYPO=         Store_Msr)
        (@RHS=
                (Do      (\DS_Name\ZF_Count+1)          (\DS_Name\ZF_Count))
        )
)
```

```
(@RULE=     Print_S_D_to_Unix_B
        @INFCAT=5;
        @COMMENTS="Print S/D measurement to UNIX file.  DS_Name has been previously set in
Label_Msr rule. Also Store in DS_OLD_NEW dataset.";
        (@LHS=
                (>=     (Cursor)            (0))
                (<>     (New_msr.Discharge)      (0))
        )
        (@HYPO=     Store_Msr)
        (@RHS=
                (Reset  (Data_Xtras))
                (Do     (Data_Xtras)      (Data_Xtras))
                (Do     ("DS_OLD_NEW")        (DS_Name))
                (Reset  (Data_Xtras))
                (Do     (Data_Xtras)      (Data_Xtras))
        )
)

(@RULE=     Print_S_D_to_Unix_C
        @INFCAT=5;
        @COMMENTS="Print S/D measurement to UNIX file.  DS_Name has been previously set in
Label_Msr rule. Also Store in DS_OLD_NEW dataset.";
        (@LHS=
                (>=     (Cursor)            (0))
                (=      (New_msr.Discharge)      (0))
                (Equal  (DS_OLD_NEW.No_Flow_Stg)    (New_msr.Stage))
        )
        (@HYPO=     Store_Msr)
        (@RHS=
                (Reset  (Data_Xtras))
                (Do     (Data_Xtras)      (Data_Xtras))
                (Do     ("DS_OLD_NEW")        (DS_Name))
                (Reset  (Data_Xtras))
                (Do     (Data_Xtras)      (Data_Xtras))
        )
)


#*****************************************************************
#*  End of KB Code:  A5_main.tkb                                 *
#*****************************************************************
```