TR-163

A LUMPED PARAMETER MODEL FOR THE EDWARDS AQUIFER

Nisai Wanakule, Ph.D., P.E.
Robert Anaya

Texas Water Resources Institute

Texas A&M University

SEPTEMBER 1993

RESEARCH PROJECT COMPLETION REPORT

A LUMPED PARAMETER MODEL FOR THE EDWARDS AQUIFER

Project Number - SWTSU 92-1
September 1, 1992 - August 31, 1993
Grant Number
14-08-0001-G2048

by

Nisai Wanakule, Ph.D., P.E.
Roberto Anaya

The research on which this report is based was financed in part by the U. S. Department of Interior, Geological Survey, through the Texas Water Resources Institute. Non-Federal matching funds were provided by the Edwards Underground Water District, San Antonio, Texas, and Southwest Texas State University, San Marcos, Texas.

Technical Report No 163
Texas Water Resources Institute
The Texas A&M University System
College Station, TX 77843-2118

September 1993

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (CONT.)

LIST OF FIGURES (CONT.)

LIST OF TABLES

A LUMPED PARAMETER MODEL FOR THE EDWARDS AQUIFER

ABSTRACT

A lumped parameter model has been developed to simulate monthly water levels and spring flows in the Edwards Aquifer. It is less complex and easier to use than the existing complex finite difference models for the Edwards Aquifer. The lumped parameter model was formulated using a discrete, nonlinear, nonstationary system based on control theory. The physical system of the Edwards Aquifer is conceptualized as a series of connected rock filled tanks representing major drainage basins of the aquifer. The model incorporates recharge functions derived from flow loss analysis of the drainage basins above and within the recharge area. The recharge functions estimate monthly recharge and allow for the interaction between groundwater and surface water for each drainage basin. Pumpage distribution coefficients were derived for each drainage basin to estimate monthly pumpage values. Monthly stream gage and aquifer water level data were used for calibrating and verifying the model. Model parameters were obtained with the aid of a nonlinear optimization algorithim. A Kalman filter was used to improve simulation results. The lumped parameter model proved to be very efficient in simulating 189 monthly iterations of water levels for nine drainage basins in less than four minutes on a 68040 based microcomputer. The model should prove useful for assessing pumpage regulations necessary to maintain springflows under historic drought conditions and for exploring management alternatives for the Edwards Aquifer. Future plans and research consist of re-coding the model as a spreadsheet function and perhaps using a Laplace transform as a solution method. Other research should include generalizing the model for use in other karst aquifers, integrating the model with a rainfall-runoff model, and developing a better method for estimating recharge.

INTRODUCTION

The entire Edwards (Balcones Fault Zone) Aquifer extends along the narrow belt of the Balcones Fault Zone from north of Georgetown through Austin, San Marcos, New Braunfels, San Antonio, Hondo, Sabinal and Uvalde to Brackettville. This limestone aquifer is separated into three portions by groundwater divides at Kyle in Hays county and at the Colorado River. The central portion, from Kyle to Colorado River is referred to as the Barton Springs segment. The part between the Colorado River and Salado, in Bell County is referred to as the northern segment of the Edwards Aquifer. This study will focus on the western portion, referred to as the San Antonio region. The Edwards Aquifer to be referred to hereafter will be restricted to the San Antonio region.

The Edwards Aquifer is approximately 160 miles in length from Brackettville to Kyle and varies in width from 5 to 40 miles. It extends to cover the major part of five counties namely, Uvalde, Medina, Bexar, Comal and Hays (see Figure 1). It traverses several streams in three major river basins including the Nueces, San Antonio and Guadalupe. The aquifer is a very unique carbonate aquifer located in south-central Texas. Karst characteristics of the Edwards Aquifer make it one of the most productive aquifers in the United States and yet groundwater flow within the aquifer is very complex and difficult to predict. The Edwards Aquifer is designated by the EPA as a "sole source" drinking water supply for the 1.5 million people of San Antonio and the Austin-San Antonio corridor. The aquifer is also vital to the agricultural and light industrial economy of the region. Springflows from the Comal and San Marcos Springs provide water for the tourist and recreation industry, critical habitat of several endangered species, appropriated water use downstream on the Gulf Coastal Plain, and the San Antonio Bay ecosystem.

Background Information

Early Europeans first settled along perennial streams sustained by natural spring flows from the Edwards Aquifer (Maclay and Land, 1988). Substantial well discharge from the aquifer began in the late 1800's and steadily increased from 101,900 ac-ft in 1934 to a record high of 542,400 ac-ft in 1989. The groundwater is used extensively for public water supply and agriculture, and accounted for, respectively, 56.6% and 30.1% of the total well discharge during 1981-90. The continual increase in well discharge has had an effect on the natural spring

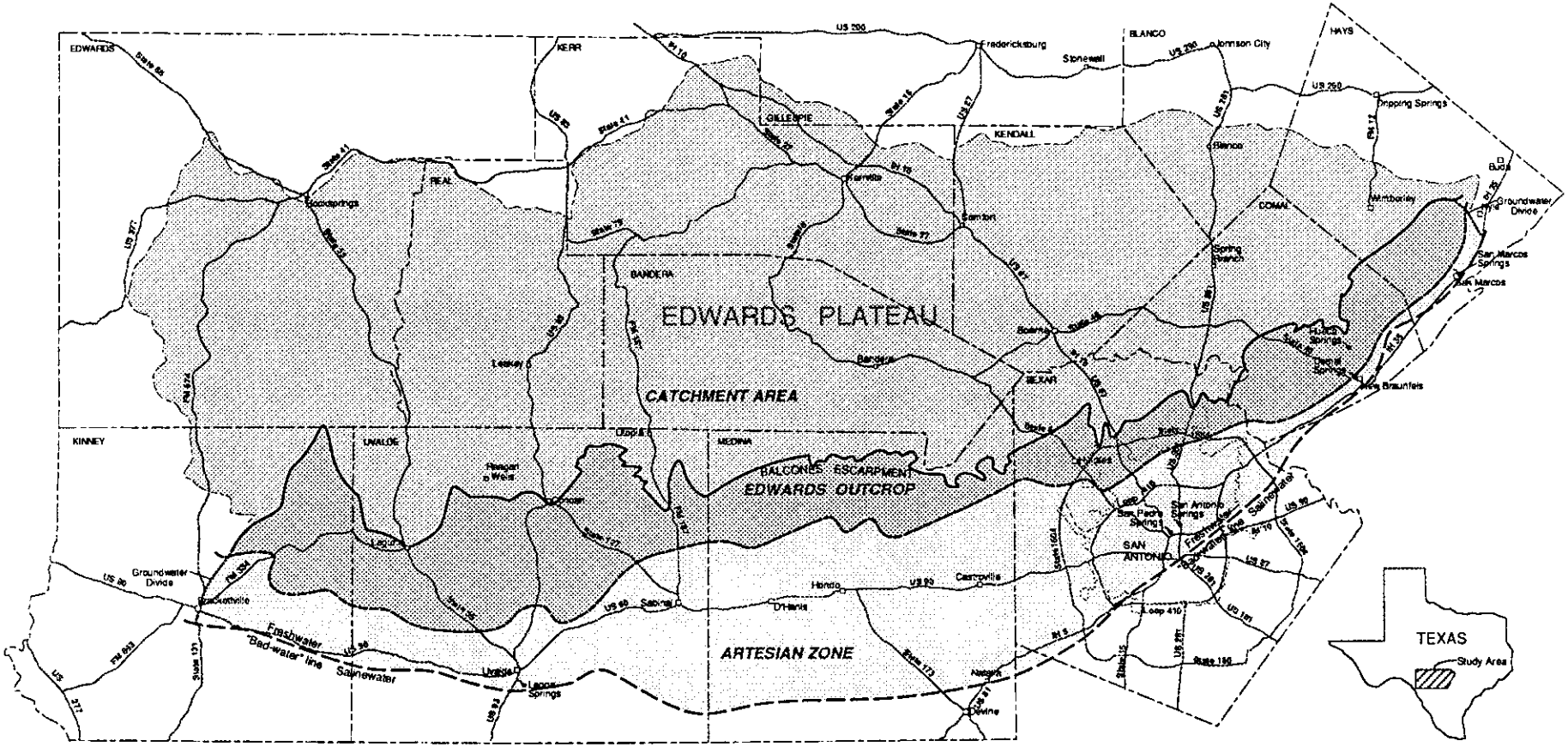


Figure 1. Physiographic location of the Edwards Aquifer (San Antonio Region)

discharge, which comprises 42.2% (1981-90) of the total withdrawal from the Edwards Aquifer. San Pedro and San Antonio Springs in San Antonio have become intermittent. Historically, the Comal Springs stopped flowing for two months in the summer of 1956. In 1984 and 1990, some of the higher Comal Springs ceased to flow and water levels in the index well (J17) dropped to within twelve feet of the 1956 record low.

Major problems facing the Edwards Aquifer are the threat of overdrafting the average annual recharge and maintaining natural springflows. Accordingly, the Edwards Underground Water District (EUWD) was authorized by the state legislature to develop, implement, and enforce a Drought Management Plan (DMP). The Cities of San Antonio, New Braunfels, and San Marcos have had to enforce Water Conservation Plan ordinances and water use disputes among aquifer users have become more frequent. In the most current case, the Sierra Club sued the Secretary of the Interior and the U.S. Fish and Wildlife Service (FWS) for failure to perform duties under the Endangered Species Act (ESA) and for injunctive relief. The U. S. District Court for the Western District of Texas reached a decision on January 30, 1993, and ordered FWS to determine required springflows, and the Texas Water Commission (TWC, now the Texas Natural Resource Conservation Commision or TNRCC) to prepare a plan assuring that springflows will not drop below jeopardy levels. The court threatened additional orders if the state legislature did not set up a regulatory system to limit withdrawals from the aquifer. In response, the 73rd Texas Legislature passed the Senate Bill No. 1477 to create the Edwards Aquifer Authority (EAA) and abolish the EUWD, effective September 1, 1993. Governed by an appointed Board of Directors, the authority's primary function is to regulate the aquifer pumpage by limiting the long-term annual withdrawal to 400,000 ac-ft. This amount is believed to be adequate to maintain the flows at the Comal and the San Marcos Springs, although the court opinion stated this level to be 200,000 ac-ft. Aquifer modeling is a part of the effort to improve the understanding of quantitative relationships among recharge, pumpage, spring-flow, and water levels. Such models will allow efficient and prudent management options to be explored without actually implementing a plan that would cause jeopardy to or loss of federally-listed endangered species.

Purpose and Scope of the Research

For the Edwards Aquifer to be successfully managed, an accurate, near real-time simulation model is needed in addition to institutional changes and new legislation. Existing models are too complicated and difficult to use in real-time management practices for pumpage control. The use of distributed parameter models such as a finite difference model require massive data input, calculate unwarranted intermediate steps, and generate a bulk of unnecessary output, and yet have not yielded satisfactorily accurate simulations.

The purpose of this research is to develop and test a lumped parameter model for simulating monthly water levels and springflows of the Edwards Aquifer. The goal of the new model is to keep it simple and easy to use without sacrificing simulation accuracy and to allow for advanced research using stochastic optimization in groundwater management. In addition, this study performs flow loss analysis to determine recharge functions for each river basin. These functions will provide a mechanism for the groundwater model to interact with surface runoff which has not been considered in previous models. Instead of taking recharge as a fixed input, the model will calculate recharge using streamflow input and allows recharge to vary with aquifer water levels.

The scope of the research is limited to the Edwards Aquifer, Balcones Fault Zone of the San Antonio Region. The data used for calibrating and verifying the model will consist of stream and spring flow data from the United States Geological Survey (USGS), well data from the Texas Natural Resources Information Systems (TNRIS), Texas Water Development Board (TWDB) and the Texas Water Commission (TWC).

Literature Review

Most of the conventional aquifer simulation models, such as PLASM (Prickett and Lonnquist, 1971), USGS-2D-FLOW (Trescott et al., 1976), and MODFLOW (McDonald and Harbaugh, 1988), were developed based on a distributed parameter finite-difference model. A list of other distributed parameter models using finite elements can also be found in van der Heijde et al. (1988). These distributed parameter models were developed based on the alluvial type aquifers, in which groundwater moves slowly through the small pores of a porous matrix. In contrast, groundwater in a karst aquifer, having the characteristics of channelization and

large cavities, flows rapidly through conduit networks similar to a water distribution system. Previous karst aquifer modeling has been done with the above mentioned finite-difference models. For example, Klemt, et al. (1978) and Maclay and Land (1988) modified PLASM and USGS-2D-FLOW, respectively, to develop planning models for the Edwards Aquifer. Despite using an annual time-step, they still required massive data input since the model contained over 800 active cells. Thorkildsen and McElhaney (1992) refined and calibrated the former model for use in evaluating management scenarios. The model had monthly time-steps and encountered some difficulties in calibration and verification.

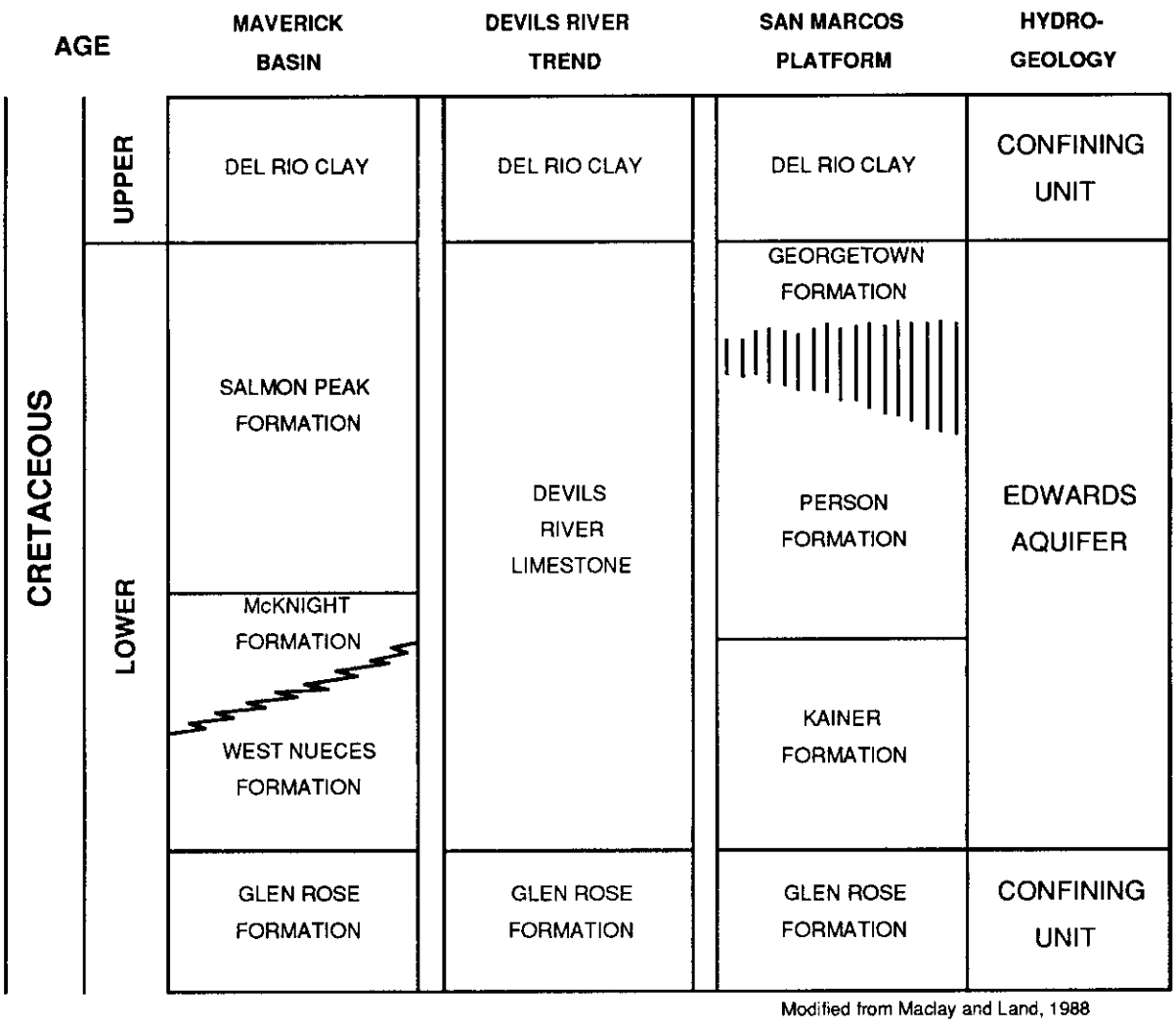
The source of recharge to the Edwards Aquifer is from stream flow loss and infiltration due to precipitation over the Edwards outcrop area. Methods of estimating natural recharge to the Edwards Aquifer were developed by the USGS (Garza, 1966; Puente, 1978) and HDR Engineering, Inc. (Choffel and Vaugh, 1993). Although both methods produced similar annual historic recharge estimates, they differed in spatial and temporal distributions. The principal difference in the two methods was in calculating runoff in the drainage area between the upper and lower stream gages. The USGS method assumed that precipitation-runoff conditions were the same for the catchment and recharge areas and calculated runoff from the intervening area based on the per-unit-area streamflow at the upper gage. When precipitation within the catchment and recharge areas differ by more than 20 percent, linear adjustments were made according to a precipitation ratio. Adjustments were also made for baseflows within each basin using baseflow recession curves developed by Puente (1975).

HDR calculated runoff in the intervening drainage area with a modified version of the Soil Conservation Service (SCS) runoff curve number method. The method accounted for spatial variation in precipitation by using the Thiessen Polygon procedure. Soil cover differences and water rights diversions were also accounted for in the HDR method. The HDR method indicated a greater contribution to total estimated recharge from the Guadalupe River Basin and less from the Nueces River Basin than that calculated by the USGS.

HYDROGEOLOGY

Table 1 shows the stratigraphic units of a typical Edwards Aquifer cross section. The aquifer is an association of Lower Cretaceous limestones overlying the Glen Rose Limestone and underlying the Del Rio Clay. The base of the aquifer is

Table 1. Stratigraphic units of the Edwards Aquifer



Modified from Maclay and Land, 1988

confined by the upper part of the Glen Rose Formation and in the artesian section, the top of the aquifer is confined by the Del Rio Formation. The lateral boundaries of the Edwards Aquifer consist of groundwater divides on the east and west near Kyle in Hays County and near Brackettville in Kinney County, respectively. The aquifer is bounded on the south by the "bad water line", (line marking water with more than 1000 mg/l of total dissolved solids), and on the north by the northern most edge of the Balcones Fault Zone.

The Edwards Group and associated limestones were deposited as shallow marine platform carbonates consisting of reef and deep marine lithofacies in the western portion of the aquifer region. Most of the Edwards Group and associated limestones within the San Marcos Platform and the Devils River Reef Trend were

subjected to subaerial exposure after deposition which enhanced the development of secondary porosity (Maclay and Land, 1988). Enhanced porosity has also occurred along the high angle faults and fractures of the Balcones Fault Zone.

The Edwards Aquifer receives water primarily from streams and rivers originating from the catchment areas on the Edwards Plateau (Figure 2). Streams and rivers that cross the outcrop of the Edwards Aquifer lose major portions of their flow to the aquifer through joints, faults, and sink holes. There are three river basins that cross the aquifer area: the Nueces, the San Antonio, and the Guadalupe River. Extending from the west, the Nueces River Basin covers over a half of the aquifer area. Several major tributaries in the basin traverse the aquifer recharge zone including the Nueces; the West Nueces; the Frio; the Dry Frio; and the Sabinal Rivers; as well as the Seco and the Hondo Creeks. The portion of the San Antonio River Basin that is located in the recharge zone extends from the Medina River to the Cibolo Creek and includes headwaters of the Helotes, Leon, and Salado Creeks. Only a small portion of the Guadalupe River Basin intersects the eastern aquifer area. However, two of the basin tributaries, the Comal and San Marcos Rivers, are primarily fed by the aquifer at the Comal and San Marcos Springs. The tributaries crossing the recharge area include the Guadalupe River, the Blanco River, and tributary headwaters of the Comal and San Marcos Rivers. Small creeks in the headwaters are the Dry Comal, Alligator, York, Purgatory, and Sink Creeks.

Based on the USGS estimates, the aquifer has an average annual recharge of 651,700 ac-ft with about 58.5% contributed by the Nueces River Basin. Generally, the water flows south-southeastward from the recharge zone under steep hydraulic gradients and low permeabilities within the unconfined portion of the aquifer. As the water flows into the confined portion of the aquifer, the flow direction changes toward the east and northeast within the low gradient, highly permeable grabens in the artesian zone. The water then discharges from several springs — mostly the Comal and San Marcos Springs — which account for 355,500 ac-ft annually. The two major springs contribute about 25% of the flow in the Guadalupe River downstream. The contribution was about 66% during the drought year of 1956.

MODEL DEVELOPMENT

A first step in the modeling process involves selecting the type of system to be represented. A *lumped parameter* model considers mixed effects of the system

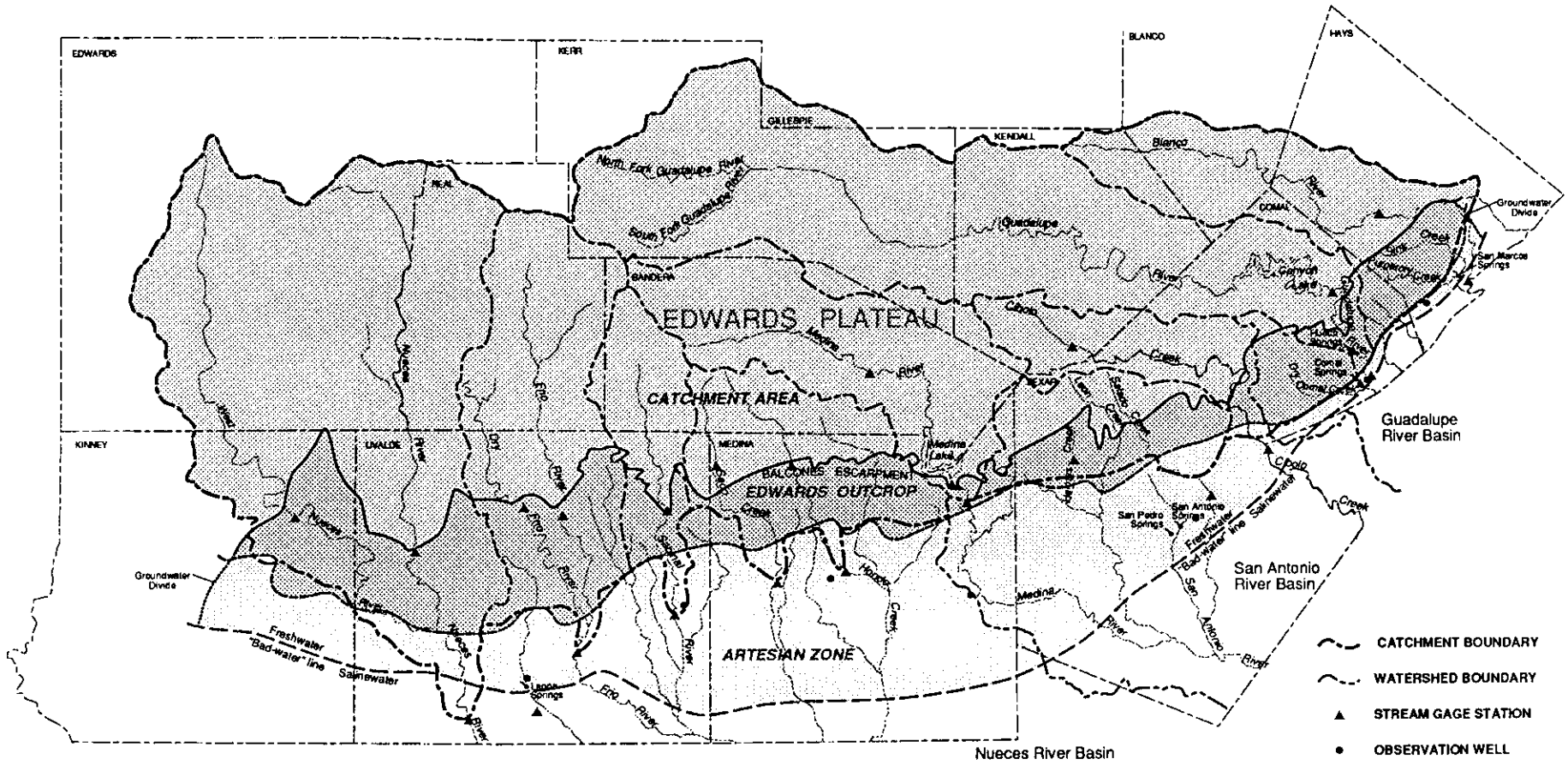


Figure 2. Streams and river basins that cross the Edwards Aquifer recharge zone

composed of ideal elements having uniform characteristics. In contrast, a model represented by a heterogeneous system consisting of infinitely small elements is called a *distributed parameter* model. Thus, it is expressed by partial rather than ordinary differential equations. A system is said to be *linear* if the system output can be produced by superposition. Furthermore, a system whose parameters change with time is called *nonstationary*, otherwise it is called *stationary*.

The following sections will discuss in some detail the development of a lumped parameter model for the Edwards Aquifer. The model is formulated based on the state-space point of view in control system theory (Takahashi, et al., 1972). A discrete, nonlinear, nonstationary system is used to conceptualize the physical system of the Edwards Aquifer, which consists of a series of connected leaking tanks. The model is focused on simulating monthly water levels at key observation wells in eight river sub-basins and two springflows at Comal and San Marcos. Built-in recharge functions for each sub-basin are also developed to estimate monthly recharge from streamflow input. They allow for the interaction between ground-water and surface water to occur.

First, the basic differential equations and their solutions will be presented. Then, the tank model and the Edwards conceptual model will be discussed. Finally, the solution algorithm and computer program will also be described.

Approach

Differential equations for lumped parameter models and their solutions are presented below. Their derivations can be found in Takahashi, et al. (1972).

The first order ordinary differential equation (ODE) for a linear stationary system can be represented as:

$$\frac{d}{dt}x(t) = Ax(t) + Bu(t) \tag{1}$$

where u is the control input, x is the system state, and A and B are system parameters. The solution of the above equation for a general u(t) is

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-\tau)}Bu(\tau)d\tau\,, \tag{2}$$

where x_0 is the initial value of x(t), and τ is a dummy variable of integration. The first term of the solution (2) is called the *free response* as opposed to the *forced*

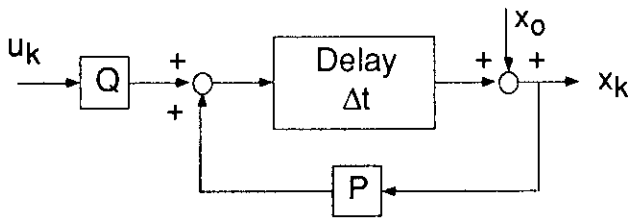


Figure 3. Graphical representation of a first-order, linear, stationary system

response for the second term. For a discrete system with Δt time interval, one can apply the above solution for a period from $k\Delta t$ to $(k+1)\Delta t$ and solve

$$x_{k+1} = x[(k+1)\Delta t] = e^{A\Delta t} x_k + \int_{k\Delta t}^{(k+1)\Delta t} e^{A[(k+1)\Delta t-\tau]} B u_k d\tau$$

$$= e^{A\Delta t} x_k + (e^{A\Delta t} - 1) A^{-1} B u_k$$

$$= P x_k + Q u_k , \qquad (3)$$

where $P = e^{A\Delta t}$ and $Q = (P-1)A^{-1}B$. The computational scheme following the above equation is graphically shown in Figure 3. The response x_k at time $t = k\Delta t$ can be computed by repeatedly applying of equation (3) resulting in

$$x_k = P^k x_o + \sum_{i=0}^{k-1} P^{k-1-i} Q u_i \qquad (4)$$

For a higher order system, the solution can be generalized with the aid of vector calculus. Considering an nth order system of n state variables $(x_1, x_2, \ldots, x_n)$ with r inputs $(u_1, u_2, \ldots, u_r)$, the scalar equation (1) becomes a vector differential equation

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \qquad (5)$$

where $\mathbf{x}$ and $\mathbf{u}$ are vectors of state and input variables, respectively. $\mathbf{A}$ and $\mathbf{B}$ are constant parameter matrices with dimensions of $n \times n$ and $n \times r$, respectively. The matrix difference equation equivalent to (3) can be expressed as

$$\mathbf{x}_{k+1} = e^{\mathbf{A}\Delta t}\mathbf{x}_k + e^{\mathbf{A}\Delta t}\int_0^{\Delta t} e^{-\mathbf{A}\tau}\mathbf{B}\mathbf{u}(\tau)d\tau$$

$$= e^{\mathbf{A}\Delta t}\mathbf{x}_k + e^{\mathbf{A}\Delta t}(\mathbf{I} - e^{-\mathbf{A}\Delta t})\mathbf{A}^{-1}\mathbf{B}\mathbf{u}_k$$

$$= e^{\mathbf{A}\Delta t}\mathbf{x}_k + (e^{\mathbf{A}\Delta t} - \mathbf{I})\mathbf{A}^{-1}\mathbf{B}\mathbf{u}_k$$

$$= \mathbf{P}\mathbf{x}_k + \mathbf{Q}\mathbf{u}_k , \qquad (6)$$

where,

$$\mathbf{P} = e^{\mathbf{A}\Delta t}, \qquad \mathbf{Q} = (\mathbf{P} - \mathbf{I})\mathbf{A}^{-1}\mathbf{B}, \qquad (7)$$

The exponential matrix $e^{\mathbf{A}\Delta t}$ is also called the state-transition matrix and has an $n \times n$ dimension for an $n \times n$ matrix **A**. The matrix can be evaluated using the following series expansion,

$$e^{\mathbf{A}\Delta t} = \mathbf{I} + \mathbf{A}\Delta t + \frac{1}{2!}\mathbf{A}^2(\Delta t)^2 + \frac{1}{3!}\mathbf{A}^3(\Delta t)^3 + \ldots \qquad (8)$$

Equation (6) is applied repeatedly starting from $\mathbf{x}_0$ and the result can be written similar to (4)

$$\mathbf{x}_k = \mathbf{P}^k\mathbf{x}_0 + \sum_{i=0}^{k-1}\mathbf{P}^{k-1-i}\mathbf{Q}\mathbf{u}_i \qquad (9)$$

Lumped Parameter Model

Attention is turned to the derivation of the governing differential equation by the aid of a conceptual tank model. The modeling concept is similar to the hydrologic routings of river, watershed, and reservoir hydrographs used by surface water hydrologists. The lumped parameter conceptual model of the surface and ground water flow in the area below the upper gages of the Nueces River Sub-Basin is shown in Figure 4. The source or sink $u_1(t)$, is the combined flow of the upper gages less pumpage and flow at the lower gage of a rock-filled tank with storage and transmissivity related parameters S_1 and T_1, respectively. The output $y_1(t)$ is the underflow to the nearby system, the Frio River Sub-Basin. The state variable $x_1(t)$ is the flow-driven potential, which can be expressed as the water level in an

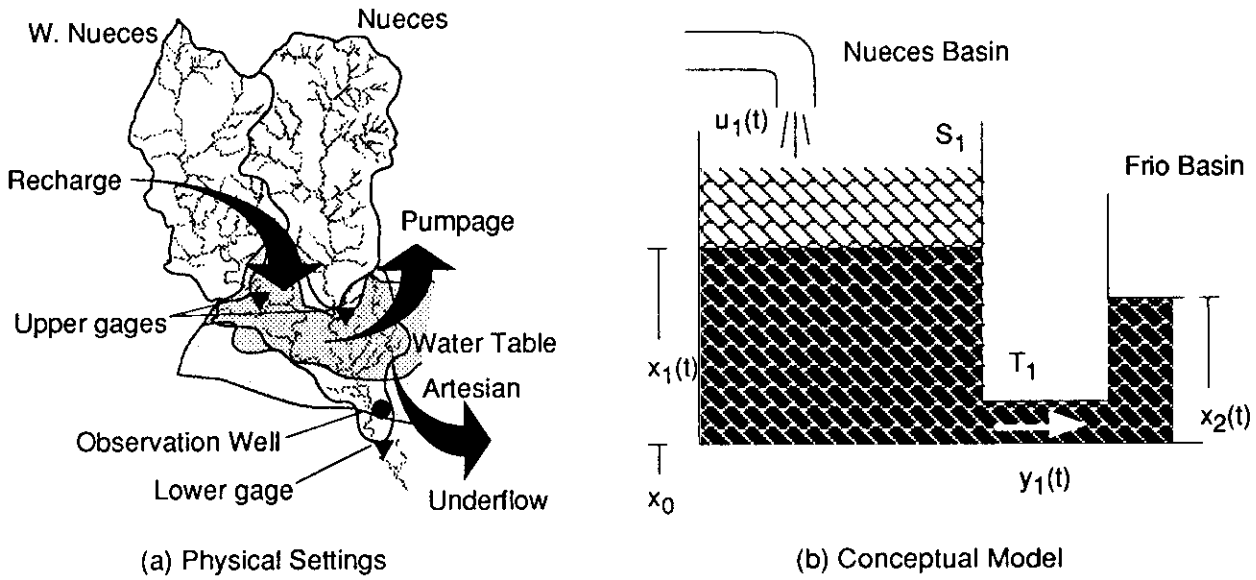


Figure 4. Recharge and water level relationships for the Edwards Aquifer in the Nueces River Basin and underflow to the Frio River Basin

observation well near the lower gage. S_1 and T_1 are conceived as aggregate or lumped parameters for surface and groundwater flow in the intermediate area between the upper and the lower gages. Using the lumped parameter concept, the aquifer portion under the Nueces can be considered as an elementary control volume of water in a homogeneous medium as opposed to multiple cells in a finite difference distributed parameter model. The flow regime becomes isotropic and can be described by a single continuity, or mass balance equation. The resulting system differential equation describing groundwater flow under the Nueces River Sub-Basin can be derived as follows:

$$u_1(t) - y_1(t) = \frac{d\forall}{dt}, \tag{10}$$

where $\forall$ is the volume of water under the sub-basin and can be expressed as a function of x_1:

$$\forall(x_1) = S_1'(x_1 - x_0)^n, \tag{11}$$

where S_1' is the storage constant related to aquifer porosity, the exponent n is related to the volumetric geometry, and x_0 is the aquifer base elevation for the sub-basin. Equation (11) can then be expressed as:

$$\frac{d\forall}{dt} = S_1'n(x_1-x_0)^{n-1}\frac{dx_1}{dt} = S_1(x_1)\frac{dx_1}{dt}, \tag{12}$$

where the storage related parameter $S_1(x_1) = S_1'n(x_1-x_0)^{n-1}$. For a linear storage model, i.e. n = 1 in equation (11), S_1 becomes a constant and is equal to S_1'.

Applying Darcy's Law, the underflow, $y_1(t)$ may be expressed as:

$$y_1(t) = K\frac{(x_1-x_2)}{L}A, \tag{13}$$

where; K is the hydraulic conductivity, $(x_1-x_2)/L$ is the hydraulic gradient, A is the cross-sectional area of the flow path, L is the representative distance of the flow path, and x_2 is the potential in the adjacent sub-basin, (the Frio River Basin). If w is the average cross-sectional width along the flow path, then by substitution, equation (13) becomes:

$$y_1(t) = K\frac{(x_1-x_2)}{L}(x_1-x_0)w = T_1'\frac{(x_1-x_2)}{L}w = T_1(x_1-x_2), \tag{14}$$

where $T_1'(x_1) = K(x_1-x_0)$ is the transmissivity, (x_1-x_0) is the average aquifer thickness, and $T_1 = T_1'w/L$ is the aggregate transmissivity related parameter which depends on a flow path geometry. For an artesian condition, the aquifer thickness becomes a constant independent of x_1 and so does T_1'. By substituting equations (12) and (14) back into equation (10), the mass balance equation can be expressed as:

$$u_1(t)-T_1(x_1-x_2) = S_1(x_1)\frac{dx_1}{dt}, \tag{15}$$

which can be rearranged to the canonical form (1) as:

$$\frac{dx_1}{dt} = \frac{T_1}{S_1}x_1(t) + \frac{1}{S_1}u_1(t) \tag{16}$$

The Edwards Aquifer Conceptual Model

Watersheds of the streams that cross the aquifer recharge zone can be divided into nine sub-basins. Although, The Edwards Underground Water District (Wokhour,1993) suggests that the Guadalupe River Basin may recharge the aquifer during low water levels, data was not available to us during our research to confirm this recharge. Therefore, recharge from the Guadalupe River Basin was not considered in the lumped parameter model and only eight sub-basins losing their

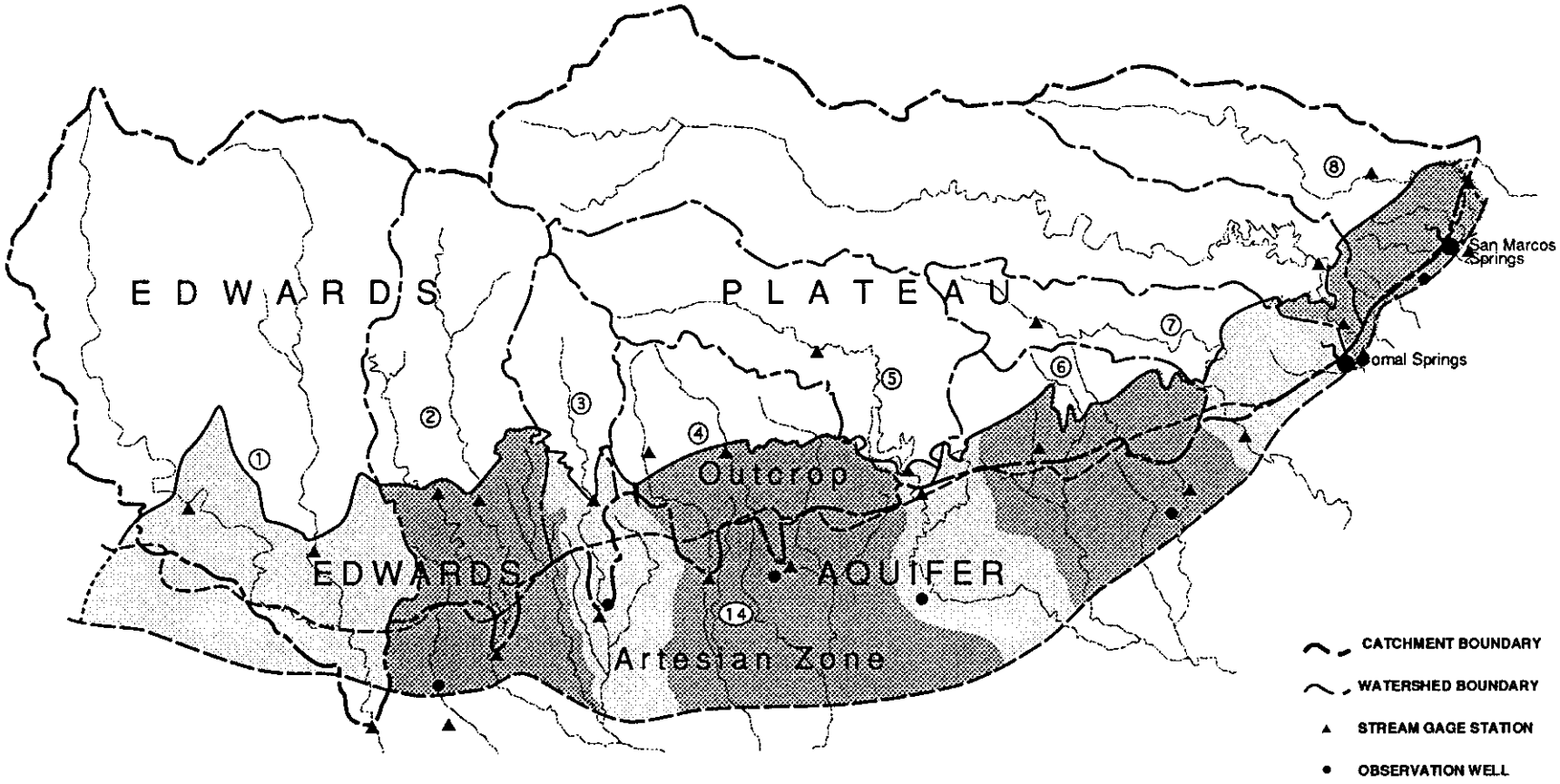


Figure 5a. Delineation of sub-basins for the Edwards Aquifer

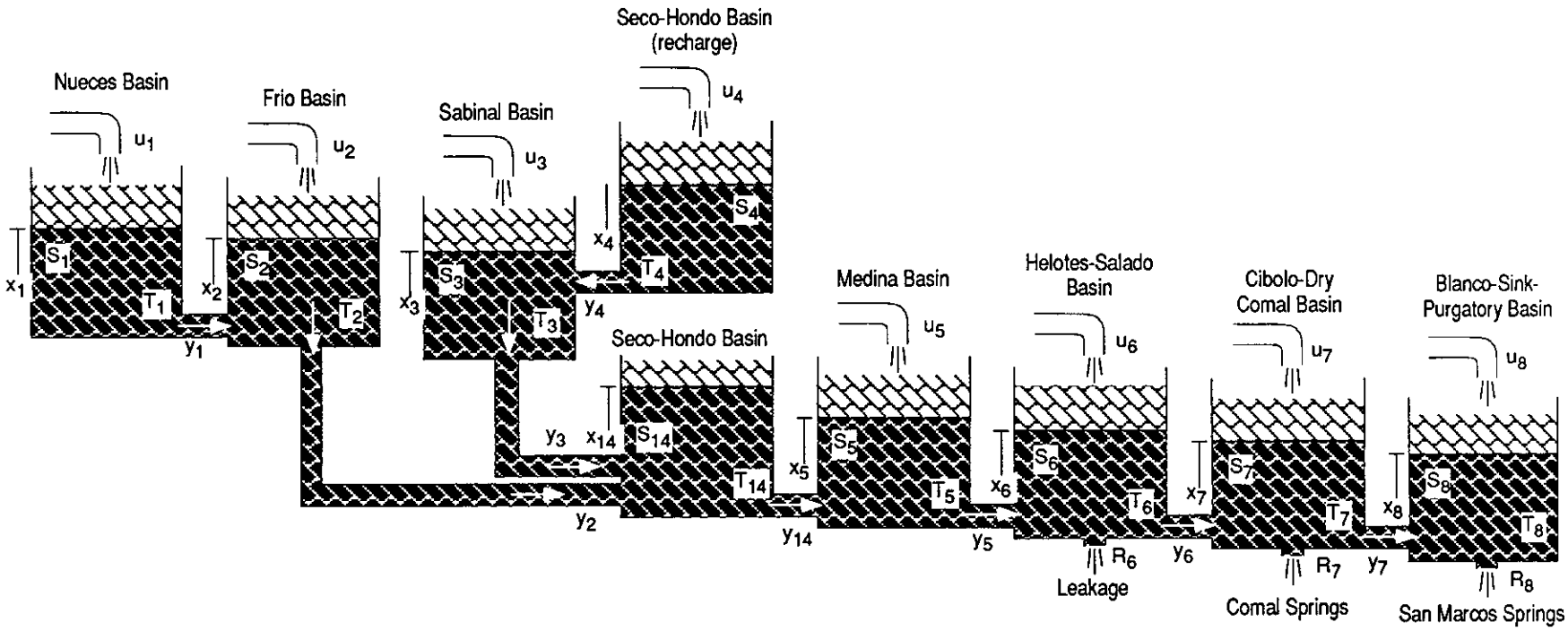


Figure 5b. The conceptual tank model for the Edwards Aquifer

Table 2 Lists of sub-basins and their associated observation wells

Sub-Basin	Upper Gage	Lower Gage	Observation Well
1 Nueces River Basin	08190500 08190000	08192000	YP 69-50-302 (H-5-1)†
2 Frio River Basin	08196000 08195000	08197500	YP 69-43-804
3 Sabinal River Basin	08198000	08198500	YP 69-45-401 (I-4-4)
4 Seco-Hondo Creek Basin-Upper	08201500 08200000	08202700 08200700	
14 Seco-Hondo Creek Basin-Lower	N/A	N/A	TD 69-47-302 (I-3-148)
5 Medina River Basin	08179500	N/A	TD 68-41-301 (J-1-82)
6 Helotes-Salado Creek Basin	08181400	08178700	AY 68-37-203 (J-17)
7 Cibolo-Dry Comal Creek Basin	08183900	08185000	DX 68-23-302 (G-49)
8 Alligator-York-Purgatory-Sink Creek-Blanco River Basin	08171000	08171300	LR 67-09-110

† Old well number

water as recharge to the aquifer were conceptualized. Figure 5a delineates those sub-basins with their labels, associated gaging stations, and representative observation wells listed in Table 2. All basins have the same general direction of groundwater flow, i.e. flowing from the recharge area to the underground outlet. Due to effects of the Knippa Gap on the flow pattern, the Seco-Hondo Creek Basin is subdivided into upper and lower sub-basins. The upper basin receives all the basin recharge while the lower basin (14) serves as a main supply line where the groundwater is transferred and pumped.

Each basin is conceptually represented by a tank in which the flow can be described by equation (16). Thus, the Edwards Aquifer conceptual model can be portrayed as a series of connected tanks assembled according to the aquifer general flow pattern. The current configuration, as shown in Figure 5b, considers only two major springs and one area of leakage. The system of ODE's describing the model has the form of equation (5) with their parameters derived following equations (10)-

(15). Flows at Comal and San Marcos Springs are determined using linear relationships between springflows and water levels in the nearby observation wells.

Computer Model Algorithm

There are two primary parameters in the tank model, namely the storage and the transmissivity related parameters. The former parameter is associated with the tank properties while the latter is dependent on the link characteristics. A tank can have more than one link to represent boundary conditions, leakage, and springflow in addition to the underflow to the nearby sub-basin. Consider the ODE similar to equation (16) for the middle leaky tank in a series of three connected tanks as shown in Figure 6,

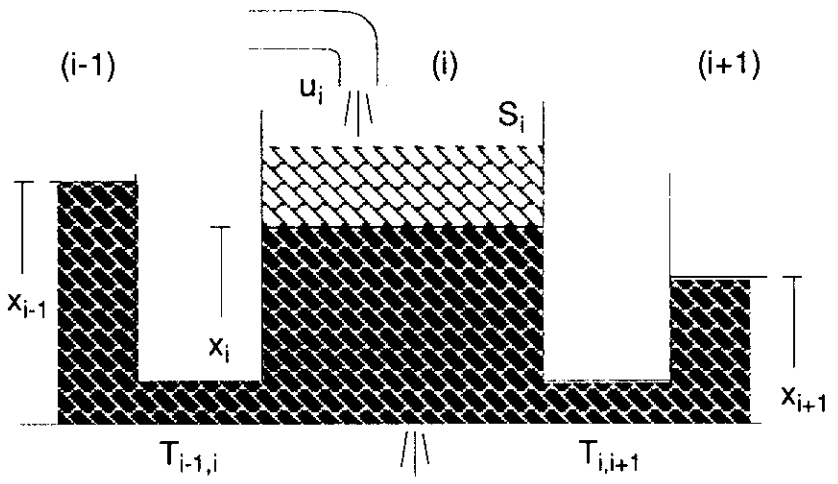


Figure 6 A tank model with multiple links as boundary conditions, springs, and leakage

$$\frac{d}{dt} x_i = \frac{1}{S_i}\left[(x_{i-1}-x_i)\, T_{i-1,i} + (x_{i+1}-x_i)\, T_{i+1,i} + (x_L-x_i)\, T_L\right] + \frac{1}{S_i} u_i\,, \qquad (17)$$

where i-1,i, and i+1 are the indices for the upstream, the middle, and the downstream tanks, respectively, (i-1,i) and (i,i+1) are the link indices, and L is the leakage index. If link (i-1,i) is a boundary, one can set $x_{i-1} = \alpha_i + \beta_i x_i$ for a mixed boundary condition while $\alpha_i = 0$ and $\beta_i = 0$ gives constant flux and constant head boundary conditions, respectively. If link (i,i+1) represents springs, the relationship between springflow and water level can be used to determine flow through the link.

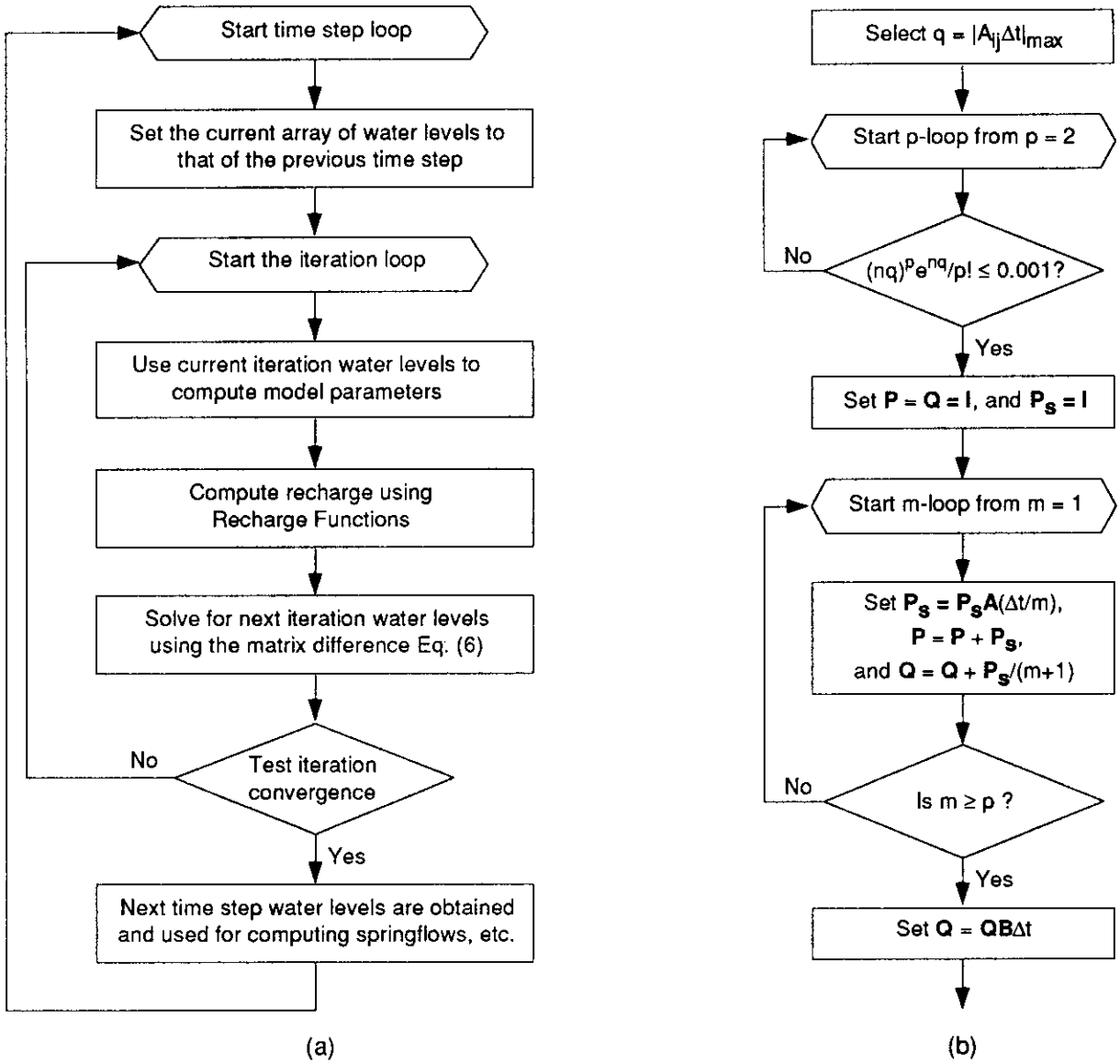


Figure 7. Flow charts for (a) iterative procedure for solving a nonlinear system, (b) Paynter's algorithm for obtaining **P** and **Q**

For a linear relationship, e.g. springflow = $\alpha_i + \beta_i x_i$, the second term in the brackets of equation (17) should be replaced by $\alpha_i + \beta_i x_i$. Similarly, leakage can be treated as a link, and x_L and T_L become constants representing the adjacent aquifer water levels and the leakage parameter, respectively. Here, the leakage parameter is defined as the product of the hydraulic conductivity of the aquitard and the leakage area divided by the aquitard thickness.

Since the model parameters are a function of state variables or water levels, the resulting model governing equations are nonlinear. Therefore, the solution

given in equation (9) for the linear parameter model can not be applied directly. An iterative procedure based on the linear solution is devised to solve the nonlinear model as shown on Figure 7. At the beginning of the iteration process, the model parameters are computed using the previous time-step water levels and the solution equation (6) is applied. As iterations proceed, the parameters are recomputed using water levels obtained from the previous iteration. For the storage parameter function S(x), its value is evaluated using the time-average water levels. However, the aquifer thickness used for evaluating the transmissivity parameter function T(x) is the spatial and temporal average value between the linked basins over the consecutive time-steps. The iteration process is converged when the sum of changes in water levels between two consecutive iterations is less than the preset tolerant limit. Then, water levels for the next time-step are obtained and used for computing other output, such as springflows, leakage, and basin underflows. The computation advances to the next time-step and the iteration process is repeated.

To solve equation (6), the matrix exponent equation (8) as well as system parameter **P** and **Q** must be evaluated analytically or numerically. We chose a numerical solution using the algorithm suggested by Paynter (see Takahashi, et al., 1972). The algorithm shown in Figure 7b is simply a series expansion of equation (8) to the p^{th} term, in which p can be determined from the following relationship,

$$\frac{1}{p!}(nq)^p e^{nq} = 0.001, \tag{18}$$

where n is the order of the system, $q = \max|A_{ij}\Delta t|$, and A_{ij} is an element of the **A** matrix.

Flow loss and Recharge Function

There are two existing methods for calculating recharge to the Edwards Aquifer, i.e. the methods used by the USGS and HDR Engineering, Inc. Both methods are inappropriate to implement in the current modeling for several reasons. First, they require lower gage flow data which are unobtainable during predictive simulation. Second, in addition to streamflow data, they both use rainfall data which the current model intends to avoid to keep the model input simple. They also do not consider the influence of groundwater levels on recharge capacities. Despite their complexity, the methods have not proved advantageous over a less complex method as will be discussed in the following.

To be able to project recharge for predictive simulation, a relationship between estimated losses and the upper gage flows for each sub-basin must be established from a flow loss analysis. In some cases, the relationship must also account for the effect of groundwater level, especially for the sub-basins with shallow water tables. This relationship, which is referred to as a recharge function, is used by the model to calculate the sub-basin recharge given a streamflow at the upper gage and a groundwater level. The function is empirically developed based on the regression analysis of estimated losses, observation well data, and the surface inflow above the lower gage computed from the gaged flow data. In sub-basins where stream gage data are not available, inflows are determined by interpolating from adjacent river basins.

The estimated losses are obtained by simplifying the USGS method of recharge calculation. Per-unit-area runoffs from the intermediate drainage area between the upper and lower gages and from the drainage area above the upper gage are assumed to be the same. The water balance for the river stretch crossing the recharge zone can be expressed as:

$$\text{Loss} = Q_U - Q_L + Q_I ; \qquad Q_I = Q_U \frac{A_I}{A_U}, \tag{19}$$

where Q_U and Q_L are the monthly streamflows at the upper and lower gages, respectively. Q_I is the surface runoff in the drainage area between gages, and A_U and A_I are drainage areas above the upper gage and between upper and lower stream gages. It is more convenient to express a loss in terms of a Loss Ratio (LR), which is defined as the loss per the total surface inflow above the lower stream gage or,

$$LR = \frac{\text{Loss}}{\text{Inflow}} = 1 - \frac{Q_L}{Q_U + Q_I} \tag{20}$$

The ratios are then plotted against monthly inflows from the upper gage and average water levels from previous months. Figures 8-17 show the plots of each sub-basin except the Medina where recharge is determined based on the Medina reservoir content. Since only the Nueces and Sabinal Sub-Basins show the effects of water levels on recharge, the plots of LR versus water level for other sub-basins are not depicted. Recharge functions for each sub-basin are determined by analyzing the plots using linear regression models. Results of analyses are given below, in

which the loss, inflow, and recharge capacity (RC) are given in thousand ac-ft and the water level (WL) is in feet above the mean sea level (MSL).

Nueces River Basin

The aquifer under this basin is shallow. Figure 8 illustrates a line where losses equal to the capacity. As expected, the plot in Figure 9 reveals a close relationship between LR and water level. The basin losses are also believed to be controlled by a recharge capacity, a physical limitation of the recharge features. In general, the recharge function can be expressed as follows:

$$\text{Loss} = \min(\text{RC}, \text{Inflow} \times \text{LR}) \tag{21}$$

$$0 \le \text{LR} = 1 - \left[\frac{k}{(H_m - WL)}\right]^m \le 1 \tag{22}$$

where constants k = 37.5902 ft; m = 3.6814; H_m = 927.65 ft MSL; and the recharge capacity, RC = 29.0 × 10^3 ac-ft.

Frio River Basin

The correlation of LR and inflow can be observed in the semi-log plot of Figure 10. However, the plot in Figure 11 shows no significant correlation between LR and water level even though the basin is in the shallow aquifer region. This may partly be due to the poor water level records used in the analysis. The recharge function for this basin can be given as

$$\text{Loss} = \text{Inflow} \times \text{LR} \tag{23}$$

$$0 \le \text{LR} = \frac{\ln(\text{Inflow}) - b}{a} \le 1 \tag{24}$$

where parameters a and b are -1.72581 and 4.95753, respectively.

Sabinal River Basin

The LR-inflow plot (Figure 12) seems to follow the similar pattern of the one in the Frio River Basin. Losses in this basin show a weaker correlation to groundwater level (Figure 13) than those in the Nueces. The recharge is determined from both water levels and inflows by the following relationships

$$\text{Loss} = \text{Inflow} \times \text{LR} \tag{25}$$

$$0 \leq \text{LR}' = \frac{\ln(\text{Inflow}) - b}{a} \leq 1 \tag{26}$$

$$\text{LR} = \begin{cases} 1 - \dfrac{k}{(H_m - \text{WL})} & \text{if LR}' > 0.9 \text{ and WL} < 824 \\ \text{LR}' & \text{else} \end{cases} \tag{27}$$

where parameters a and b are -2.06389 and 3.63713, respectively, and constants k = 4.67863 ft; and H_m = 870 ft MSL.

Seco-Hondo Creeks Basin

The loss function in this basin is presumed to be similar to the Frio River Basin. Based on the plot in Figure 14, the recharge function can be given as:

$$\text{Loss} = \text{Inflow} \times \text{LR} \tag{28}$$

$$0 \leq \text{LR} = \frac{\ln(\text{Inflow}) - b}{a} \leq 1 \tag{29}$$

where parameters a and b are -2.77614 and 5.36413, respectively.

Helotes-Salado Creek Basin

The only available gaging stations are the upper gage on Helotes Creek and the lower gage on Salado Creek. Interpolation of runoff from the nearby streams were attempted to obtain adequate flow data for stream loss analysis. The procedure did not provide good recharge estimates as the plot of LR versus Inflow failed to show correlation (Figure 15). Thus, the recharge function is formulated assuming that the basin inflow losses are the same rate as losses of the nearby Cibolo Creek Basin. Hence

$$\text{Loss} = \text{Inflow} \times \text{LR}_{\text{cibolo}} \tag{30}$$

Cibolo-Dry Comal Creek Basin

The recharge function for the basin is determined from losses in the Cobolo and Cry Comal Creeks at different loss rates. Runoff in the Dry Comal Creek drainage area is assumed to lose all water to the aquifer up to its recharge capacity.

The LR in the Cibolo Creek is derived from the plot (Figure 16) using linear regression on the semi-log scale. The basin recharge function becomes:

$$\text{Loss} = [\text{Inflow} \times \text{LR}]_{\text{cibolo}} + \min [\text{RC}, \text{Inflow}]_{\text{drycomal}} \tag{31}$$

$$0 \leq \text{LR}_{\text{cibolo}} = \frac{\ln[\text{Inflow}]_{\text{cibolo}} - b}{a} \leq 1 \tag{32}$$

where parameters a and b are -1.84034 and 4.44984, respectively.

Blanco River Basin

The LR for the Blanco River Basin showed a strong presence of the recharge capacity. All data points with inflows greater than the RC are fallen along the line LR = RC/Inflow, as shown in Figure 17. A weak dependency of the LR to groundwater level is also observed. However, the analysis was unable to provide a concrete functional relationship. Hence, the recharge function for the basin can be given simply as:

$$\text{Loss} = \min [\text{RC}, \text{Inflow}] \tag{33}$$

where RC = 1.331×10^3 ac-ft.

Medina River Basin

Recharge from the basin largely comes from seepage losses of the Medina Lake. The procedure used to estimate the losses follows Lowry (1955) who constructed correlation curves between reservoir seepage losses and the monthly average reservoir contents. The procedure is simplified by replacing the curves with the following function:

$$\text{Loss} = \frac{b}{\left[\frac{a}{V - y_p} - 1\right]^n} + x_p \tag{34}$$

where V is the reservoir content in thousand ac-ft, parameters a and y_p are 263.8 and 1.533 thousand ac-ft, respectively, and constants b = 2.99645 and n = 0.29455. The loss constant x_p is equal to 1.0 and 3.38084 thousand ac-ft for falling and rising of reservoir stages, respectively. The curves are plotted as shown in Figure 18.

Adjacent Areas

Additional recharge from adjacent areas of the above sub-basins is calculated from the recharge of those basins. Except the area adjacent to the Blanco River Basin, all losses from other adjacent areas are assumed to be proportional to their corresponding basin losses. Losses from the area adjacent to the Blanco River Basin is thought to be controlled by the recharge capacity at the same magnitude as Dry Comal Creek.

Figure 19 compares the estimated recharge by the USGS and estimated recharge using the recharge functions developed in this study.

Pumpage Estimation

Monthly pumpage data used in the recent TWDB simulation model was adjusted for each sub-basin. The TWDB simulation cell data was grouped into sub-basins by encoding a basin cell map. The monthly data was available only from 1978 to 1989. Monthly pumpage data by basin outside this period was estimated using USGS annual county pumpage data and pumpage distribution coefficients derived for each basin. First, annual county pumpage data was converted into annual basin pumpage data according to the percentage of the surface area of the basin in each county. Then, the annual pumpage for each basin was multiplied by the pumpage distribution coefficients to estimate monthly pumpage within each basin for the periods of missing pumpage data. The pumpage distribution coefficients are the average ratios of monthly to annual pumpage within each basin computed from adjusted TWDB data shown in Figures 20 to 27. Note that irrigation pumpage in the western basins are season dominated in contrast to municipal pumpage in the eastern basins.

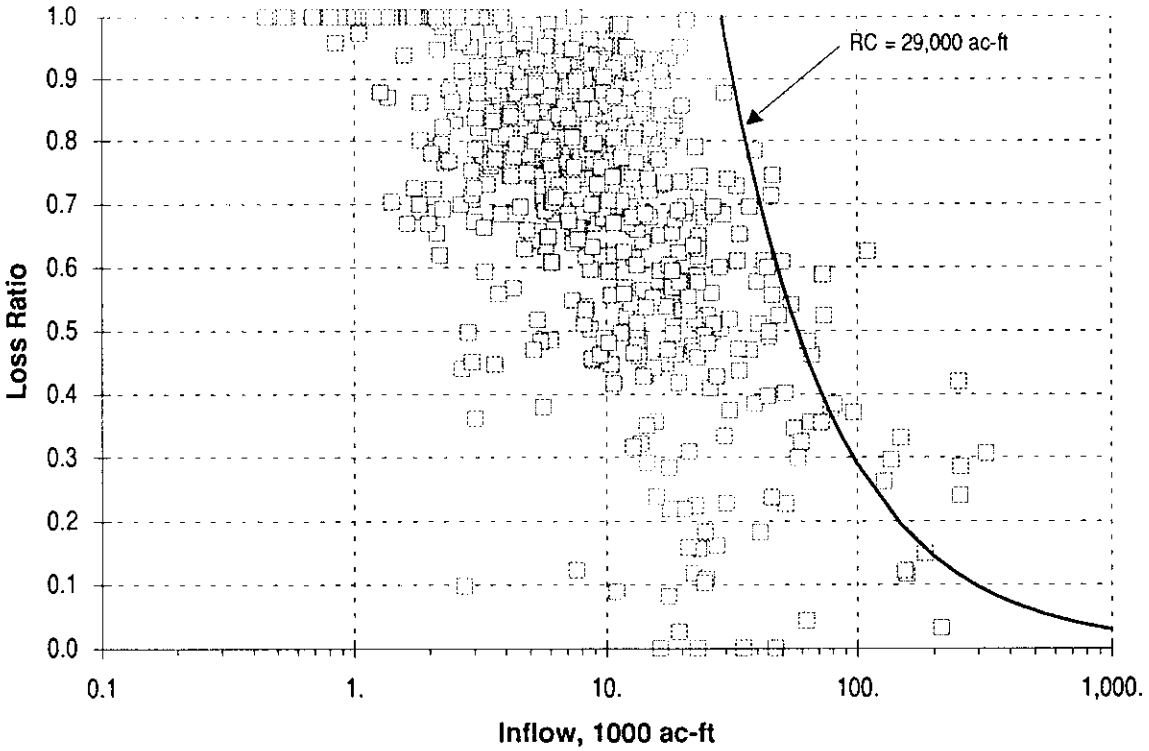


Figure 8. A plot of loss ratio versus inflow in the Nueces River Basin

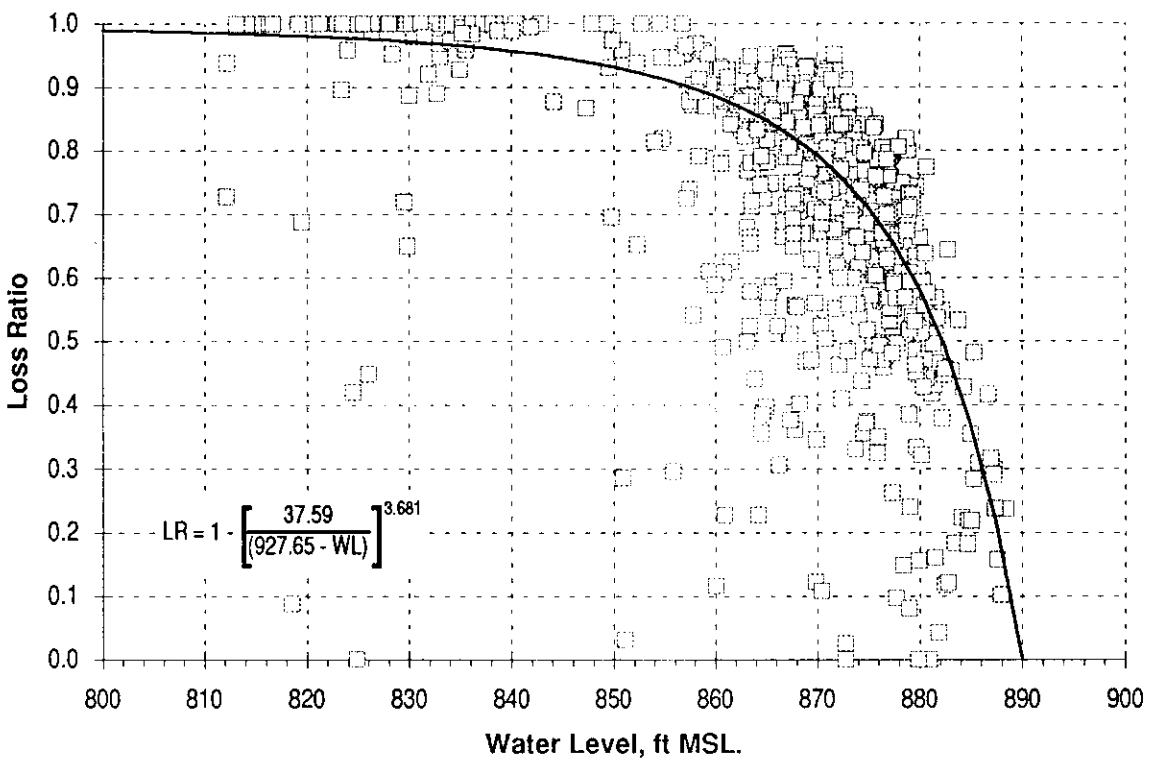


Figure 9. A plot of loss ratio versus water level in the Nueces River Basin

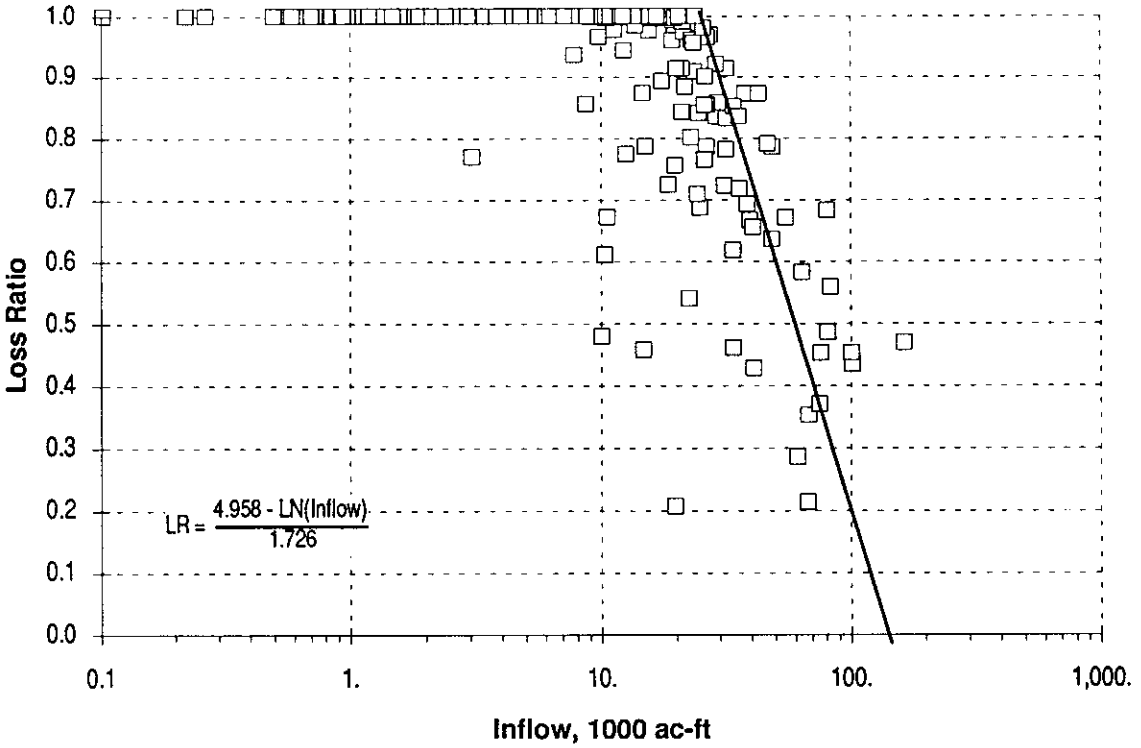


Figure 10. A plot of loss ratio versus inflow in the Frio River Basin

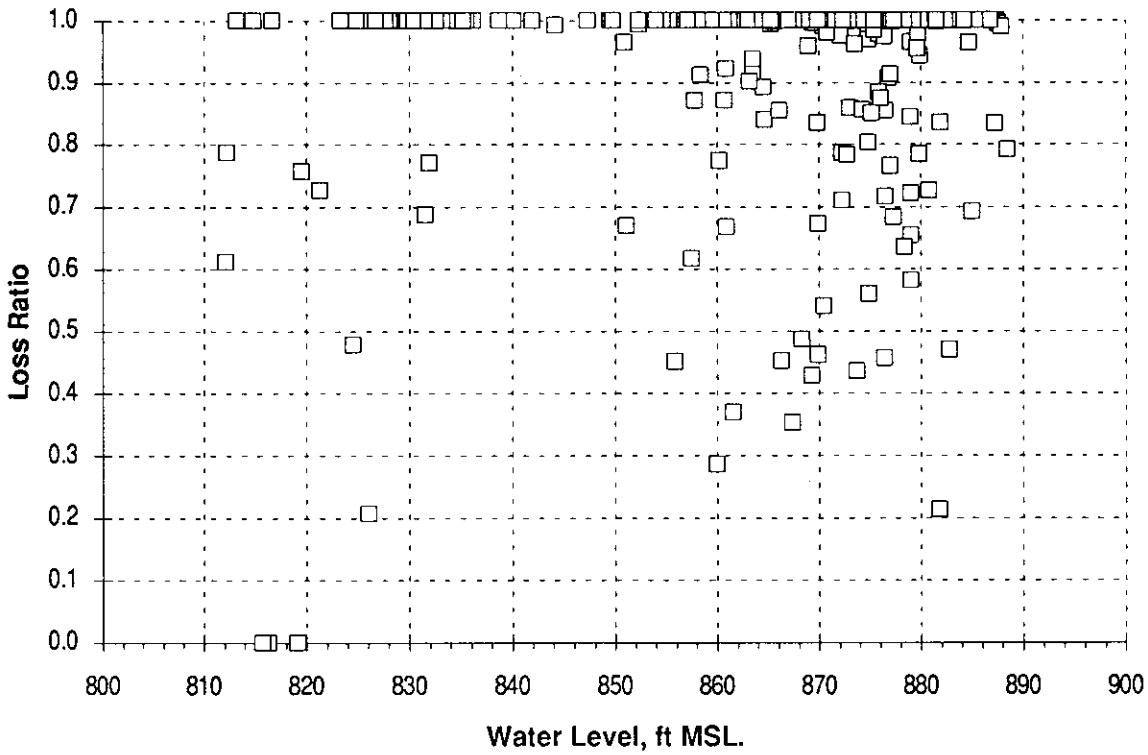


Figure 11. A plot of loss ratio versus water level in the Frio River Basin

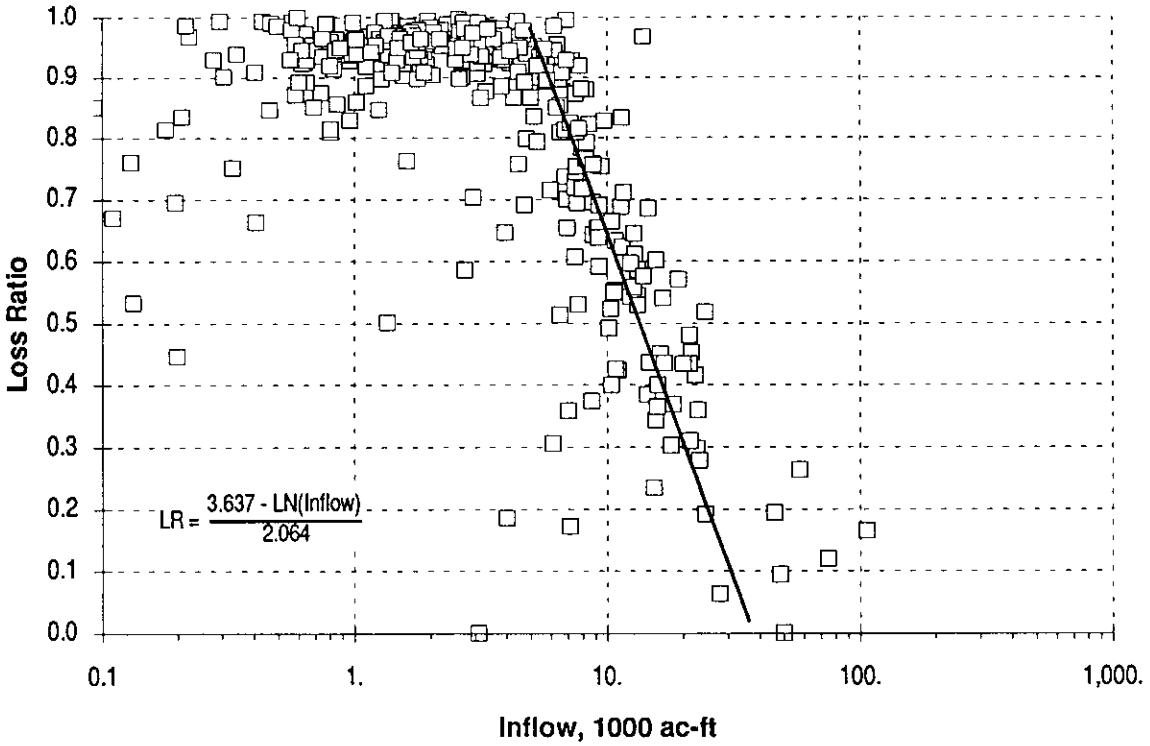


Figure 12. A plot of loss ratio versus inflow in the Sabinal River Basin

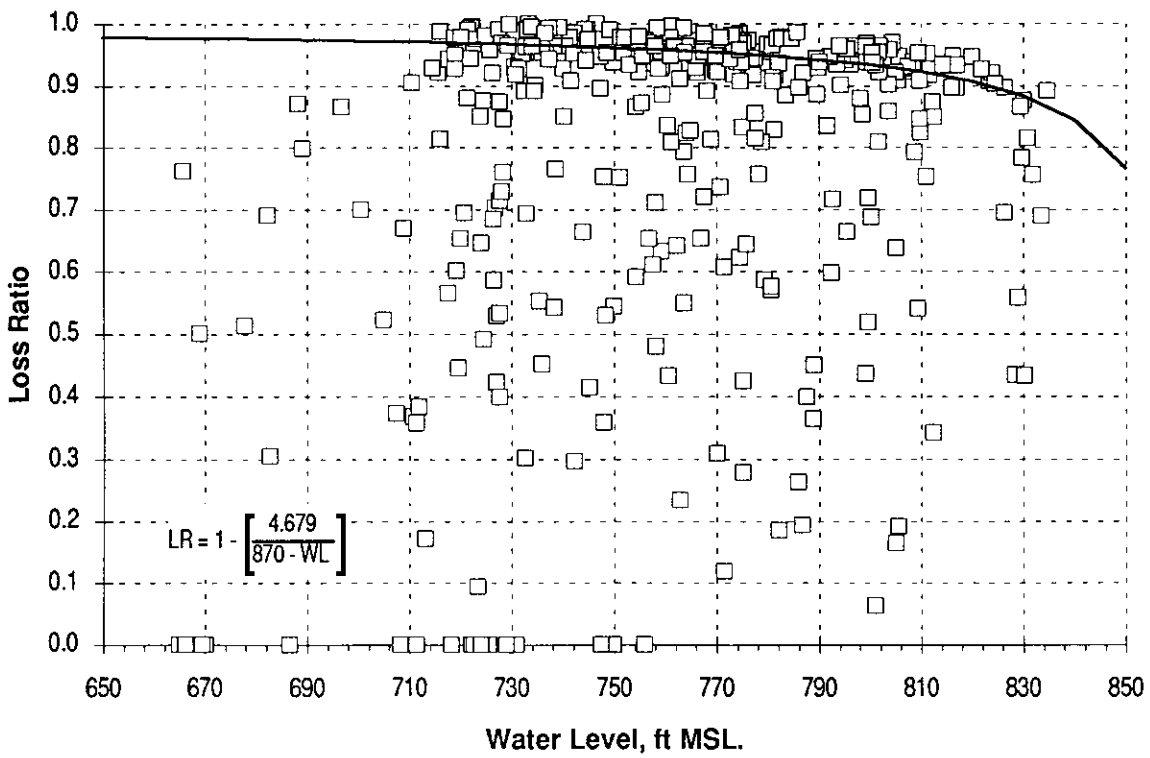


Figure 13. A plot of loss ratio versus water level in the Sabinal River Basin

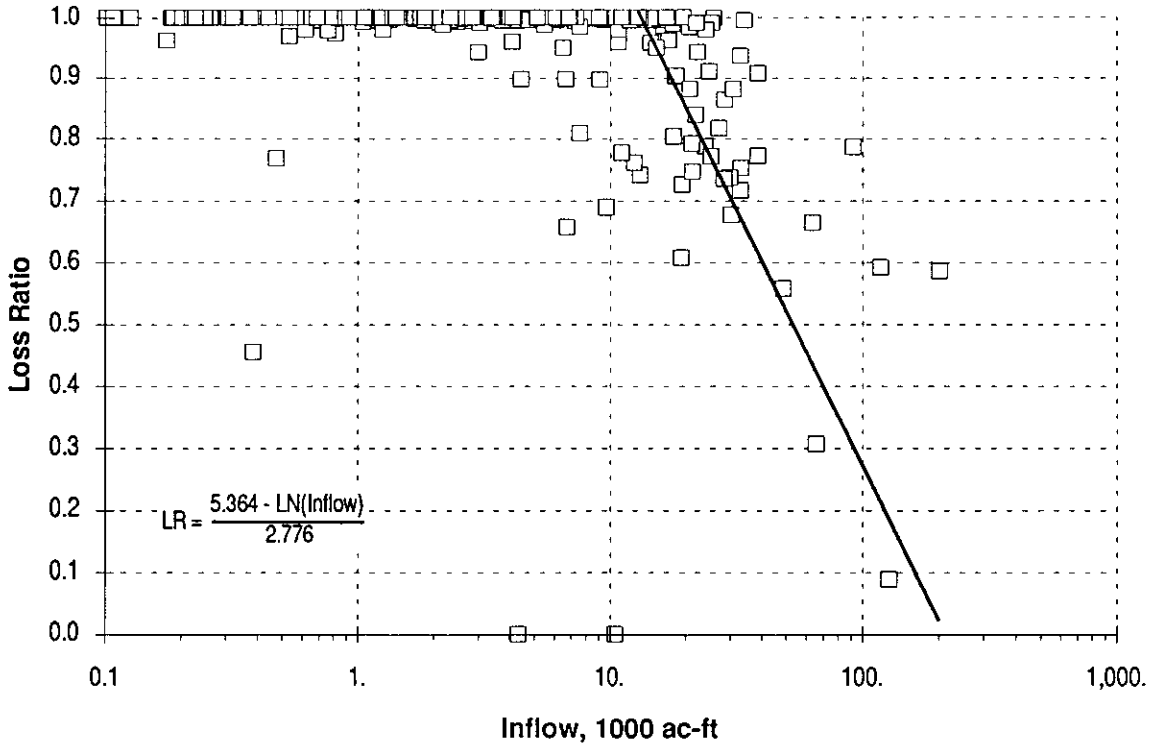


Figure 14. A plot of loss ratio versus inflow in the the Seco-Hondo Creek Basin

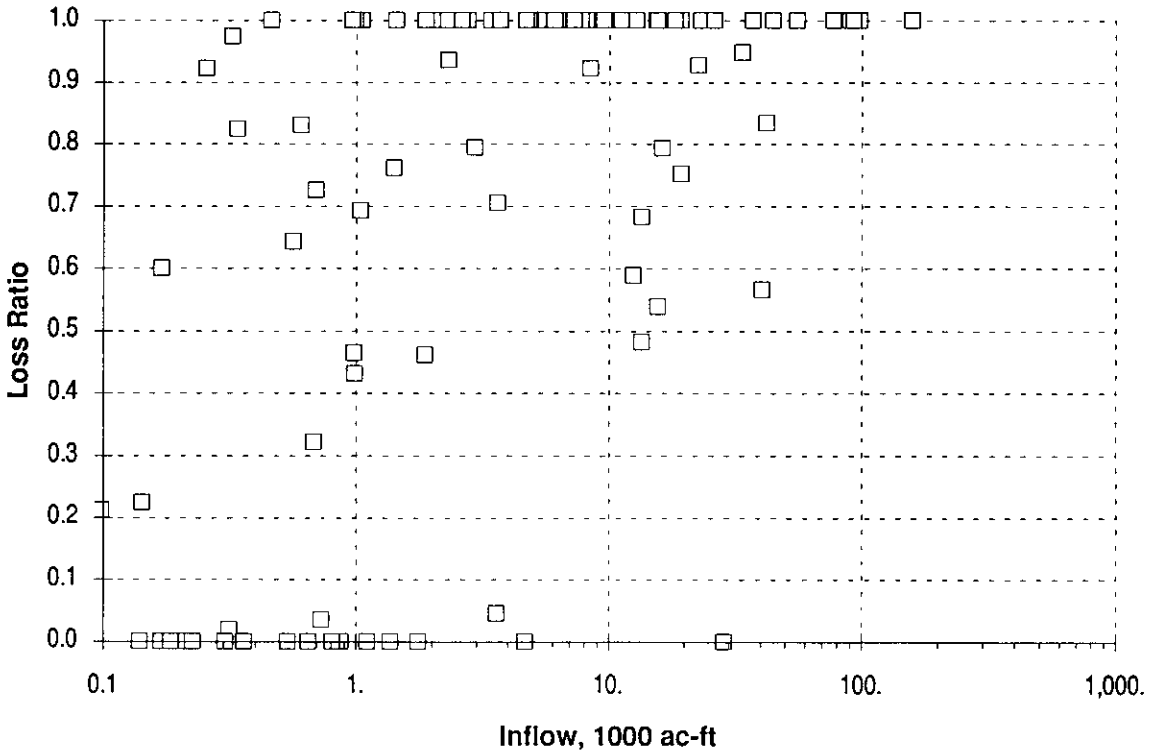


Figure 15. A plot of loss ratio versus inflow in the Helotes-Salado Creek Basin

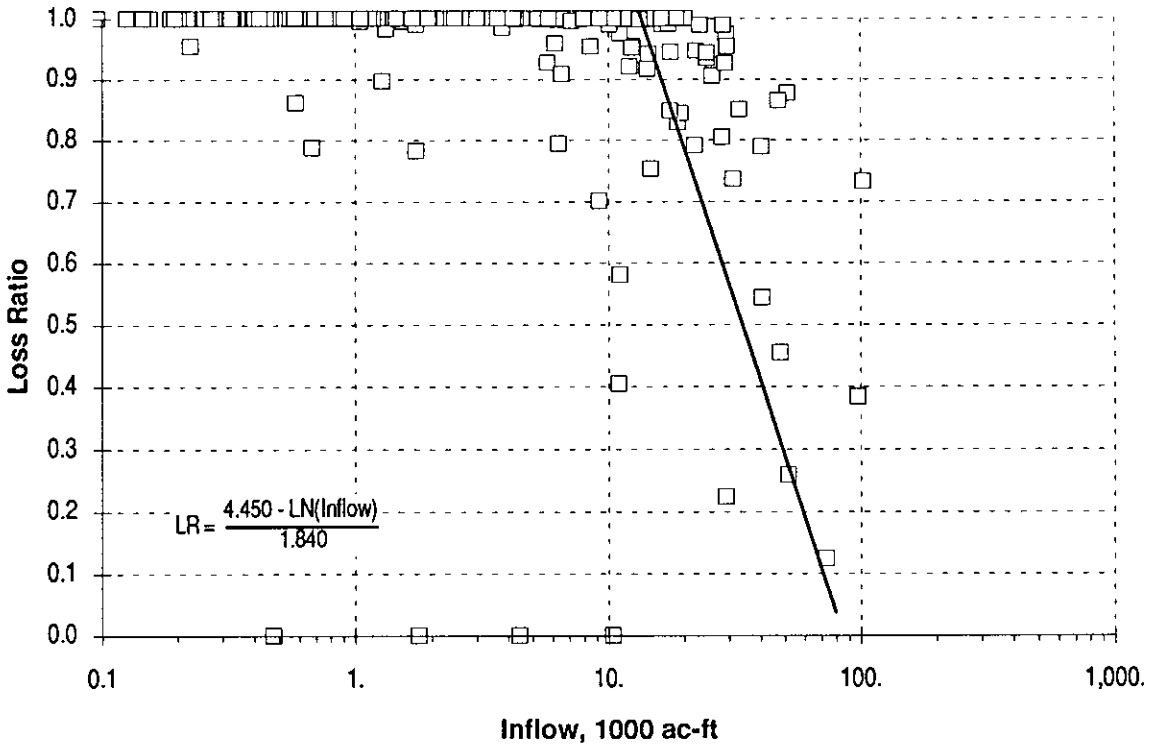


Figure 16. A plot of loss ratio versus inflow in the Cibolo-Dry Comal Creek Basin

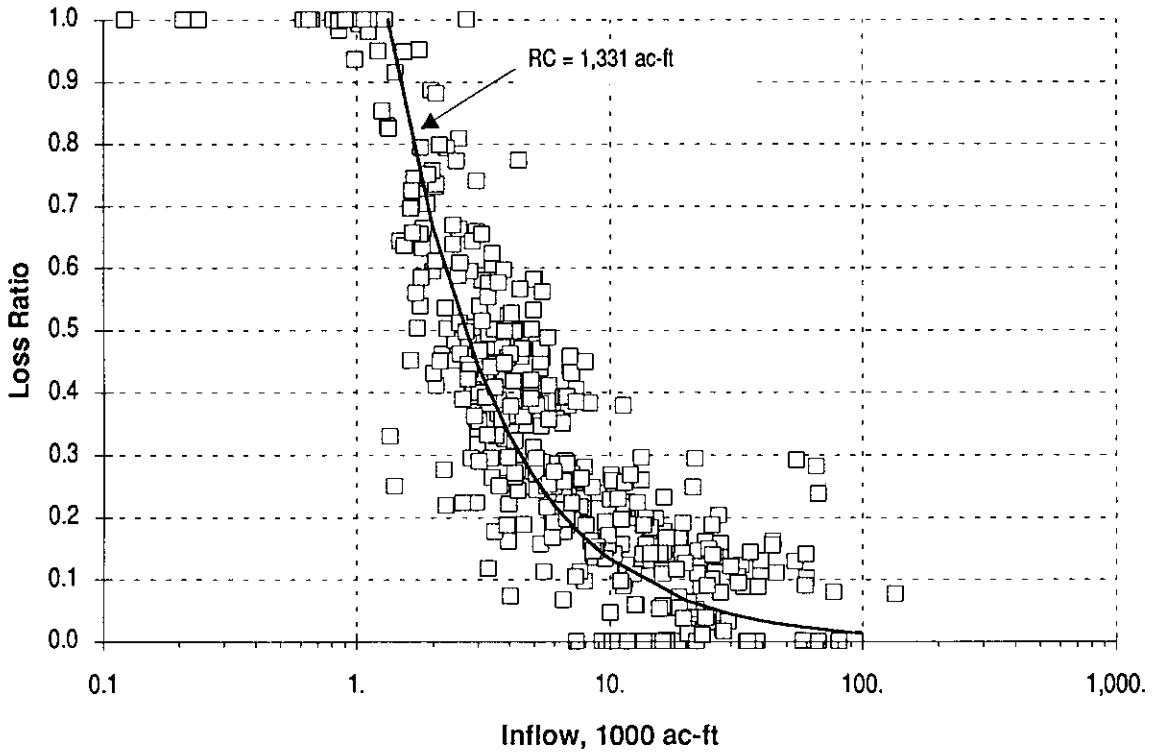


Figure 17. A plot of loss ratio versus inflow in the Blanco River Basin

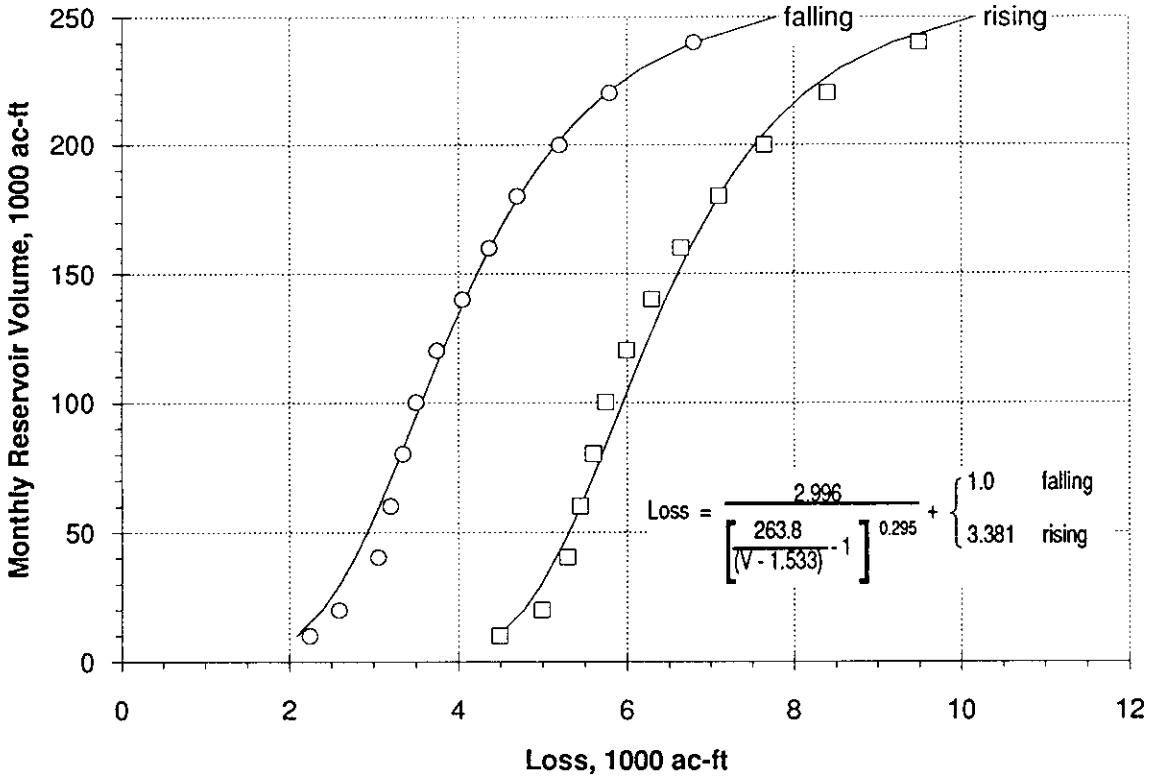


Figure 18. The loss curves for estimating recharge from Medina Reservoir

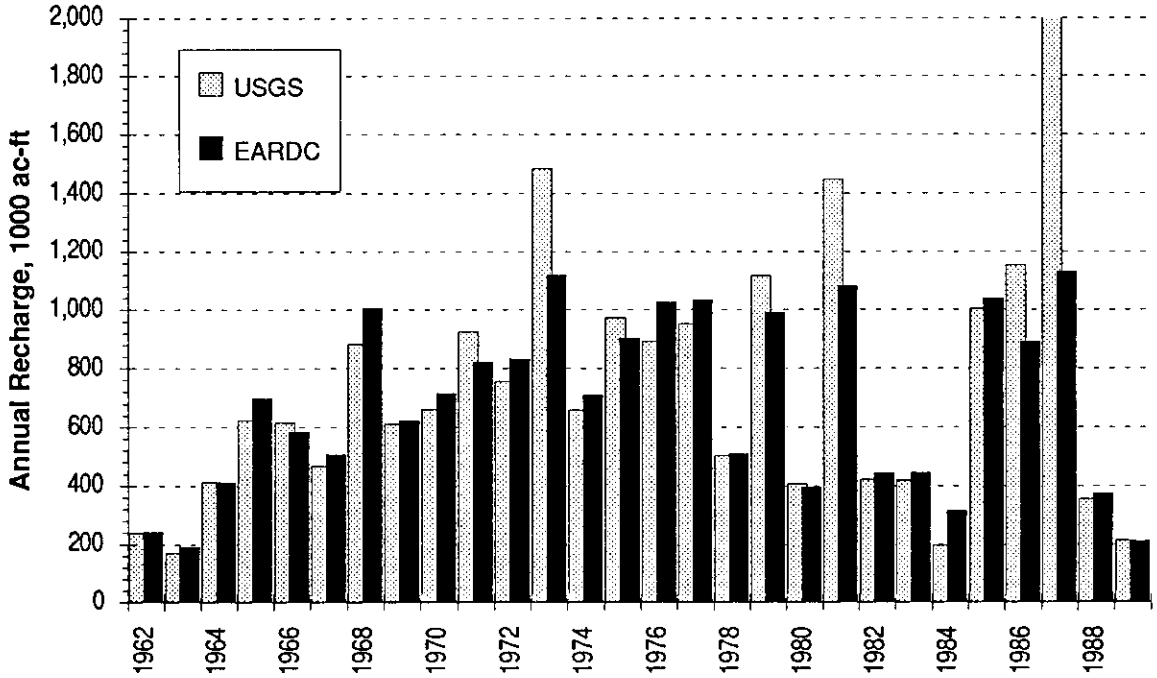


Figure 19. Comparison of recharge estimates by the USGS and the recharge functions developed by this study

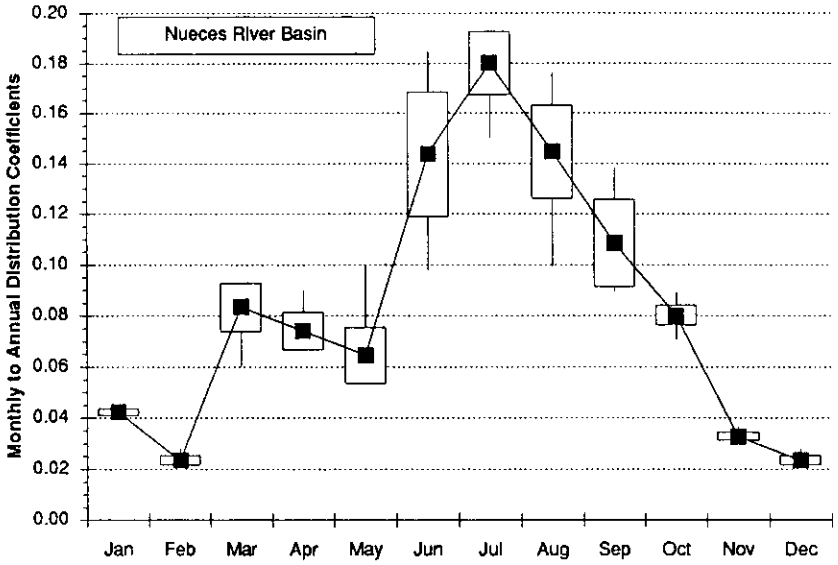


Figure 20. Nueces River Basin pumpage distribution coefficients

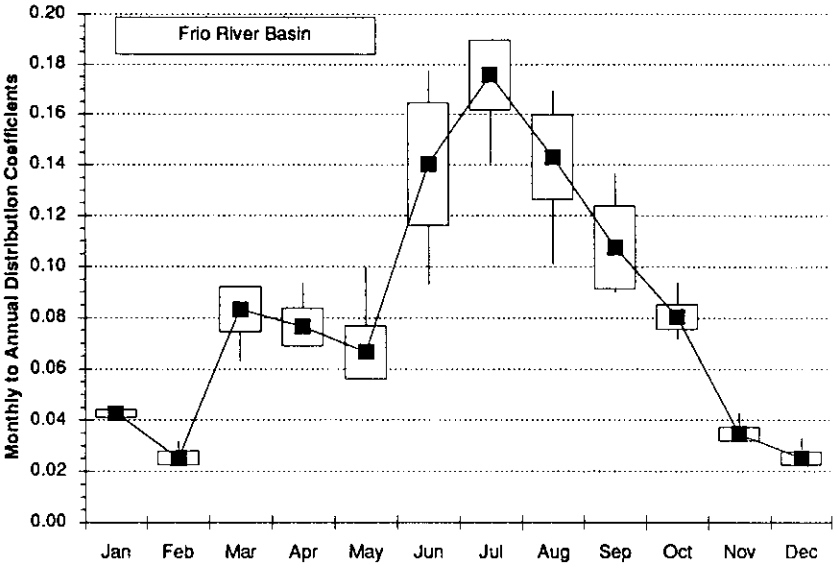


Figure 21. Frio River Basin pumpage distribution coefficients

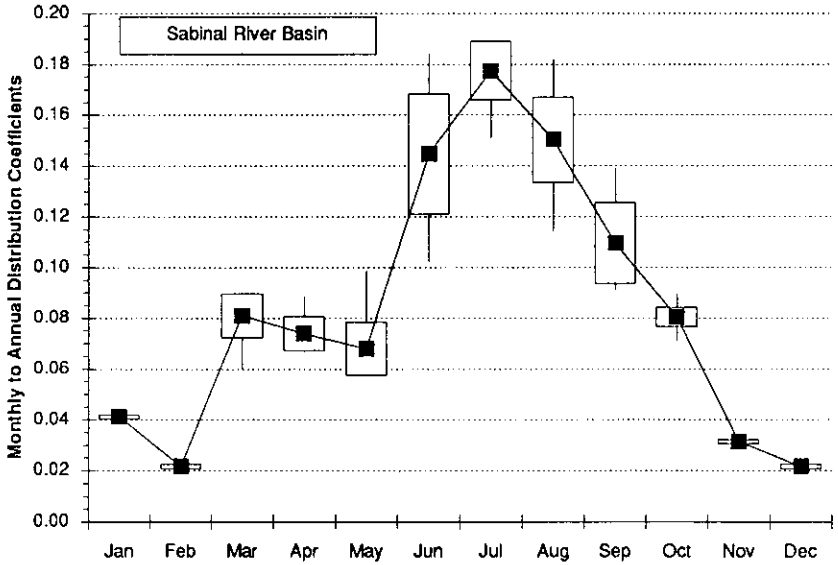


Figure 22. Sabinal River Basin pumpage distribution coefficients

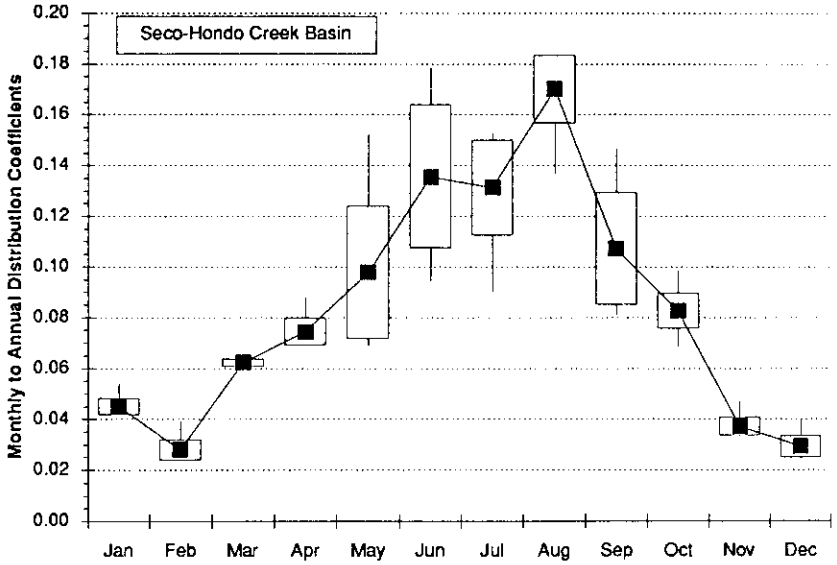


Figure 23. Seco-Hondo Creek Basin pumpage distribution coefficients

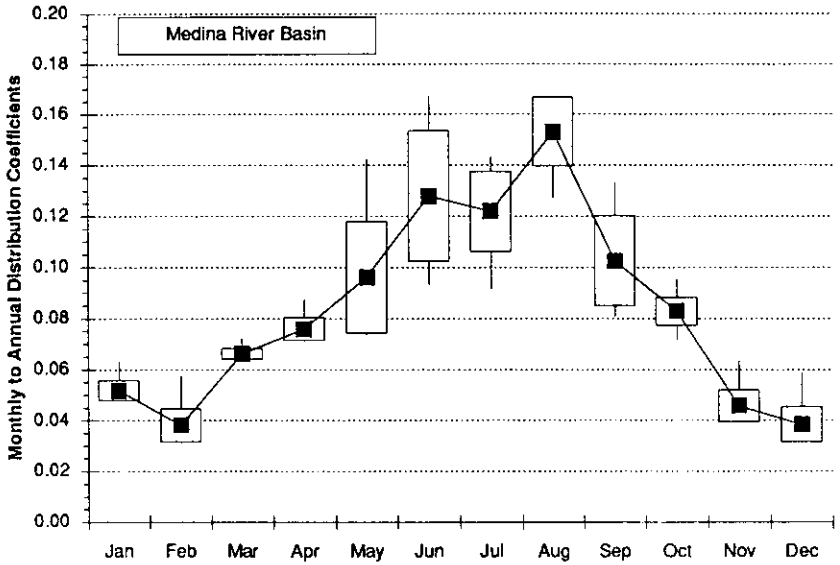


Figure 24. Medina River Basin pumpage distribution coefficients

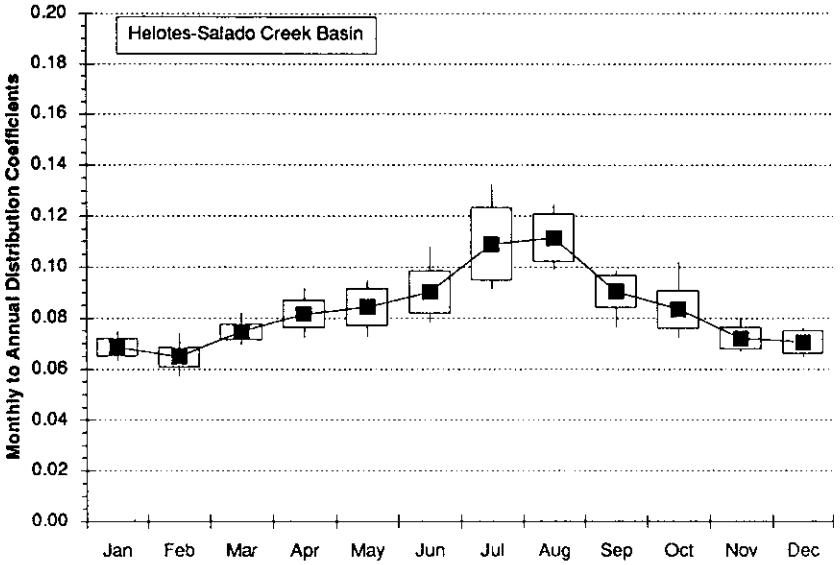


Figure 25. Helotes-Salado Creek Basin pumpage distribution coefficients

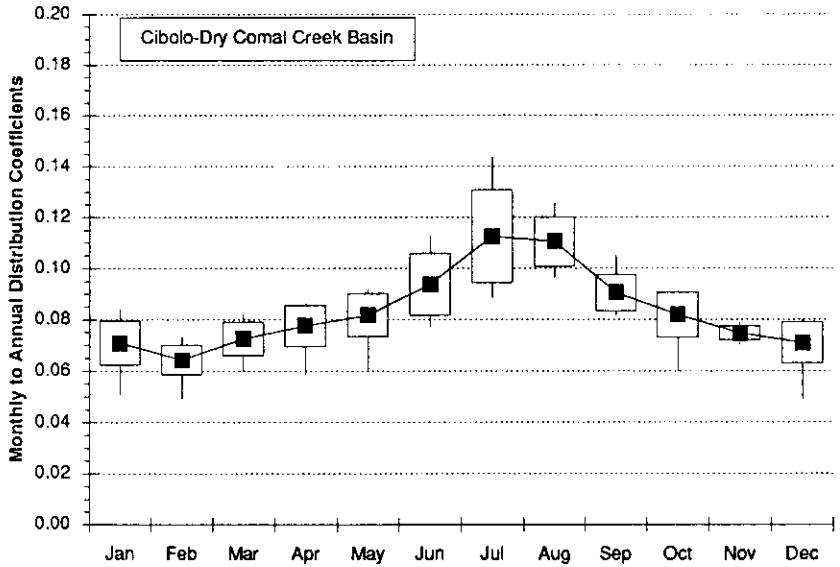


Figure 26. Cibolo-Dry Comal Creek Basin pumpage distribution coefficients

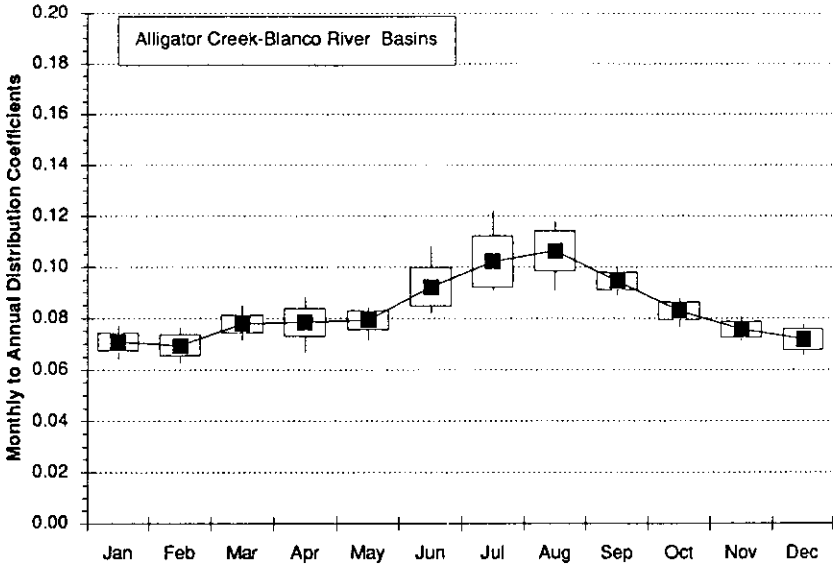


Figure 27. Alligator Creek-Blanco River Basin pumpage distribution coefficients

MODEL VERIFICATION AND CALIBRATION

Before the model can be used, its physical and flow parameters had to be calibrated and verified with two separate data sets. Stream flow, water level, and pumpage data are the only time dependent inputs and are divided into two sets, January 1975 to September 1990 and March 1962 to December 1974. The former were used in calibration while the latter were used for validation. Conventional procedures for model calibration use a trial and error method in accordance with a sensitivity analysis. In this study, a modern optimization procedure for model calibration was adopted. The procedure automates the trial and error process by formulating and solving a nonlinear programming problem to calibrate parameters.

Database Development

A database for the model input has been established on a microcomputer spreadsheet. Many useful features and functions in the spreadsheet, such as cell formulas, command macros, auto-update, and linking make the software very attractive for use in preparing and manipulating the model input. For instance, the software makes it more convenient to change stream-flow units or to obtain ungaged stream-flow data by interpolation. Furthermore, this model's development was geared toward using the model in a microcomputer spreadsheet.

Estimation of Parameters

The calibrating parameters include the storage related parameters in each sub-basin and the flow transmissivity related parameters in each link. Initial values of the parameters are required as a starting point in the calibration process. To estimate the initial values, relationships between the aquifer storativity versus the model storage parameter and the aquifer transmissivity versus the model flow transmissivity parameter need to be determined. By definition, the storativity, S_y, is defined as the volume of water that an aquifer releases from storage per unit surface area per unit decline in hydraulic head. Using equation (11), S_y for a sub-basin can be expressed in terms of storage parameter, S, as follows:

$$S_y = \frac{1}{A_b}\frac{d\forall}{dx} = \frac{1}{A_b} S' n(x-x_0)^{n-1} = \frac{1}{A_b} S \qquad (35)$$

where A_b is the basin surface area, and $d\forall/dx$ is the change in water storage per unit change in water level. Klemt, et al. (1978) reported the values of S_y to be 0.02 in the unconfined portion of the Edwards Aquifer and 0.0004-0.0007 in the confined. Maclay and Land (1988) suggested wider ranges of S_y, i.e., 2-20% for the unconfined and 10^{-5}-10^{-4} for the confined. For a S_y value of 0.01 or 1%, the equivalent S value in 10^3 ac-ft/ft can be computed from

$$S = 0.0064\, A_b S_y \tag{36}$$

where A_b is given in square miles and S_y is in percent. This equation is used to estimate the basin storage related parameters in which n is initially set to unity.

For transmissivity, Klemt, et al. (1978) performed pumping tests and reported the values of 0.1-0.2 M gal/ft/day (0.155-0.309 ft^2/s) in the water table zone and 10-20 M gal/ft/day (15.5-30.9 ft^2/s) in the artesian zone. Maclay and Land (1988) suggested the values are as high as 0.1-20.0 ft^2/s in the unconfined and 20-100 ft^2/s for the confined. In equation (14) we defined the flow transmissivity related parameter as $T = T'w/L$ where T' is the transmissivity. For a T' of 1 ft^2/s, the equivalent value of T in 10^3 ac-ft/ft/mo can be computed from

$$T = 0.06033 \left(\frac{w}{L}\right) T' \tag{37}$$

where w and L are in feet and T' is in ft^2/s. The equation is used to estimate the flow transmissivity related parameters for each basin. Note that the parameters used in model input and calibration are in terms of T/b where b is the basin thickness.

Other physical parameters, e.g. elevations of the top and bottom of the aquifer in each basin, are not adjusted during the calibration. Each basin value is calculated from the TWDB model input by averaging the cell values in the basin. Springflow parameters are determined from the regression analysis as shown in Figure 28.

Optimization in Calibration Process

The calibration process is automated by solving the following nonlinear programming problem ,

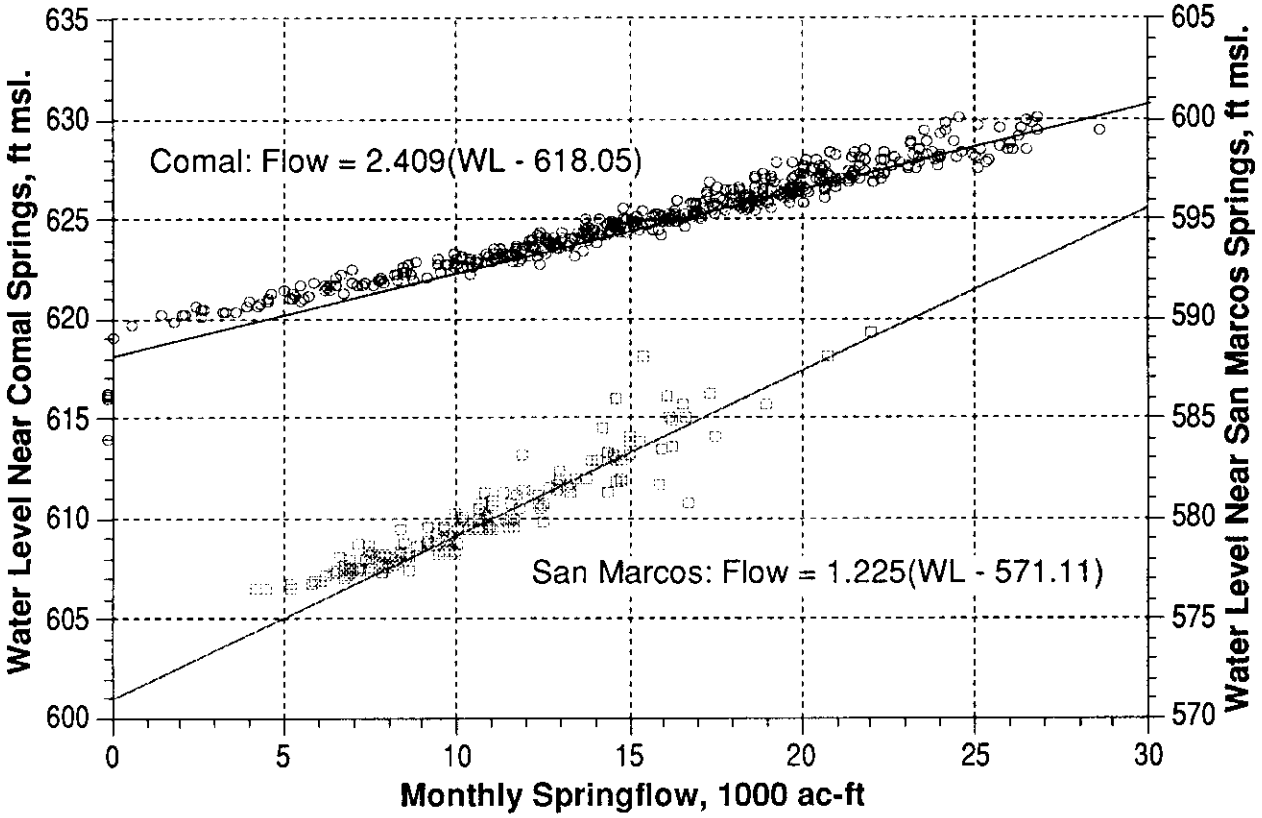


Figure 28. Relationship between springflows and water levels in the nearby wells

$$\text{Minimizing} \qquad f(x) = \sum_i (x_i - X_i)^2 \tag{38}$$

$$\text{subject to} \qquad \mathbf{g}(x,u,T,S) = \mathbf{0}, \tag{39}$$

$$T_i \geq 0, \; S_i \geq 0, \text{ for all } i \tag{40}$$

where the objective function is the sum of squares of differences between observed and simulated water levels. Except variable bounds, the only constraint in this problem is the system of equations governing the simulation model. Solving this mathematical programming problem requires an interaction between an optimizer and the simulation. The optimizer used in this study is called GRG2 (Lasdon and Waren, 1986). The constraint subroutine is modified such that the simulation model is called to compute water levels before the objective function is evaluated. During each iteration, GRG2 assesses the current search status and suggests a new strategic set of parameters. The simulation model takes the parameters, and

computes a new set of water levels for GRG2. The iteration stops when an optimal set of parameters is obtained.

Typically, a few initial sets of guess parameter values must be used to avoid a local optimum. Due to the extreme nonlinearity in the nature of the simulation model, especially when the storage exponent is not at unity, steep valleys and ridges in constraint function are realized. Thus, a number of initial sets of parameters were attempted during the course of the calibration process. The best parameter values obtained by comparing all solutions are given in Appendix B as the model input. Results of the calibration are depicted in Figures 29-30.

Kalman Filtering

In this section we introduce a stochastic model and use it to condition the simulation results on the actual head measurements. Variability in input data and model simplification are common causes of uncetainty in the groundwater models. For this lumped parameter model, the input contains some variation due to errors in recharge and pumpage estimates and/or water level measurements although model assumptions are necessary to approximate reality. The model uncertainty can be extended from the deterministic model by introducing noise terms. If the statistics of system noise are known, an optimum filtering is usually used to quantify the model uncertainties. Kalman filtering is one of the widely used filtering techniques for hydrologic real time simulations. The rest of this section will explore how this filtering can improve the simulation results. The derivation of the algorithm can be found elsewhere (e.g. Sage and Melsa, 1976) and will not repeated. Consider the following system, a stochastic version of equation (6),

$$\mathbf{x}_{k+1} = \mathbf{P}_k\mathbf{x}_k + \mathbf{Q}_{u,k}\mathbf{u}_k + \mathbf{Q}_{w,k}\mathbf{w}_k \,, \tag{41}$$

$$\mathbf{y}_k = \mathbf{C}_k\mathbf{x}_k + \mathbf{v}_k \,, \tag{42}$$

where $\mathbf{u}_k$ is a deterministic input, $\mathbf{y}_k$ is the system measurable output, and $\mathbf{w}_k$ and $\mathbf{v}_k$ are the zero mean Gaussian noises for the system and output with the following covariance matrix:

$$E\left\{\begin{bmatrix}\mathbf{w}_k\\ \mathbf{v}_k\end{bmatrix}\begin{bmatrix}\mathbf{w}_k & \mathbf{v}_k\end{bmatrix}\right\} = \begin{bmatrix}\mathbf{W}_k & \mathbf{0}\\ \mathbf{0} & \mathbf{V}_k\end{bmatrix} \tag{43}$$

The Kalman filter algorithm is given as (Takahashi, et al., 1972):

$$\bar{\mathbf{x}}_{k+1} = \mathbf{P}_k\hat{\mathbf{x}}_k + \mathbf{Q}_{u,k}\mathbf{u}_k \tag{44}$$

$$\hat{\mathbf{x}}_{k+1} = \mathbf{K}_{k+1}\left\{\mathbf{y}_{k+1} - \mathbf{C}_{k+1}\bar{\mathbf{x}}_{k+1}\right\}, \qquad \hat{\mathbf{x}}_{-1} = \mathbf{0} \tag{45}$$

$$\mathbf{K}_{k+1} = \mathbf{M}_{k+1}\mathbf{C}_{k+1}^T\left(\mathbf{C}_{k+1}\mathbf{M}_{k+1}\mathbf{C}_{k+1}^T + \mathbf{V}_{k+1}\right)^{-1}, \tag{46}$$

$$\mathbf{M}_{k+1} = \mathbf{P}_k\mathbf{Z}_k\mathbf{P}_k^T + \mathbf{Q}_{w,k}\mathbf{W}_k\mathbf{Q}_{w,k}^T, \qquad \mathbf{M}_0 = \left\{\mathbf{x}_0\mathbf{x}_0^T\right\} \tag{47}$$

$$\mathbf{Z}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C}_k)\mathbf{M}_k \tag{48}$$

where

$\bar{\mathbf{x}}_{k+1}$ is the simulated state vector (water levels) using the optimal linear estimate of the state vector;

$\hat{\mathbf{x}}_{k+1}$ is the updated state vector conditioned on the measured water levels;

$\mathbf{C}_k$ is the measurement matrix which transforms the simulated state vector to the output vector where measurements are available;

$\mathbf{M}_k$ is the covariance matrix of the simulated error, $\mathrm{Cov}(\mathbf{x}_k - \bar{\mathbf{x}}_k)$;

$\mathbf{Z}_k$ is the covariance matrix of the measurement update error, $\mathrm{Cov}(\mathbf{x}_k - \hat{\mathbf{x}}_k$;

$\mathbf{K}_k$ is the Kalman gain; and

$\mathbf{I}$ is the unity matrix.

In this study, the output matrix $\mathbf{C}$ is unity and we assume $\mathbf{Q}_{w,k} = \mathbf{Q}_{u,k}$. The covariances $\mathbf{W}$ and $\mathbf{V}$ are predetermined and fixed. The diagonal elements of $\mathbf{W}$ are determined from the variances of simulation errors while the measurement errors are assumed to have a standard error of 2 ft. The result of applying the Kalman filtering improved the simulation dramatically as shown in Figures 31 and 32.

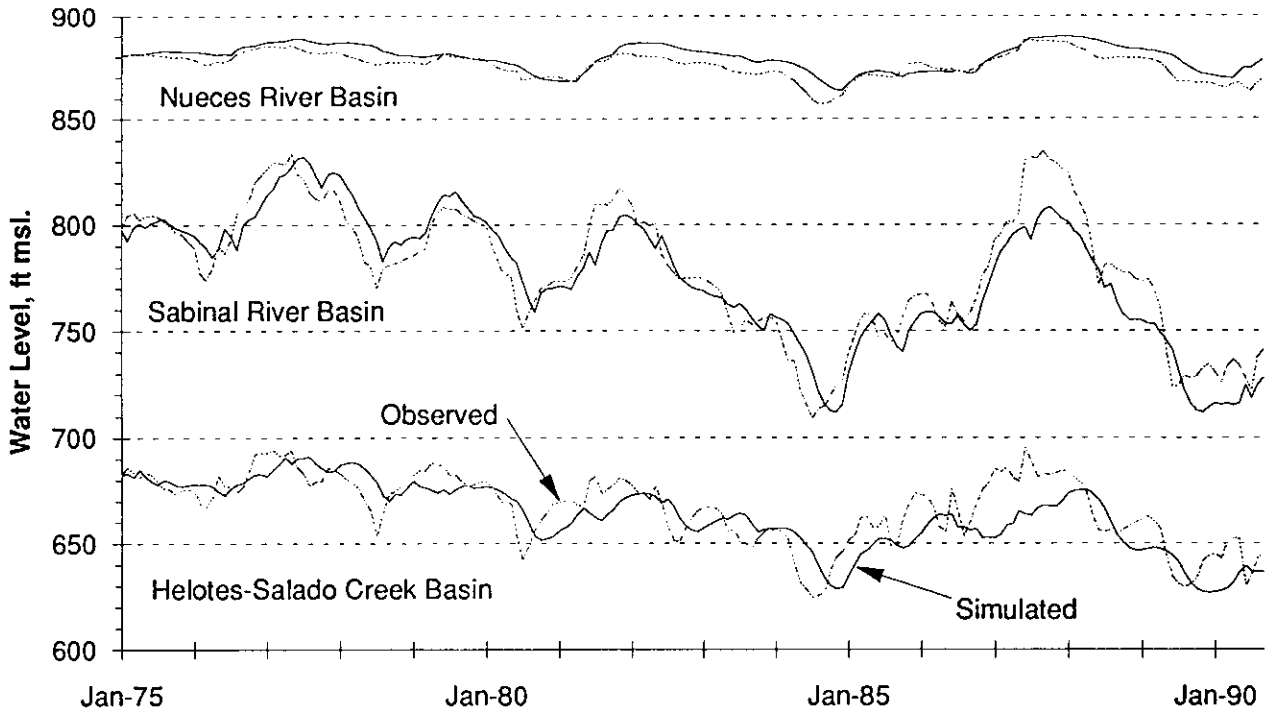


Figure 29. Comparision of the simulated and the observed water levels in selected river basins

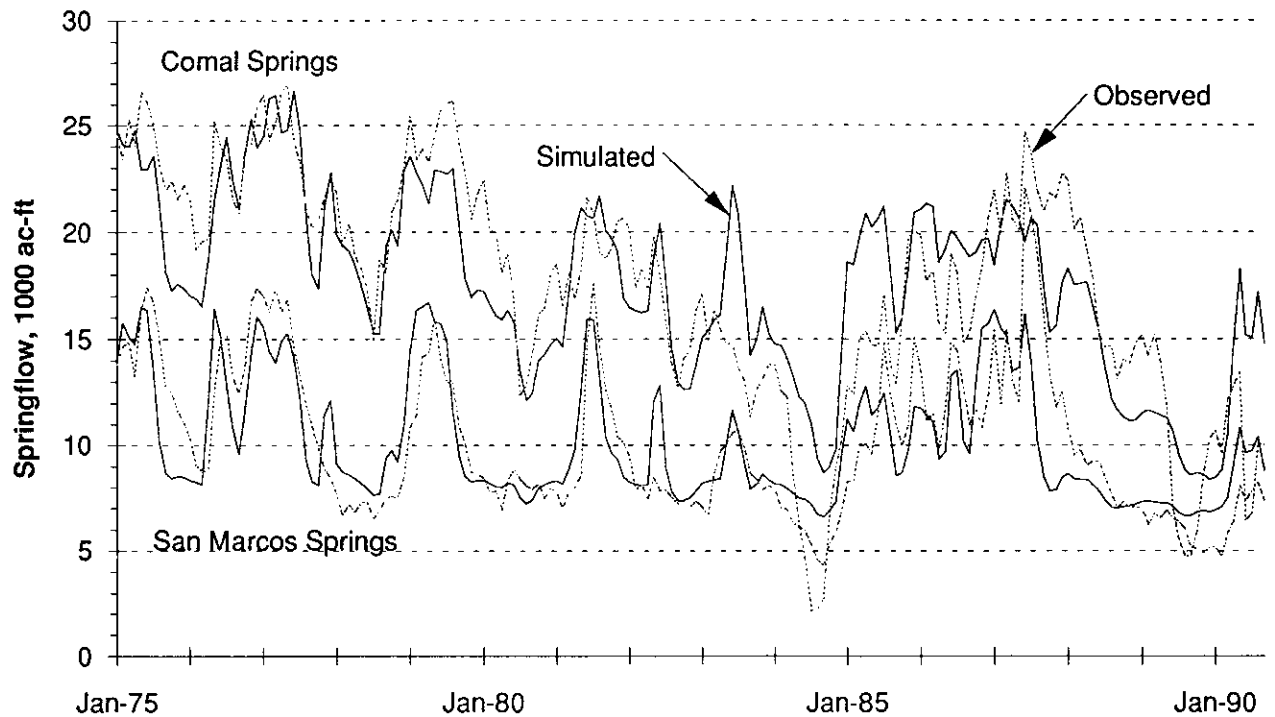


Figure 30. Comparision of the simulated and the observed springflows in the Comal and San Marcos Springs

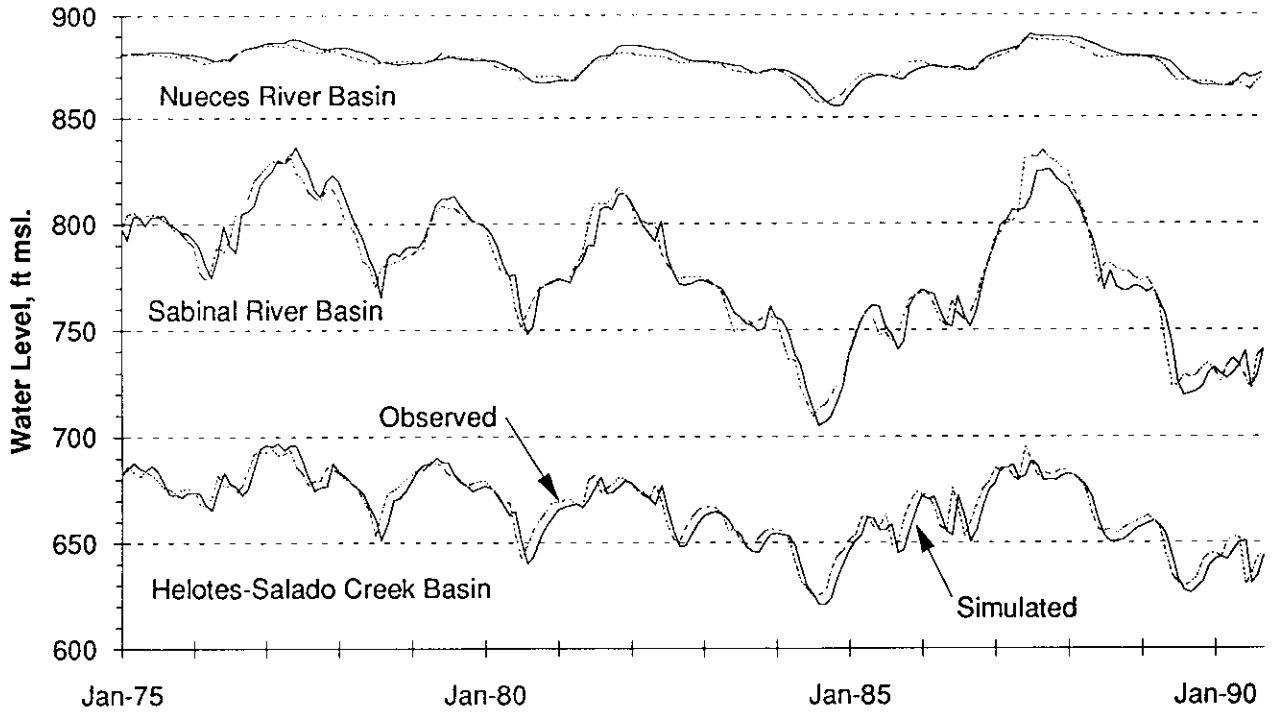


Figure 31. Comparision of the simulated and the observed water levels in selected river basins when applying the Kalman filter

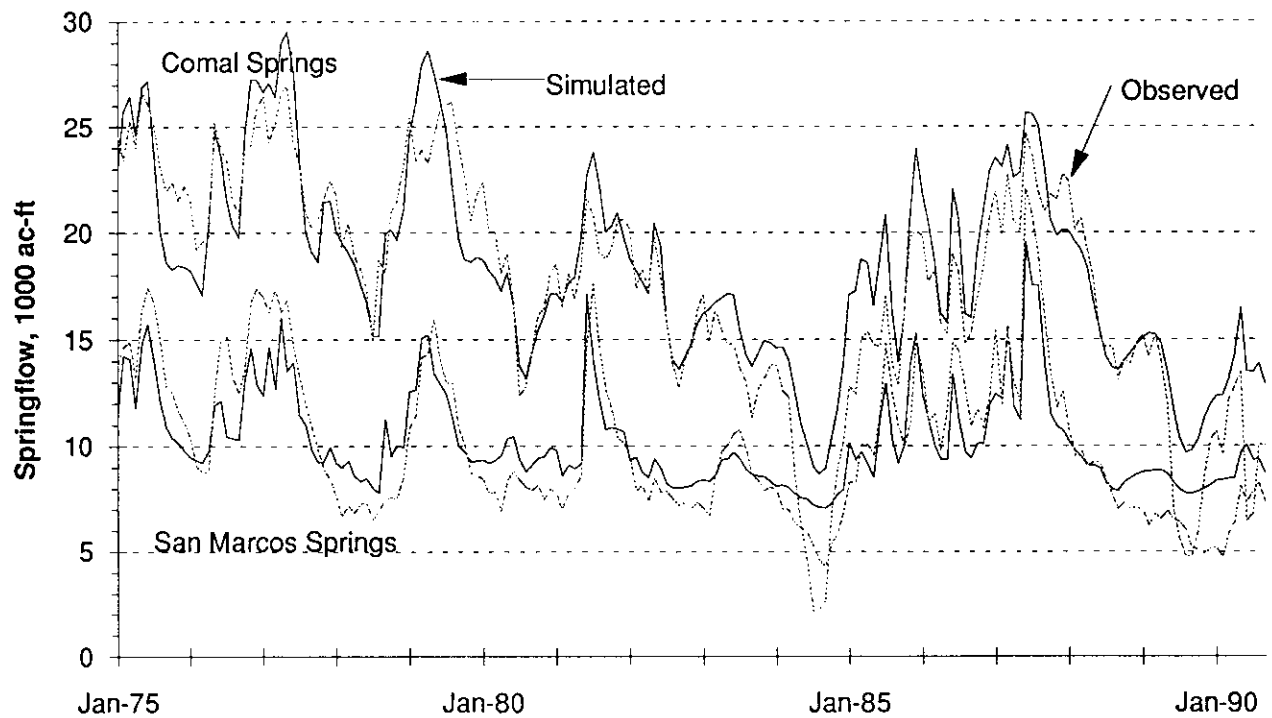


Figure 32. Comparision of the simulated and the observed springflows in the Comal and San Marcos Springs when applying the Kalman filter

Sensitivity Analysis

Sensitivity analysis was performed by comparing the results from small perturbations of system parameters. Only the storage and flow transmissivity related parameters were considered in this analysis. Figures 33-34 show changes of water levels as the transmissivity related parameters for all sub-basins are increased by 2, 5, and 10%. In effect, the groundwater flows faster from upstream basins to downstream basins and, thus, causes the water levels in the upstream basins to drop. However, when the parameter is increased in only a specific link, the hydraulic gradient in the link must drop to maintain the water balance in the system. As a result, the water levels will decrease in the upstream basin and increase in the downstream basin. These changes can be observed in Figures 35-38.

Sensitivity of the storage constant to the aquifer water levels is totally dependent on the basin. Generally, if storage related parameters are decreased, one should expect water levels to increase to maintain the same net flow in the water balance equation (Figures 39 and 40). This explanation is also applied to a small basins with a greater net inflow (Figures 43 and 44). Ironically, this is not the case for a larger basin with a small net flow, especially for the upstream basins (Figures 41 and 42).

In general, water levels are more sensitive to changes in storage related parameters than flow transmissivity related parameters. However, an increase in both parameters causes the water level to change in the opposite direction.

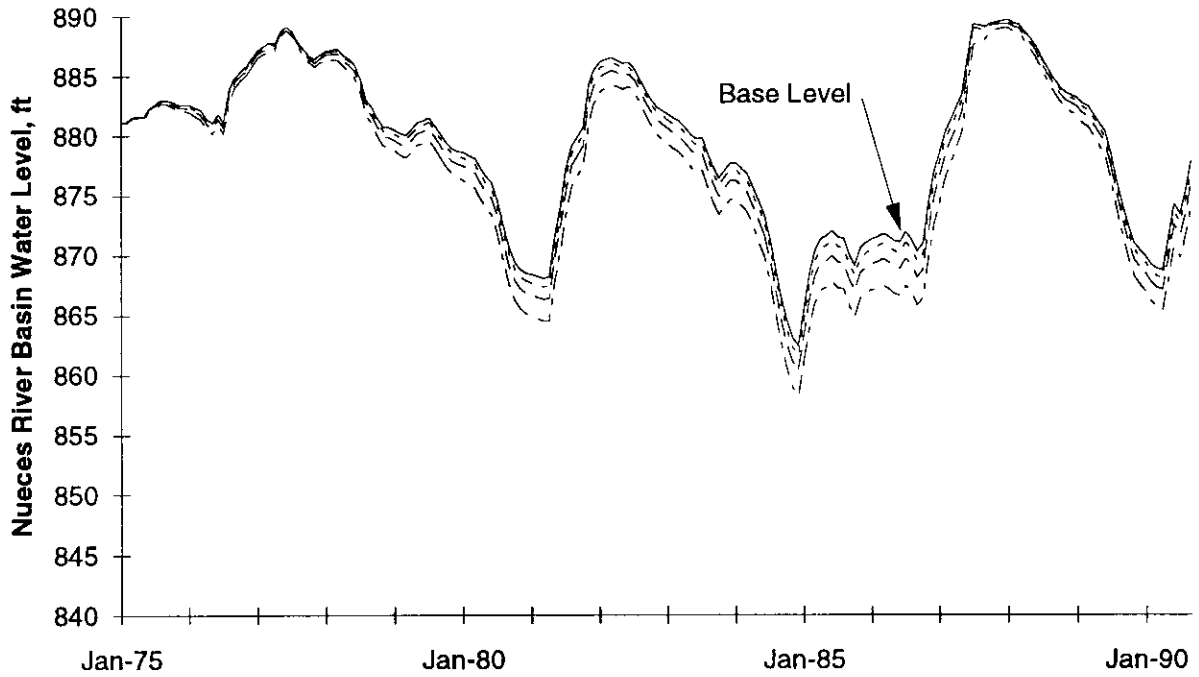


Figure 33. Water levels changes in the Nueces River Basin as all transmissivity related parameters are increased by 2, 5, and 10%

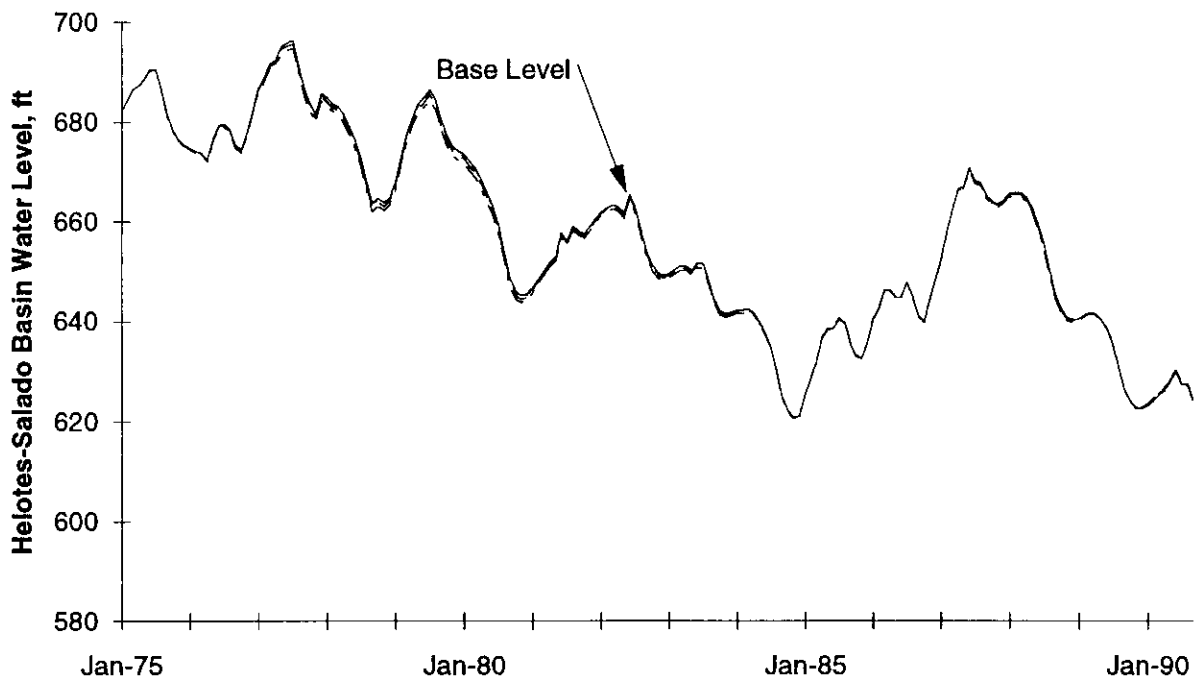


Figure 34. Water levels changes in the Helotes-Salado Creek Basin as all transmissivity related parameters are increased by 2, 5, and 10%

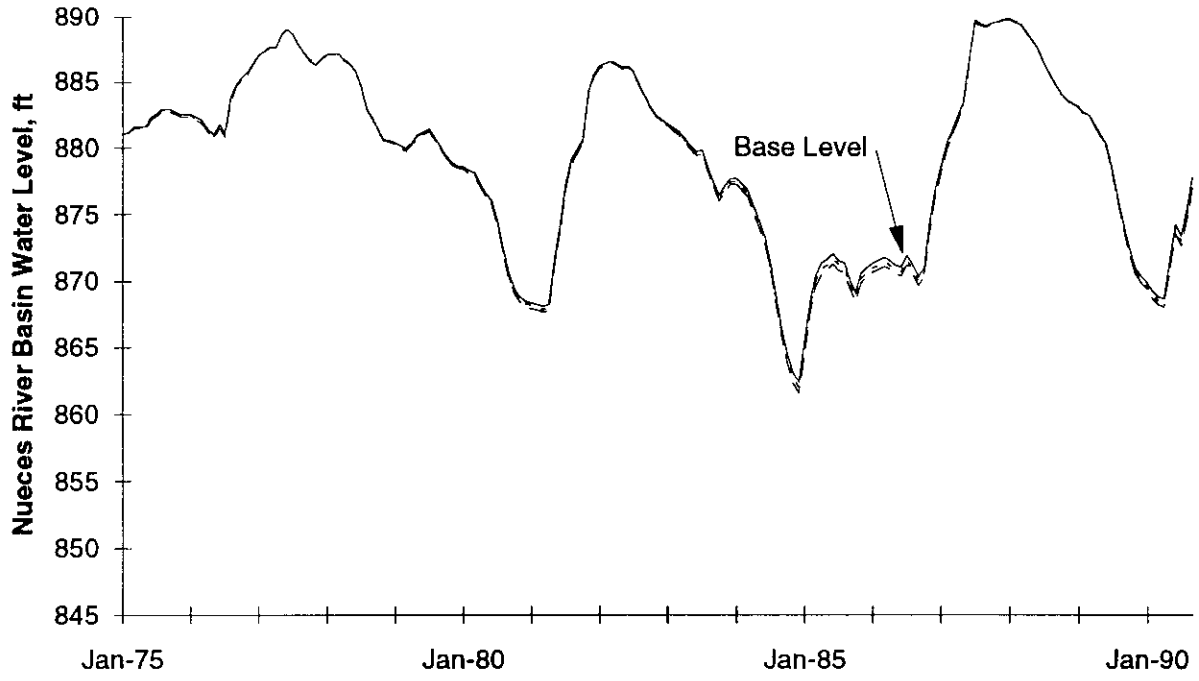


Figure 35. Water levels changes in the Nueces River Basin as the transmissivity related parameters in the link between Nueces and Frio River Basins are increased by 5 and 10%

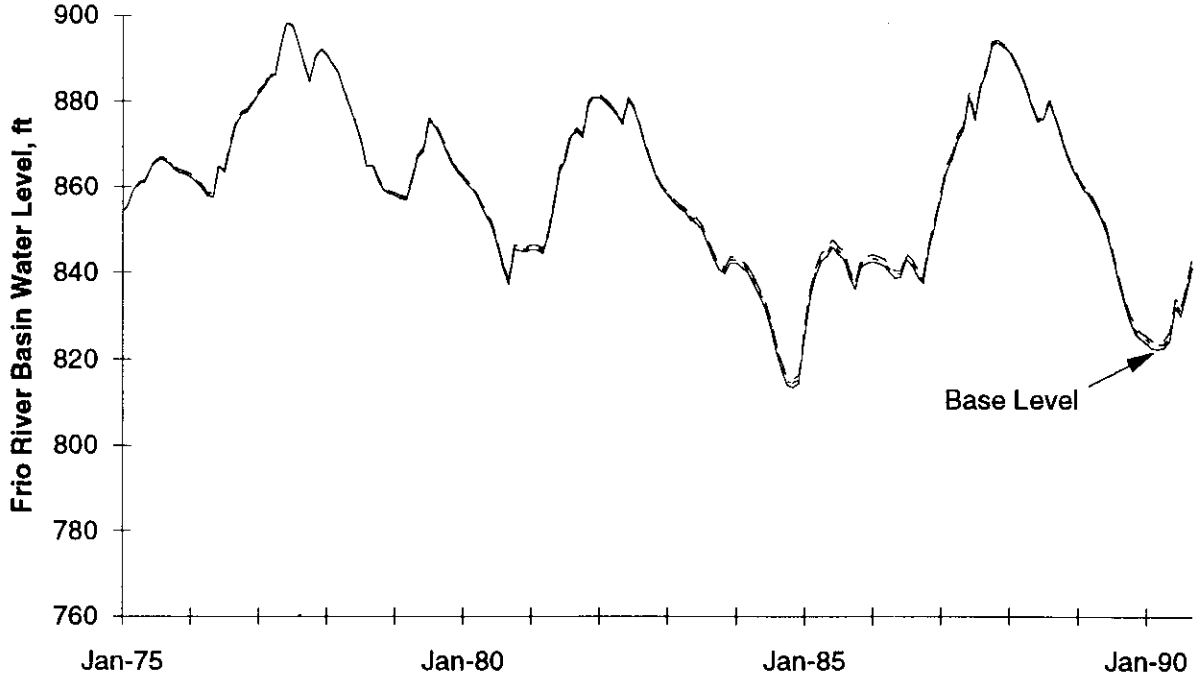


Figure 36. Water levels changes in the Frio River Basin as the transmissivity related parameters in the link between Nueces and Frio River Basins are increased by 5 and 10%

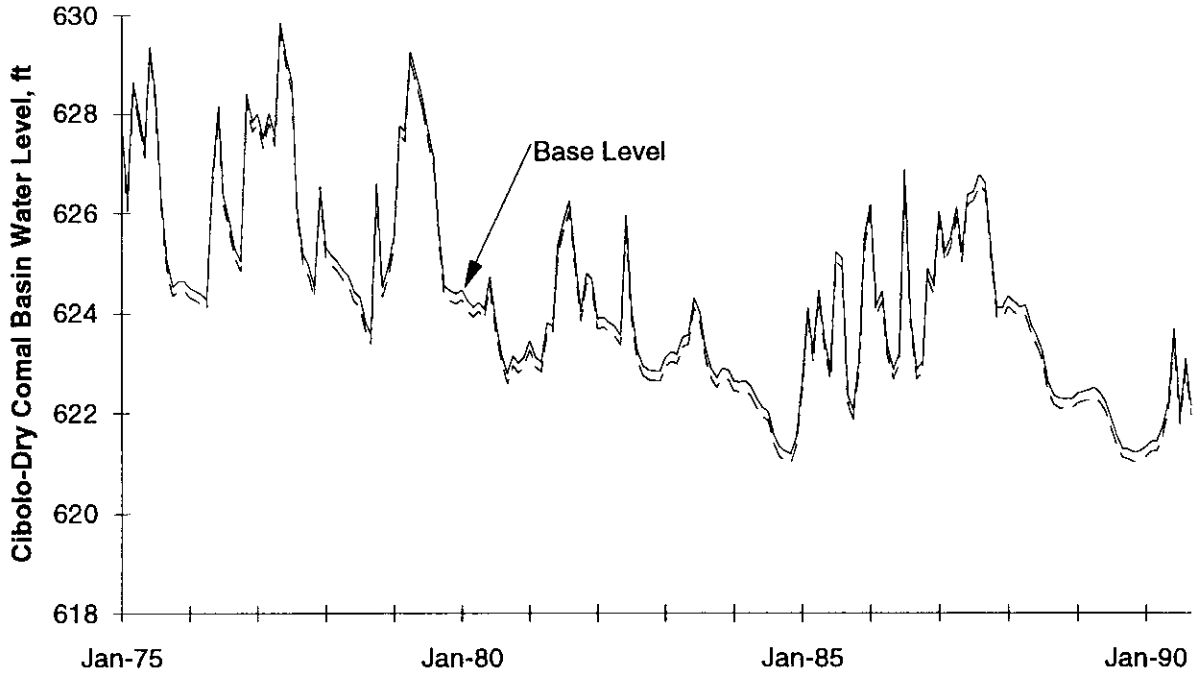


Figure 37. Water levels changes in the Cibolo-Dry Comal Creek Basin as the transmissivity related parameters in the link between Cibolo-Dry Comal and Blanco-Alligator River Basins are increased by 10%

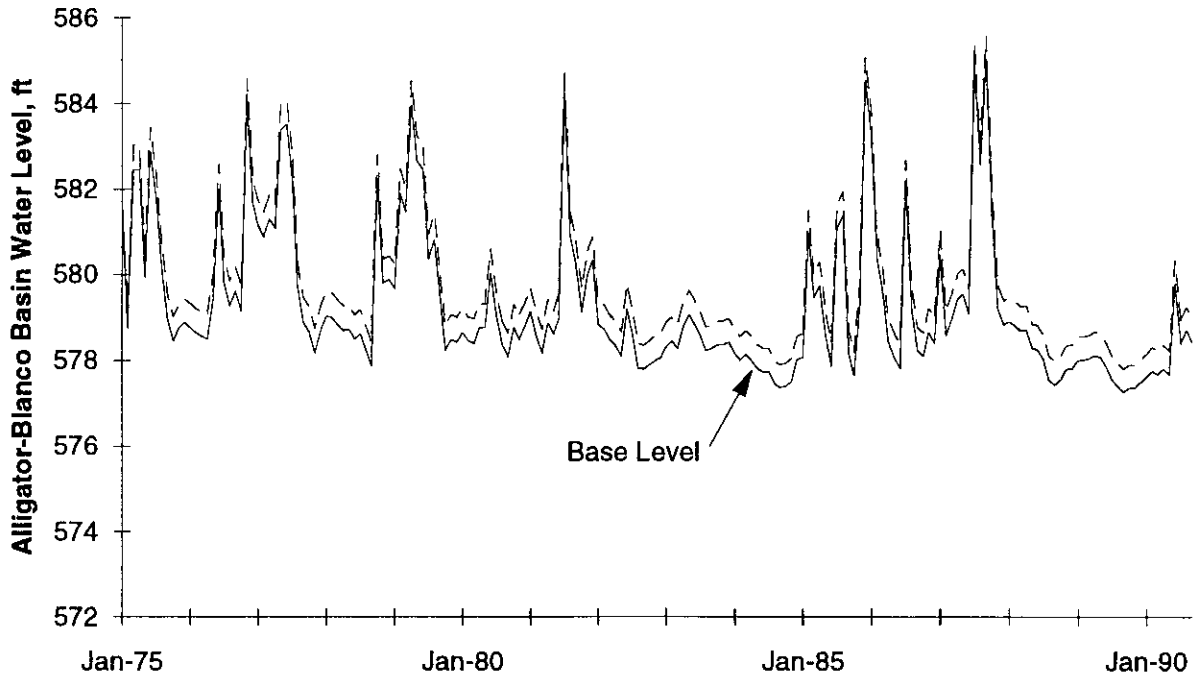


Figure 38. Water levels changes in the Alligator Creek-Blanco River Basin as the transmissivity related parameters in the link between Cibolo-Dry Comal Creek Basin and Alligator Creek-Blanco River Basin are increased by 10%

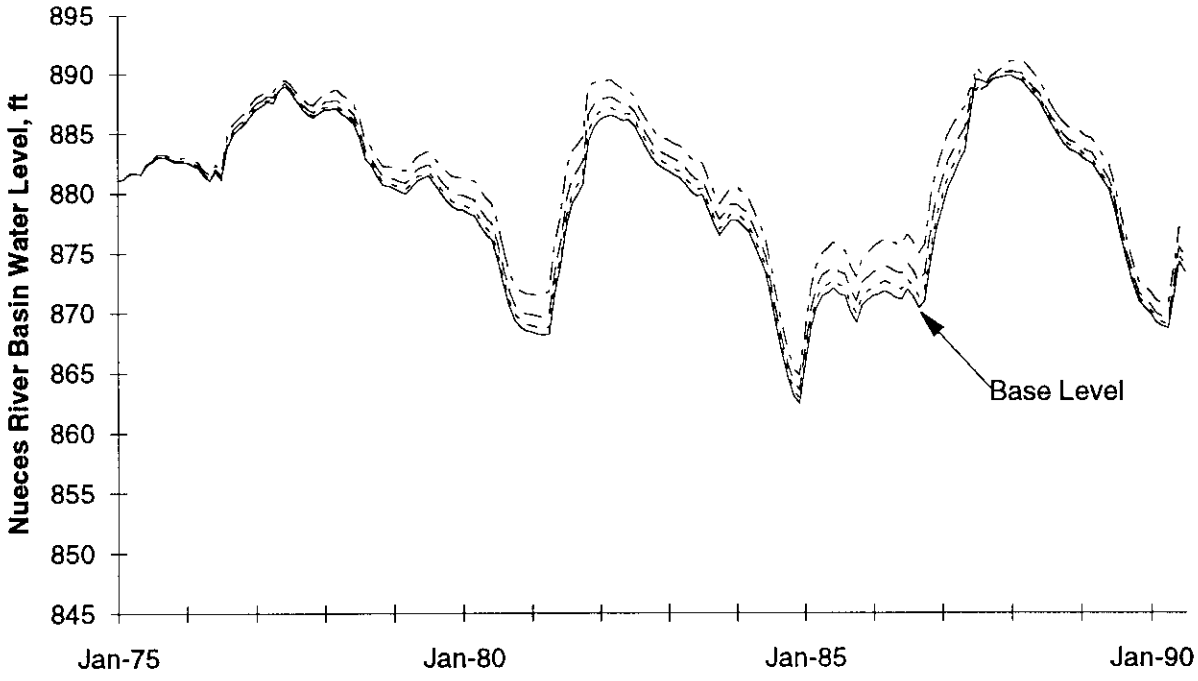


Figure 39. Water levels changes in the Nueces River Basin as all storage related parameters are decreased by 2, 5, and 10%

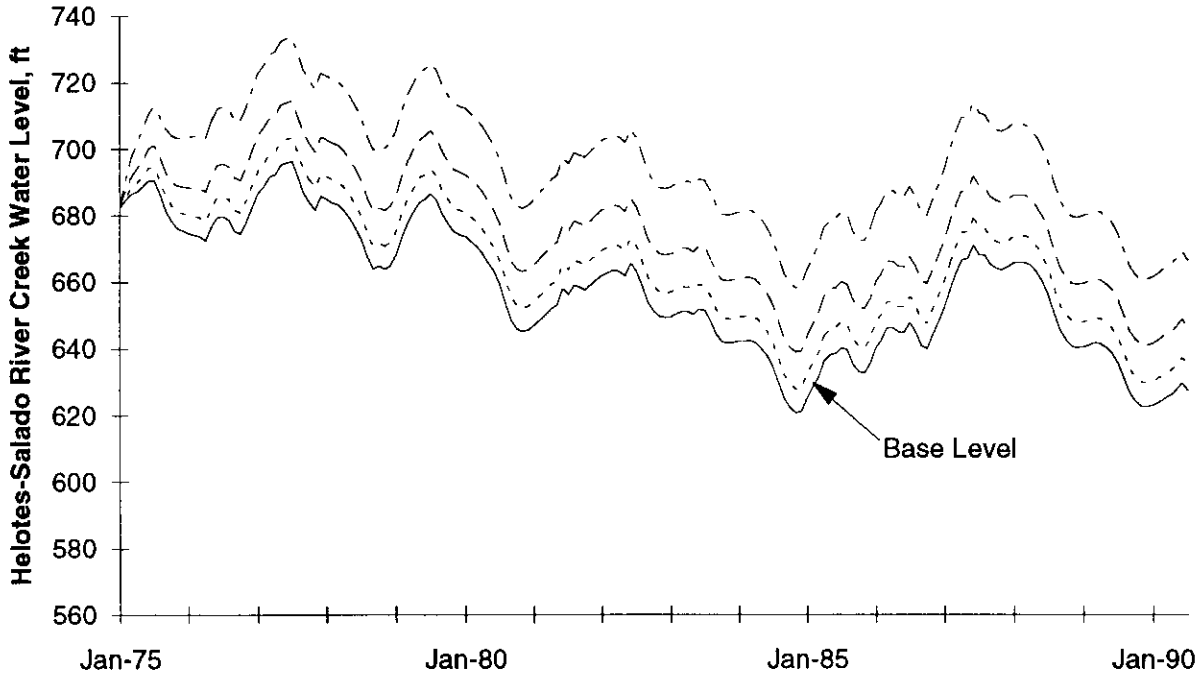


Figure 40. Water levels changes in the Helotes-Salado Creek Basin as all storage related parameters are decreased by 2, 5, and 10%

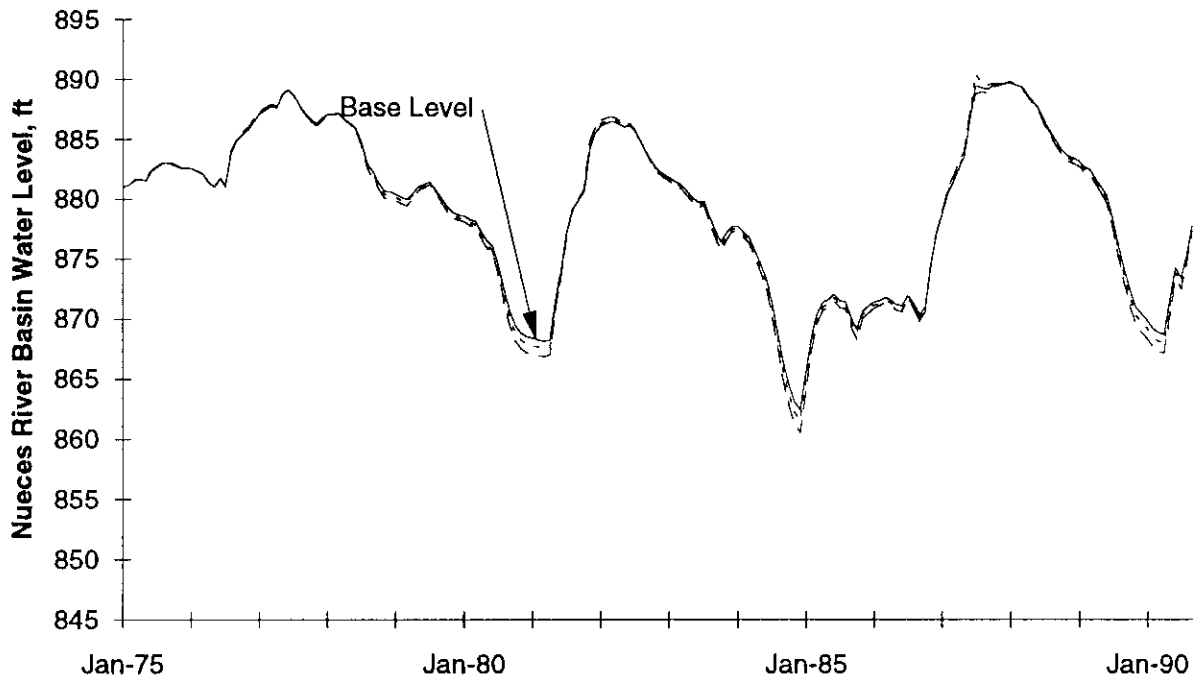


Figure 41. Water levels changes in the Nueces River Basin as its storage related parameters are decreased by 5 and 10%

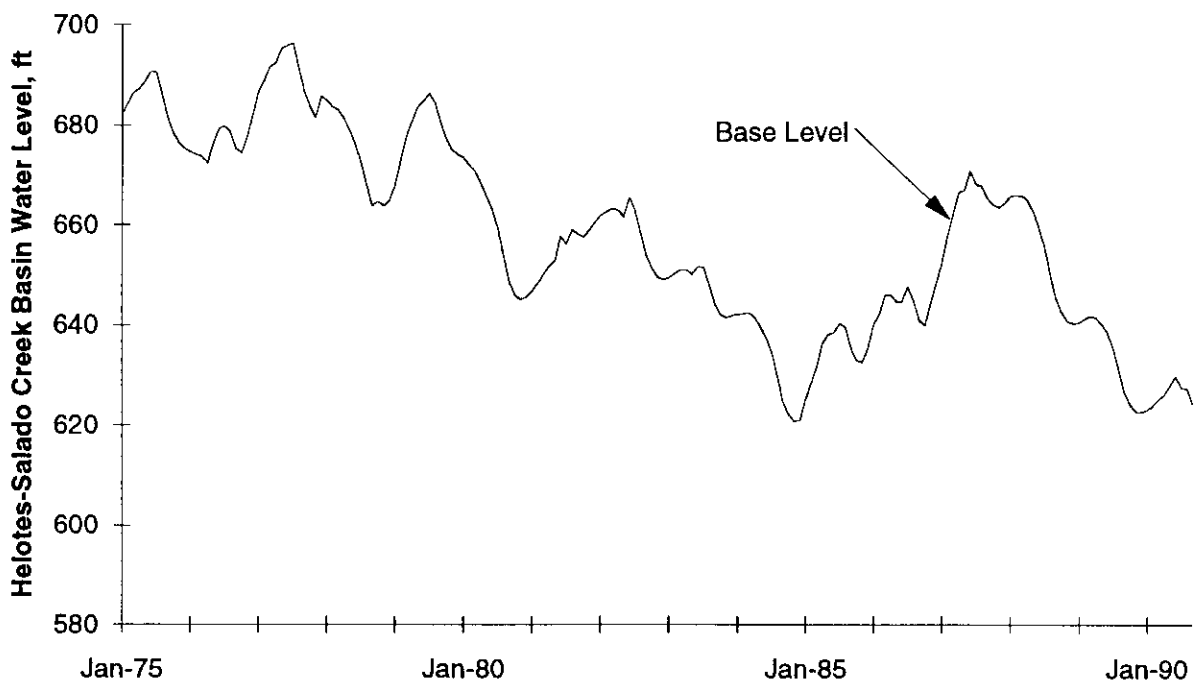


Figure 42. Water levels changes in the Helotes-Salado Creek Basin as the storage related parameters in the Nueces River Basin are decreased by 5 and 10%

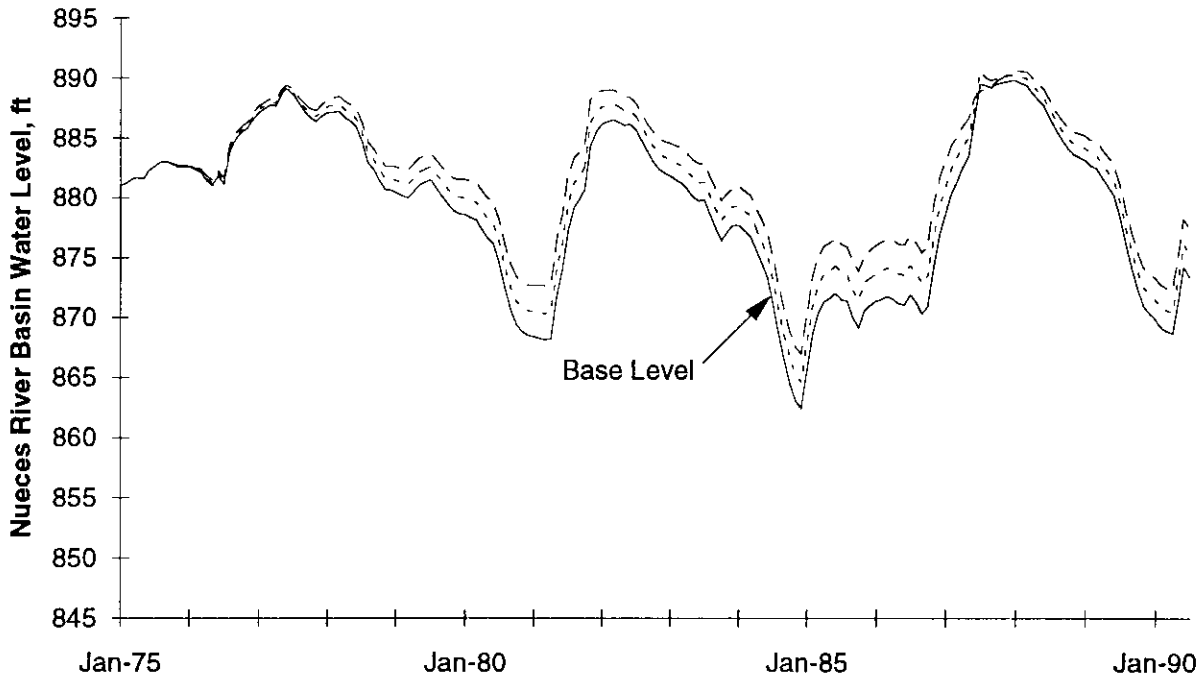


Figure 43. Water levels changes in the Nueces River Basin as the storage related parameters in the Helotes-Salado Creek Basin are decreased by 5 and 10%

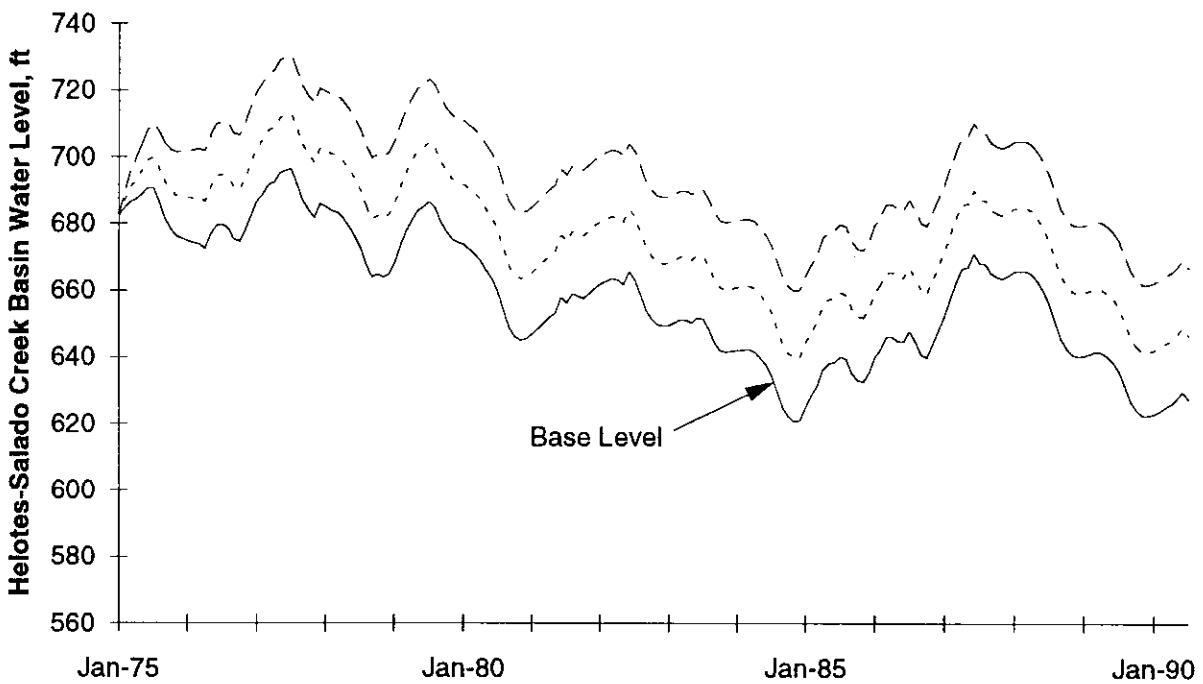


Figure 44. Water levels changes in the Helotes-Salado Creek Basin as its storage related parameters are decreased by 5 and 10%

SUMMARY AND CONCLUSIONS

A lumped parameter, conceptual tank-based model has been successfully applied and calibrated to the Edwards Aquifer. However, refinement and calibration should be done as soon as new data sets become available. The methodology has proved to be very efficient and provides good accuracy. Of 189 months, water levels in nine sub-basins and springflows at the Comal and San Marcos Springs were simulated in less than four minutes on an 68040 based microcomputer. The model is very easy to use since much less data is required, i.e., the average physical and hydrologic data for only nine basins is needed. We have planned to re-code the model as a spreadsheet function, which will allow the model to run inside a spreadsheet.

The success of model calibration suggests that the recharge functions incorporated in the model provide good estimates of monthly recharge. In addition, when summed to annual recharge estimates, the monthly recharge estimates from the recharge functions were very similar to recharge estimates by the USGS except for peak years. The method for estimating monthly recharge developed for the lump parameter model may be preferable to other methods for estimating recharge because the recharge functions **account for recharge capacity, water levels, and are less complex** than existing methods for estimating recharge.

Future research to improve model performance should include (1) the use of the Laplace transform as a solution method, (2) the generalization of the model for use with other karst aquifers, (3) exploration of ways to improve model accuracy such as; better recharge estimation, especially in the ungaged sub-basins of the Helotes, Salado and Dry Comal Creeks; re-delineating sub-basins to better represent regional flow patterns in the recharge zone; and implementing a non-linear regression for springflows and water levels, especially in the low flow regime.

Using the model for future research and study is suggested in different areas. First, the model can be easily integrated with a rainfall-runoff model so that the gage flows, which currently are inputs to the model, can be predicted from rainfall. Such an extension will be compatible with the use of rainfall to determine future pumpage management. Second, with the aid of a multi-site flow generator, reliabilities of aquifer management alternatives can be assessed. Using a stochastic model, water managers can explore the risk of various scenarios and make decisions without jeopardizing springflows below the regulated levels. Third, an aquifer

management model can be formulated using an optimization-simulation model. As a result, management alternatives can be evaluated and the best management plan can be selected. Finally, there will be a need for refining the model, perhaps to a daily operational model for use in the real-time pumpage regulation.

REFERENCES

Choffel, K. L. and S. K. Vaugh, 1993, Historical Recharge of the Edwards Aquifer in the Nueces, San Antonio, and Guadalupe River Basins. HDR Engineering, Inc. Proceedings: American Institute of Hydrology and American Water Resources Association Texas Sections Joint Meeting and Texas Hydrology Roundup, April 16,1993.

Garza, Sergio, 1966, Ground-Water Resources of the San Antonio Area, Texas, a progress report on studies, 1960-1964, Texas Water Development Board Report No. 34, Austin, Texas.

Klemt, W. B., T. R. Knowles, G. R. Elder, and T. W. Sieh, 1979, Ground-Water Resources and Model Applications for the Edwards (Balcones Fault Zone) Aquifer in the San Antonio Region, Texas. Texas Department of Water Resources Report, No. 239, Austin, Texas.

Lasdon, L. S., and A. D. Waren, 1986, "GRG2 User's Guide," School of Business Administration, University of Texas at Austin, Austin, Texas.

Lowry, R. L., 1955, Recharge to the Edwards Ground-Water Reservoir, Consulting Engineer Report to the San Antonio City Water Board, San Antonio, Texas.

Maclay, R. W. and L. F. Land, 1988, Simulation of Flow in the Edwards Aquifer, San Antonio Region, Texas, and Refinement of Storage and Flow Concepts. USGS Water- Supply Paper No. 2336-A, Denver, Colorado.

McDonald, M. G., and Harbaugh, A. W., 1988, A Modular Three-Dimensional Finite-Difference Ground-Water Flow Model, Techniques of Water Resources Investigations of the USGS, Book 6, Chapter A1, Denver, Colorado.

Prickett, T. A., and C. G. Lonnquist, 1971, Selected Digital Computer Techniques for Groundwater Resource Evaluation. Illinois State Water Survey Bulletin No. 55, Urbana, Illinois.

Puente, C, 1975, Relation of Precipitation to Annual Ground-Water Recharge in the Edwards Aquifer, San Antonio, Texas, USGS Open-File Report 75-298, Austin, Texas.

Puente, C, 1978, Method of Estimating Natural Recharge to the Edwards Aquifer in the San Antonio Area, Texas. USGS Water-Resources Investigations 78-10, Austin, Texas.

Sage, A. P., and J. L. Melsa, 1976, *Estimation Theory with Applications to Forecasting and Control*, McGraw-Hill, New York.

Takahashi, Y., M. J. Rabins, and D. M. Auslander, 1972, *Control and Dynamic Systems.* Addison-Wesley Publishing Co., Reading, Massachusetts, 2nd. edition.

Thorkildsen, D. and P. D. McElhaney, 1992, Model Refinement and Applications for the Edwards (Balcones Fault Zone) Aquifer in the San Antonio Region, Texas. Texas Water Development Board, Report No. 340, Austin, Texas.

Trescott, P. C., G. F. Pinder, and S. P. Larson, 1976, Finite-Difference Model for Aquifer Simulations in Two Dimensions with Results of Numerical Experiments. Techniques of Water-Resources Investigations of the USGS. Book 7, Chapter C1, Reston, Virginia.

van der Heijde, P. K. M., I. E. Aly, and S. A. Williams, 1988, "Groundwater Modeling: An Overview and Status Report," U. S. EPA 600/2-89/028, R. S. Kerr Environmental Research Laboratory, Ada, Oklahoma.

Wokhour, S., 1993, (Hydrogeologist: Edwards Underground Water District). Personal communication, August 1993.

APPENDIX A: USER'S MANUAL

The computer program for the model was written in FORTRAN programming language and was compiled using the MacFORTRAN/MPW. A portion of the code dealing with the graphic interface may have to be modified when it is transported to other machines. Data input for the program can be divided into five sets including the problem specification, the tank parameters, the link parameters, time-step (hydrologic and pumpage) data, and the output specifications. Most input is in FORTRAN free-format with at least one blank, or comma, or tab separating each field. A formatted input will require a FORTRAN format specification at the first line indicating the format of the data set to be followed. The following Table explains data requirements for each data set assumming the problem has k tanks, l links, and t time-steps.

No. of Lines	Field#	Description
I. Problem Specification		
1	1 2 3 4	Number of tanks (sub-basins) = k Number of links = l Number of time-step = t Problem specification code which is the addition of the following codes: 1 for graphic output (instead of numerical output) 2 for displaying recharge (instead of basin underflow) 8 for echoing data input (instead of silent data input)
II. Tank Parameters		
k	 1 2 3 4 5 6	Each line contains data input for a tank with the following fields: A tank label, must be an integer A storage parameter constant, S' The average elevation of the aquifer base under this basin The average elevation of the aquifer top under this basin A power of the storage function, n A base elevation in the storage function, x_0 Note that the storage function is given in equation (11) as: $\forall(x) = S'(x-x_0)^n$

III. Link Parameters		
ℓ		Each line contains input data for a link with the following fields:
	1	The label of the header tank
	2	The label of the tailer tank
	3	A flow transmissivity parameter, which is equal to Kw/L, where K is the hydraulic conductivity, and w/L is the ratio of average width to length of the flow path.
	4	An alpha coefficient for leakage or springflow.
	5	A beta coefficient for leakage or springflow.
IV. Hydrologic and Time-Step Data		
1	1	An alphanumeric string of the data set format specification
t		Each line contains time-step data with a date label field and four k-field groups. Fields in each group are arranged in the order of basins entered in the Data Set II.
	1	Date label
	1st k-field	Observed water levels for each basin; used for comparison with the simulation
	2nd k-field	Inflow data for each basin; used in computing recharge for each basin
	3rd k-field	Pumpage data for each basin
	4th k-field	Observed underflow (springflow) data for each basin, intended for comparison with simulated springflows
V. Output Specification		
1		The number of fields in this data set varies and depends on the type of output requested
		For numerical output, this line contains one field of
	1	an alphanumeric string specifying output format
		For graphic output, this line contains two fields of the 32-bit binary string as follows:
	1	Each bit in the string represents an option whether to plot a water level graph for the corresponding tank.
	2	Each bit in the string represents an option whether to plot a hydrograph of the underflow in the corresponding link.
		Note that the order of the bits are the same as the order of tanks or the order of links entered in the Data Sets II and III, respectively.

corresponding units for volume storage, elevation, and time must be in 10^3 ac-ft, ft msl, and month, respectively. For instance, the flow transmissivity parameter must have the unit of 10^3 acre/ft/month. Thus, for a hydraulic conductivity of 1 ft/s the equivalent transmissivity parameter in 10^3 ac/ft/month is

$$\left(\frac{T}{b}\right) = 0.06033\left(\frac{Kw}{L}\right)$$

provided that w and L are in the same length unit.

An example of the data input is given in Figure B1 for a problem containing 9 tanks, 11 links, and 189 time-steps. Figure B2 details the fields for each data set. Graphic and numerical outputs for the problem are shown in Figures B3 and B4, respectively.

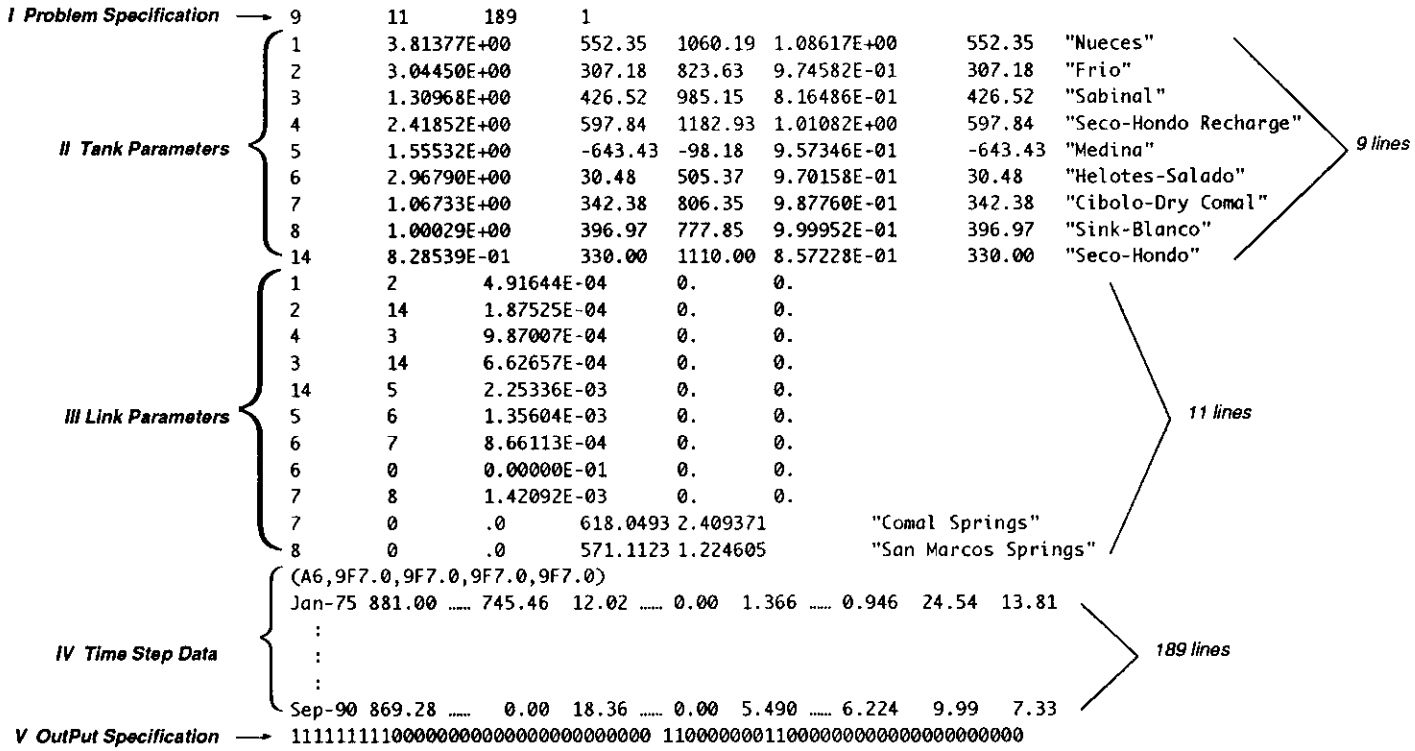


Figure B1. An example of data input

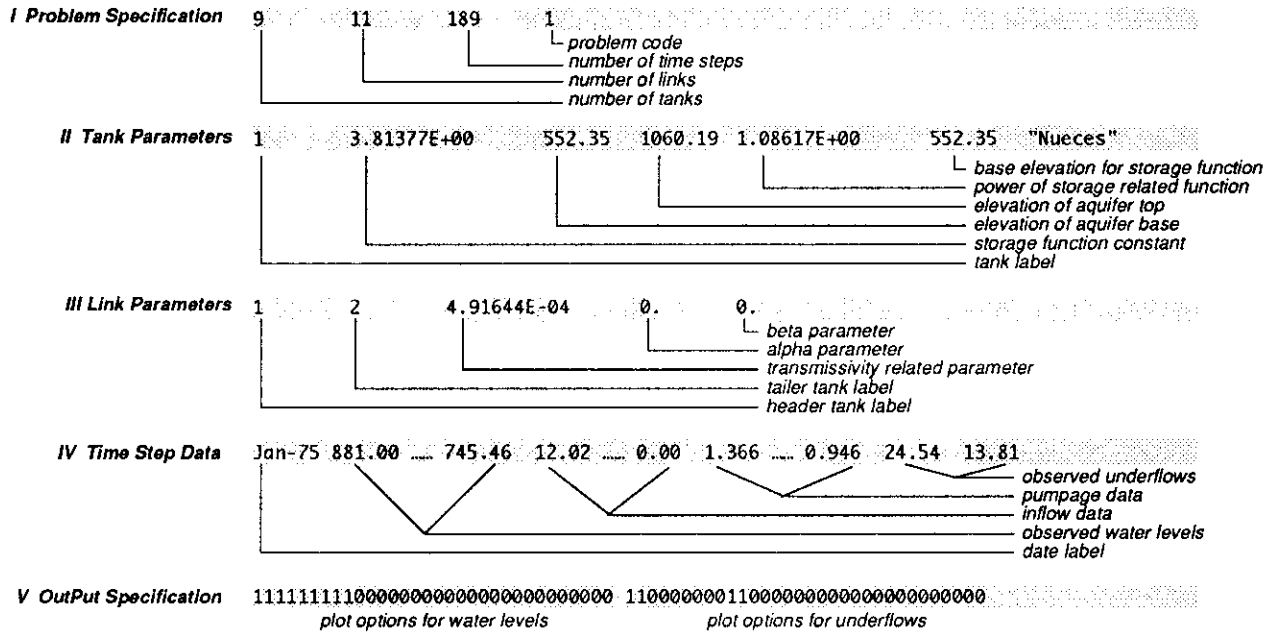


Figure B2. Detailed description of data input fields

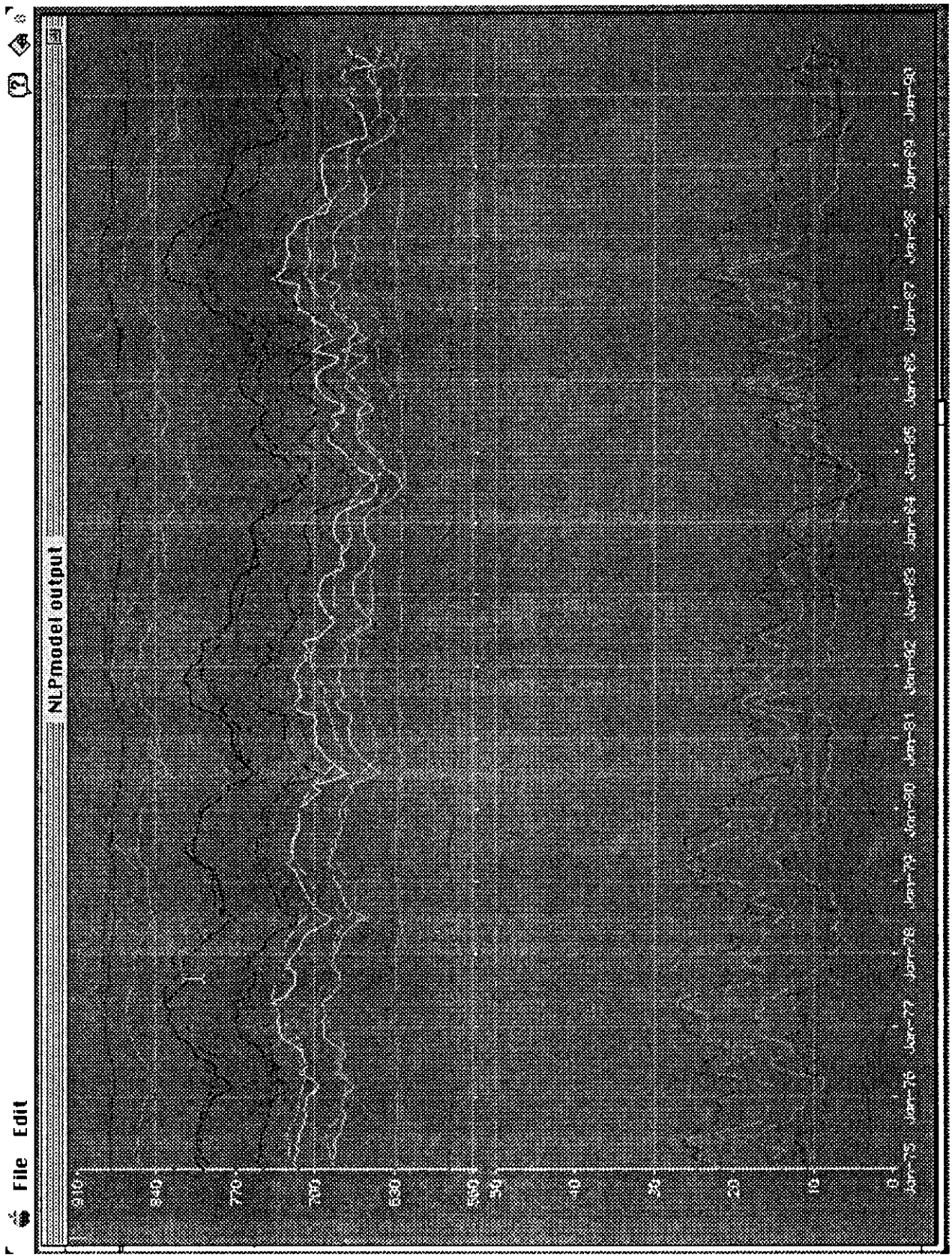


Figure B3. Example of graphic output

APPENDIX B: COMPUTER PROGRAM LISTING

The source code contains in 7 files as follows:

NLPmodel.f	contains main program
NLPmodel.inc	include file
NLPsubs.f	contains subroutines and functions for the simulation model
lossfn.f	contains subroutines and functions to estimated recharge
matrix.f	contains subroutines and functions for matrix operations
NLPgcomp.f	contains subroutines for interfacing with optimizer (GRG2) for parameter estimation
Kalman.f	contains subroutines and functions to perform Kalman filtering
NLPout.f	contains subroutines and functions to produce numerical and graphical outputs

FILE: NLPmodel.f

```
PROGRAM MAIN
IMPLICIT NONE
REAL*8 G(100),X(100)
DATA X(1)/9999./

CALL GCOMP(G,X)
PAUSE
STOP
END
```

NLPmodel.inc

```
! Use structure instead of COMMON for passing arrays
STRUCTURE /NLPARR/
   REAL*8 WL(1,1);    POINTER(p_WL,WL)      ! Observed Water level matrix
   REAL*8 WLC(1,1);   POINTER(p_WLC,WLC)    ! Calculated Water Level matrix
   REAL*8 FW(1,1);    POINTER(p_FW,FW)      ! Inflow matrix
   REAL*8 PP(1,1);    POINTER(p_PP,PP)      ! Pumpage matrix
   REAL*8 RE(1,1);    POINTER(p_RE,RE)      ! Recharge matrix
   REAL*8 U(1,1);     POINTER(p_U,U)        ! Input matrix
   REAL*8 O(1,1);     POINTER(p_O,O)        ! Output matrix
   REAL*8 SFW(1,1);   POINTER(p_SFW,SFW)    ! Scratch matrix - Obs. springflows
   INTEGER*4 LA(1);   POINTER(p_LA,LA)      ! Tank labels
   INTEGER*4 LI(1,1);     POINTER(p_LI,LI)      ! Tank links
   CHARACTER*9 DA(1);     POINTER(p_DA,DA)      ! Date string
   REAL*8 AL(1);      POINTER(p_AL,AL)      ! Boundary ALPHA vector
   REAL*8 BE(1);      POINTER(p_BE,BE)      ! Boundary BETA vector
   REAL*8 T(1);       POINTER(p_T,T)        ! Transmissivity vector
   REAL*8 S(1);       POINTER(p_S,S)        ! Storativity vector
   REAL*8 TO(1);      POINTER(p_TO,TO)      ! Top of aquifer vector
   REAL*8 A(1,1);     POINTER(p_A,A)        ! Parameter matrix A
   REAL*8 B(1,1);     POINTER(p_B,B)        ! Parameter matrix B
   REAL*8 C(1,1);     POINTER(p_C,C)        ! Parameter matrix C
   REAL*8 D(1,1);     POINTER(p_D,D)        ! Parameter matrix D
   REAL*8 P(1,1);     POINTER(p_P,P)        ! Transformation matrix P
   REAL*8 Q(1,1);     POINTER(p_Q,Q)        ! Transformation matrix Q
   REAL*8 W(1,1);     POINTER(p_W,W)        ! Working matrix
   REAL*8 W1(1,1);    POINTER(p_W1,W1)      ! Working matrix
   REAL*8 W2(1,1);    POINTER(p_W2,W2)      ! Working matrix
   REAL*8 WLA(1);     POINTER(p_WLA,WLA)    ! Temporary storage during iterations
   REAL*8 WKAL(1);    POINTER(p_WKAL,WKAL)  ! Working storage for Kalman filter
   INTEGER*4 ICODE
END STRUCTURE

INTEGER*4 INIBIT; PARAMETER (INIBIT=Z'80000000')   !Initialize step
INTEGER*4 GRGBIT; PARAMETER (GRGBIT=Z'40000000')   !Parameter optimization run
INTEGER*4 XLLBIT; PARAMETER (XLLBIT=Z'20000000')   !Running under Excel
INTEGER*4 KALBIT; PARAMETER (KALBIT=Z'00000010')   !Implement Kalman filter
INTEGER*4 ECHBIT; PARAMETER (ECHBIT=Z'00000008')   !Input data echo
INTEGER*4 GPHBIT; PARAMETER (GPHBIT=Z'00000001')   !Graphic output
INTEGER*4 RCHBIT; PARAMETER (RCHBIT=Z'00000002')   !Recharge output
INTEGER*4 AM1BIT; PARAMETER (AM1BIT=Z'00010000')   !Aquifer management option 1
INTEGER*4 AM2BIT; PARAMETER (AM2BIT=Z'00020000')   !Aquifer management option 2
```

FILE: NLPsubs.f

```
C***************************************
C   Macintosh toolbox include files
C***************************************
    GLOBAL DEFINE
        INCLUDE "NLPmodel.inc"
    END



C***************************************
C   Solve for next system state
C***************************************
    SUBROUTINE NX_STATE(T,S,XB,SN,SX0,X,FW,PP,RE,U,O,XAVG,KZ,N,NL,NT)
    IMPLICIT NONE
    INTEGER*4 KZ,N,NL,NT,I,K,KP1
    REAL*8 T(NL),S(N),XB(N),SN(N),SX0(N)
    REAL*8 X(N,NT+1),FW(N,NT),PP(N,NT),RE(N,NT),U(N,NT),O(NL,NT),XAVG(N)
    RECORD /NLPARR/ Z
    POINTER (p_Z,Z)
    REAL*8 DT
    REAL*8 TOL,TOLX,ERRSUM,XTEMP
    INTEGER*4 MAXIT,J
    PARAMETER (TOL=0.0001, MAXIT=10)

    p_Z = KZ
    DT = 1.0

C   Solve for [X(t+1)]T = [X(t)]T [P(t)]T + [U(t)]T [Q(t)]T
    DO K = 1,NT
C       For non-linear system must iterate for X(t+1) until
C       the sum of errors between iterations is within tolerant limit (TOLX), or
C       the maximum number of iterations (MAXIT) is reached

C       Kalman Filter - predefined weights
        CALL KALMAN(Z.WL,X(1,K),Z.WKAL,KZ,N,K)

C       First set estimate X(t+1) = X(t) and compute TOLX
        KP1 = K + 1
        TOLX = 0.0
        DO I = 1,N
            X(I,KP1) = X(I,K)
            TOLX = TOLX + X(I,K)
        END DO
        TOLX = TOLX*TOL
C       print *, "Time step ",K

C       Start iteration loop
        DO J = 1,MAXIT

C        Recompute T & S from water level at the middle of time step (XAVG)
         CALL PARAMS(T,S,XB,Z.TO,SN,SX0,Z.LA,Z.LI,Z.T,Z.S,X(1,K),XAVG,KZ,N,NL)

C        A,B,C,D are parameter matrices to determine next system state and output
C        Construct parameter matrices of A and B
         CALL GETAB(Z.T,Z.S,Z.LA,Z.LI,Z.AL,Z.BE,Z.A,Z.B,XAVG,U(1,K),KZ,N,NL)

C        Solve for matrix exponential P and Q
         CALL SOLVEP(Z.P,Z.Q,Z.A,Z.B,Z.W,Z.W1,Z.W2,N,DT)

C        Compute input from loss function
```

```
        CALL GETU(FW(1,K),PP(1,K),RE(1,K),XAVG,U(1,K),Z.LA,KZ,N)

C       Solve the equation
        CALL MMULT(X(1,K),Z.P,X(1,KP1),1,N,N)
        CALL MMULTPC(U(1,K),Z.Q,X(1,KP1),1,N,N)

C       Check iteration tolerant limit and saved XAVG(I)
        ERRSUM = 0.0
        DO I = 1,N
            XTEMP = ( X(I,K) + X(I,KP1) )/2.0    ! new XAVG
            ERRSUM = ERRSUM + DABS(XTEMP-XAVG(I))
            XAVG(I) = XTEMP
        END DO
C       print *, "    Iteration",J," error = ",ERRSUM
        IF (ERRSUM .LE. TOLX) EXIT

       END DO  ! end iteration loop

C      Calculate outflow in each link
       CALL GETOL(Z.T,Z.S,Z.LA,Z.LI,Z.AL,Z.BE,O(1,K),Z.C,Z.D,XAVG,U(1,K),KZ,N,NL)

    END DO ! end time step loop

    RETURN
    END


C***************************************
C   Update model parameters
C***************************************
    SUBROUTINE PARAMS(T,S,XB,XT,SN,SX0,LA,LI,TT,SS,X,XAVG,KZ,N,NL)
    IMPLICIT NONE
    INTEGER*4 KZ,N,NL
    INTEGER*4 LA(N),LI(2,NL)
    REAL*8 T(NL),S(N),XB(N),XT(N),SN(N),SX0(N)
    REAL*8 TT(NL),SS(N),X(N),XAVG(N)
    RECORD /NLPARR/ Z
    POINTER (p_Z,Z)
    REAL*8 TFN,SFN
    INTEGER*4 I,I1,I2,L,LA2I

    p_Z = KZ

    DO I = 1,N
        XAVG(I) = (X(I) + X(I+N))/2.0        ! time average
        SS(I) = SFN(S(I),SN(I),SX0(I),XT(I),XAVG(I))
    END DO
    DO L = 1,NL
        I1 = LA2I(LI(1,L),LA,N)
        I2 = LA2I(LI(2,L),LA,N)
        IF (LI(1,L) .LE. 0 .OR. I1 .EQ. 0) THEN
            TT(L) = TFN(T(L),XB(I2),XT(I2),XAVG(I2))
        ELSE IF (LI(2,L) .LE. 0 .OR. I2 .EQ. 0) THEN
            TT(L) = TFN(T(L),XB(I1),XT(I1),XAVG(I1))
        ELSE
            TT(L) = TFN(T(L),XB(I1),XT(I1),(XAVG(I1)+XAVG(I2))/2.0)
        END IF
    END DO
    RETURN
    END


C***************************************
C   Transmisivity functions
```

```
C*****************************************
      REAL*8 FUNCTION TFN(T,XB,XT,X)
      REAL*8 T,XB,XT
      REAL*8 X

      IF (T==0.) THEN    !Some boundary conditions, springs, leakage
         TFN = 0.
      ELSE           !Linear system if XT-XB is used
         TFN = T*DMAX1(0.D1,DMIN1(X-XB,XT-XB))
      END IF
      RETURN
      END


C*****************************************
C   Storativity functions
C*****************************************
      REAL*8 FUNCTION SFN(S,SN,SX0,XT,X)
      REAL*8 S,SN,SX0,XT
      REAL*8 X

      IF (SN==1.0) THEN     !Linear system
         SFN = S
      ELSE
         SFN = S*SN*(DMAX1(1.0D-7,DMIN1(X-SX0,XT-SX0))**(SN-1.0))
      ENDIF
      RETURN
      END


C*****************************************
C   Paynter's algotithm to solve for matrices P and Q
C*****************************************
      SUBROUTINE SOLVEP(P,Q,A,B,W,W1,W2,N,DT)
      IMPLICIT NONE
      INTEGER N
      REAL*8 P(N,N),Q(N,N),A(N,N),B(N,N),W(N,N),W1(N,N),W2(N,N),DT
      INTEGER NP,I,J,K
      REAL*8 AQ,TERM1,TERM2,AK

C   A,B are parameter matrices to determine next system state
C   W matrix temporarily hold Q matrix elements during series expansion
C   W1 and W2 matrices temporarily and alternately hold intermediate terms
C   during series expansion to solve for P and Q

C   Find q = max[A(i,j)*Δt]
C   Also redefine A(i,j) = A(i,j)*Δt and B(i,j) = B(i,j)*Δt
      AQ = 0.0
      DO I = 1,N
      DO J = 1,N
         A(I,J) = A(I,J)*DT
         AQ = MAX(AQ,ABS(A(I,J)))
      END DO
      END DO

C   Determine pth truncation term from (1/p!)(nq)^p e^(nq) = 0.001
      TERM1 = REAL(N)*AQ
      TERM2 = EXP(TERM1)
      NP = 0
      DO WHILE (TERM2>0.001.AND.NP<=100)
         NP = NP+1
         TERM2 = TERM2*TERM1/NP
      END DO
```

```
C   Find matrices P and Q; in transpose forms
C   P = I + AΔt + (AΔt)^2/2! + (AΔt)^3/3! + ... +(AΔt)^p/p!
C   Q = Δt[I + AΔt/2! + (AΔt)^2/3! + (AΔt)^3/4! + ... +(AΔt)^p/(p+1)!] * B
C   Or rather
C   [P]T = [I + AΔt + (AΔt)^2/2! + (AΔt)^3/3! + ... +(AΔt)^p/p!]T
C   [Q]T = [BΔt]T * [I + AΔt/2! + (AΔt)^2/3! + (AΔt)^3/4! + ... +(AΔt)^p/(p+1)!]T
C   Initialize P,Q & W
    DO I = 1,N
    DO J = 1,N
        W1(I,J) = A(I,J)/2.0
        IF (I==J) THEN
            P(I,J) = 1.0 + A(I,J)
            W(I,J) = 1.0 + W1(I,J)
        ELSE
            P(I,J) = A(I,J)
            W(I,J) = W1(I,J)
        END IF
    END DO
    END DO
    DO K = 2,NP
        AK = 1./REAL(K+1)
        IF(MOD(K,2)==0) THEN
            CALL MMULTP(A,W1,P,W2,N)
            CALL MMULTQ(W,W2,N,AK)
        ELSE
            CALL MMULTP(A,W2,P,W1,N)
            CALL MMULTQ(W,W1,N,AK)
        END IF
    END DO
    CALL MMULT(B,W,Q,N,N,N)
    RETURN
    END


C**************************************
C   Matrix multiplication
C**************************************
    SUBROUTINE MMULT(A,B,C,NA,NB,NC)
    IMPLICIT NONE
    INTEGER NA,NB,NC,I,J,K
    REAL*8 A(NA,NB),B(NB,NC),C(NA,NC)

C   FIND C(NA,NC) = A(NA,NB) * B(NB,NC)
    DO I=1,NA
    DO J=1,NC
        C(I,J) = 0.
        DO K=1,NB
            C(I,J) = C(I,J)+A(I,K)*B(K,J)
        END DO
    END DO
    END DO
    RETURN

    ENTRY MMULTPC(A,B,C,NA,NB,NC)
C   FIND C(NA,NC) = C(NA,NC) + A(NA,NB) * B(NB,NC)
    DO I=1,NA
    DO J=1,NC
        DO K=1,NB
            C(I,J) = C(I,J)+A(I,K)*B(K,J)
        END DO
    END DO
    END DO
    RETURN
    END
```

```
C**************************************
C   Special matrix multiplication to be used in SOLVEP
C**************************************
    SUBROUTINE MMULTP(A,B,C,W,NA)
    IMPLICIT NONE
    INTEGER NA,I,J,K
    REAL*8 A(NA,NA),B(NA,NA),C(NA,NA),W(NA,NA)

C   FIND W(NA,NA) = A(NA,NA) * B(NA,NA)
C   FIND C(NA,NA) = C(NA,NA) + W(NA,NA)
    DO I=1,NA
    DO J=1,NA
        W(I,J) = 0.0
        DO K=1,NA
            W(I,J) = W(I,J) + A(I,K) * B(K,J)
        END DO
        C(I,J) = C(I,J) + W(I,J)
    END DO
    END DO
    RETURN
    END

    SUBROUTINE MMULTQ(A,W,NA,AK)
    IMPLICIT NONE
    INTEGER NA,I,J
    REAL*8 A(NA,NA),W(NA,NA),AK
C   FIND W(NA,NA) = W(NA,NA) * AK
C   FIND A(NA,NA) = A(NA,NA) + W(NA,NA)
    DO I=1,NA
    DO J=1,NA
        W(I,J) = W(I,J) * AK
        A(I,J) = A(I,J) + W(I,J)
    END DO
    END DO
    RETURN
    END


C**************************************
C   Construct parameter matrices A and B
C**************************************
    SUBROUTINE GETAB(TT,SS,LA,LI,AL,BE,A,B,WL,U,KZ,N,NL)
    IMPLICIT NONE
    INTEGER*4 KZ,N,NL
    INTEGER*4 LA(N),LI(2,NL)
    REAL*8 TT(NL),SS(N),A(N,N),B(N,N),WL(N),U(N),AL(NL),BE(NL)
    RECORD /NLPARR/ Z
    POINTER (p_Z,Z)
    INTEGER I,J,L,I1,I2,LA2I

    p_Z = KZ
C   Parameters depend on system configuration
C   A,B are parameter matrices to determine next system state
C   ***A,B are computed in transpose order***
    DO J = 1,N      !col is tank row
        DO I = 1,N      !row is variable column
            A(I,J) = 0.
            B(I,J) = 0.
        END DO
        B(J,J) = 1.0/SS(J)
        U(J) = 0.

C       Scan link list
```

```
C       Link to outbound basin will be treat as boundary conditions
        DO L = 1,NL
            I1 = LA2I(LI(1,L),LA,N)
            I2 = LA2I(LI(2,L),LA,N)
C           Prevailing condition - upstream
            IF (LI(1,L) .LE. 0 .OR. I1 .EQ. 0) THEN
               IF (TT(L) .EQ. 0.) THEN
                A(I2,I2) = A(I2,I2) - BE(L)
                U(I2) = U(I2) - AL(L)
               ELSE
                A(I2,I2) = A(I2,I2) - (1.0-BE(L))*TT(L)
                U(I2) = U(I2) + AL(L)*TT(L)
               END IF
C           Terminal condition, leakage, or spring flows
            ELSE IF (LI(2,L) .LE. 0 .OR. I2 .EQ. 0) THEN
               IF (TT(L) .EQ. 0.) THEN
C               Springflow - only if WL > elevation of spring openings
C               IF(WL(I1) .GT. AL(L)) THEN
                   A(I1,I1) = A(I1,I1) - BE(L)
                   U(I1) = U(I1) + AL(L)*BE(L)
C               ENDIF
               ELSE
                A(I1,I1) = A(I1,I1) - (1.0-BE(L))*TT(L)
                U(I1) = U(I1) + AL(L)*TT(L)
               END IF
            ELSE IF (LI(1,L) .EQ. LA(J)) THEN
                A(J,J) = A(J,J) - TT(L)
                A(I2,J) = A(I2,J) + (1.0-BE(L))*TT(L)   ! For free-fall set BE(L)=1
            ELSE IF (LI(2,L) .EQ. LA(J)) THEN
                A(J,J) = A(J,J) - (1.0-BE(L))* TT(L)    ! For free-fall set BE(L)=1
                A(I1,J) = A(I1,J) + TT(L)
            END IF
        END DO

C       Divided thru by storage parameter
        DO I = 1,N     !row is variable column
           A(I,J) = A(I,J)/SS(J)
        END DO
    END DO

    RETURN
    END


C***************************************
C   Determine flow in links
C***************************************
    SUBROUTINE GETOL(TT,SS,LA,LI,AL,BE,OL,C,D,WL,U,KZ,N,NL)
    IMPLICIT NONE
    INTEGER*4 KZ,N,NL
    INTEGER*4 LA(N),LI(2,NL)
    REAL*8 TT(NL),SS(N),OL(NL),C(NL,N),D(NL,N),WL(N),U(N),AL(NL),BE(NL)
    RECORD /NLPARR/ Z
    POINTER (p_Z,Z)
    INTEGER L,I1,I2,LA2I
C   INTEGER*4 BITTST

    p_Z = KZ
C   Trap for parameter optimization
C   IF ( BITTST(Z.ICODE,GRGBIT) ) RETURN

C   C, D, and U are provided for future uses
C   Scan link list
C   Link to outbound basin will be treat as boundary conditions
```

```
      DO L = 1,NL
         I1 = LA2I(LI(1,L),LA,N)
         I2 = LA2I(LI(2,L),LA,N)
C        Upstream boundary
         IF (LI(1,L) .LE. 0 .OR. I1 .EQ. 0) THEN
            IF (TT(L) .EQ. 0.) THEN
             OL(L) = AL(L) + BE(L)*WL(I2)
            ELSE
             OL(L) = ( AL(L) - (1.0-BE(L))*WL(I2) )*TT(L)
            END IF
C        Downstream boundary, leakage, or spring flows
         ELSE IF (LI(2,L) .LE. 0 .OR. I2 .EQ. 0) THEN
            IF (TT(L) .EQ. 0.) THEN
C            OL(L) = DMAX1( 0.D1, BE(L)*(WL(I1)-AL(L)) )
             OL(L) = BE(L)*(WL(I1)-AL(L))
            ELSE
             OL(L) = ( -AL(L) + (1.0-BE(L))*WL(I1) )*TT(L)  ! For free-fall set BE(L)=AL(L)=0.
            END IF
         ELSE
             OL(L) = ( WL(I1) - WL(I2) )* TT(L)
         END IF
      END DO

      RETURN
      END


C**************************************
C   Get tank index from tank label
C**************************************
      INTEGER*4 FUNCTION LA2I(L,LA,N)
      INTEGER*4 L,LA(*),N,I

      I = N
      DO WHILE (I .GT. 0 .AND. L .NE. LA(I))
         I = I - 1
      END DO
      LA2I = I
      RETURN
      END


C**************************************
C   Compute system input U
C**************************************
      SUBROUTINE GETU(FW,PP,RE,WL,U,LA,KZ,N)
      IMPLICIT NONE
      INTEGER*4 KZ,N
      INTEGER*4 LA(N)
      REAL*8 FW(N),PP(N),RE(N),WL(N),U(N),LOSS
      RECORD /NLPARR/ Z
      POINTER (p_Z,Z)
      INTEGER I
      REAL*8 PMPCTL
C     Flow interpolation ratio
      REAL*8 CB2CB; PARAMETER (CB2CB=274./68.4)   ! Interpolate Cibolo @Selma from Cibolo @Boerne
      REAL*8 CB2HS; PARAMETER (CB2HS=314./68.4)   ! Interpolate Cibolo @Helotes-Salado from Cibolo
@Boerne
      REAL*8 GL2DC; PARAMETER (GL2DC=130./82.)    ! Interpolate Dry Comal from Guadalupe
      REAL*8 GL2YK; PARAMETER (GL2YK=92./82.)     ! Interpolate York-Purgetory-Sink from Guadalupe

      p_Z = KZ
C     Compute system input from loss functions
      DO I = 1,N
```

```
        SELECT CASE (LA(I))
           CASE (1:4)
              RE(I) = LOSS(LA(I),FW(I),WL(I))
           CASE (5)
              !Function input Medina lake vol., previous vol.
              RE(I) = LOSS(LA(I),FW(I),FW(I-N))
           CASE (6)
              !Function input Cibolo @Helotes-Salado, 2nd parmeter not used
              RE(I) = LOSS(LA(I),FW(I)*CB2HS,WL(I))
           CASE (7)
              !Function input Cibolo @Selma, Guadalupe @Dry Comal
              RE(I) = LOSS(LA(I),FW(I-1)*CB2CB,FW(I)*GL2DC)
           CASE (8)
              !Function input Blanco, Guadalupe @York-Purgetory-Sink
              RE(I) = LOSS(LA(I),FW(I),FW(I-1)*GL2YK)
           CASE DEFAULT
              RE(I) = 0.0
        END SELECT

C       Subtract pumpage and add boundary conditions
        U(I) = U(I) + RE(I) - PMPCTL(PP,RE,WL,LA,KZ,N,I)
     END DO

     RETURN
     END


C**************************************
C   Pumpage regulation
C**************************************
     REAL*8 FUNCTION PMPCTL(PP,RE,WL,LA,KZ,N,I)
     IMPLICIT NONE
     INTEGER*4 KZ,N,I
     INTEGER*4 LA(N)
     REAL*8 PP(N),RE(N),WL(N)
     RECORD /NLPARR/ Z
     POINTER (p_Z,Z)
     INTEGER*4 BITTST

     p_Z = KZ

C    Pumpage control alternatives
     SELECT CASE (BITTST(Z.ICODE,Z'000F0000'))
        CASE (AM1BIT)
C       Across-the-board cutback alternative
        IF (WL(6) .LT. 633.00) THEN
           PMPCTL = PP(I)*0.6
        ELSE IF (WL(6) .LT. 650.00) THEN
           PMPCTL = PP(I)*0.8
        ELSE IF (WL(6) .LE. 666.00) THEN
           PMPCTL = PP(I)*0.9
        ELSE
           PMPCTL = PP(I)
        END IF
        CASE (AM2BIT)
C       Dry-year option
        IF (WL(6) .LT. 633.00) THEN
           SELECT CASE (I)
              CASE (1:5,9)
               PMPCTL = PP(I)*0.50
              CASE (6:8)
               PMPCTL = PP(I)*0.70
              CASE DEFAULT
               PMPCTL = PP(I)*0.50
```

```
         END SELECT
      ELSE IF (WL(6) .LE. 649.00) THEN
         SELECT CASE (I)
            CASE (1:5,9)
             PMPCTL = PP(I)*0.70
            CASE (6:8)
             PMPCTL = PP(I)*0.85
            CASE DEFAULT
             PMPCTL = PP(I)*0.70
         END SELECT
      ELSE
         PMPCTL = PP(I)
      END IF
     CASE DEFAULT
C    No control
      PMPCTL = PP(I)
  END SELECT

  RETURN
  END


C**************************************
C  Special bit manipulation functions
C**************************************
  INTEGER*4 FUNCTION BITSET(FLAG,MASK)
  INTEGER*4 FLAG,MASK

  BITSET = FLAG .OR. MASK
  RETURN

  ENTRY BITCLR(FLAG,MASK)
  BITSET = FLAG .AND. (MASK .XOR. Z'FFFFFFFF')
  RETURN

  ENTRY BITTST(FLAG,MASK)
  BITSET = FLAG .AND. MASK
  RETURN
  END
```

FILE: lossfn.f

```
C-----------------------------------THIS PORTION FOR EXCEL FUNCTION ONLY----------------------

  GLOBAL DEFINE
     INCLUDE "xlcall.inc"
  END


  PASCAL SUBROUTINE LOSSFN(PIB,PFW,PWL)
  IMPLICIT NONE
  INTEGER*4 PIB,PFW,PWL
  VALUE PIB,PFW,PWL     ! this argument is passed by value
  RECORD /fp/ IB,FW,WL
  POINTER (P_IB,IB),(P_FW,FW),(P_WL,WL)
  INTEGER*4 N,I,J
  REAL*8 LOSS

  P_IB = PIB
  P_FW = PFW
  P_WL = PWL
```

```
      N = MIN(FW.nc,WL.nc)
      IF (N==1) THEN    ! Single basin computation, must explicitly input required flow
         DO I = 1,MIN(FW.nr,WL.nr)
            J = IB.arr(1)
            WL.arr(I) = LOSS(J,FW.arr(I),WL.arr(I))
         END DO
      ELSE           ! multiple basin computation, use LOSS0 to interpolate flow
         DO I = 1,MIN(FW.nr,WL.nr)
            J = N*(I-1)+1
            CALL LOSS0(IB.arr,FW.arr(J),WL.arr(J),N)
         END DO
      END IF

      RETURN
      END


      SUBROUTINE LOSS0(IB,FW,WL,N)
      IMPLICIT NONE
      INTEGER*4 N
      REAL*8 IB(N),FW(N),WL(N),LOSS
      INTEGER I,KBASIN

C     Flow interpolation ratio
      REAL*8 CB2CB; PARAMETER (CB2CB=274./68.4)   ! Interpolate Cibolo @Selma from Cibolo @Boerne
      REAL*8 CB2HS; PARAMETER (CB2HS=314./68.4)   ! Interpolate Cibolo @Helotes-Salado from Cibolo
                                                  ! @Boerne
      REAL*8 GL2DC; PARAMETER (GL2DC=130./82.)    ! Interpolate Dry Comal from Guadalupe
      REAL*8 GL2YK; PARAMETER (GL2YK=92./82.)     ! Interpolate York-Purgetory-Sink from Guadalupe

C     Compute system input from loss functions
      DO I = 1,N
         KBASIN = IB(I)
         SELECT CASE (KBASIN)
            CASE (5)
               !Function input Medina lake vol., previous vol.
               WL(I) = LOSS(KBASIN,FW(I),FW(I-N))
            CASE (6)
               !Function input Cibolo @Helotes-Salado, 2nd parmeter not used
               WL(I) = LOSS(KBASIN,FW(I)*CB2HS,WL(I))
            CASE (7)
               !Function input Cibolo @Selma, Guadalupe @Dry Comal
               WL(I) = LOSS(KBASIN,FW(I-1)*CB2CB,FW(I)*GL2DC)
            CASE (8)
               !Function input Blanco, Guadalupe @York-Purgetory-Sink
               WL(I) = LOSS(KBASIN,FW(I),FW(I-1)*GL2YK)
            CASE DEFAULT
               WL(I) = LOSS(KBASIN,FW(I),WL(I))
         END SELECT
      END DO
      RETURN
      END
C-------------------------------------END OF PORTION FOR EXCEL FUNCTION-------------------------


      REAL*8 FUNCTION LOSS(IBASIN,INFLOW,WATLEV)
      IMPLICIT NONE
      INTEGER*4 IBASIN
C     INFLOW = inflow in thousand ac-ft/month
C     WATLEV = water level in ft msl.
C            = inflow from secondary basin, use for interpolating
      REAL*8 INFLOW,WATLEV

      REAL*8 ZERO,ONE
```

```
      PARAMETER (ZERO=0.0, ONE=1.0)
      REAL*8 LOSSRAT,A,B,TERM1

C     Lossfn parameter
      REAL*8 NU_MAX; PARAMETER(NU_MAX=29.0)              ! Nueces River Basin
      REAL*8 A_FO,B_FO; PARAMETER (A_FO=-1.725814,B_FO=4.9575271)   ! Frio River Basin
      REAL*8 A_SL,B_SL; PARAMETER (A_SL=-2.063892,B_SL=3.6371286)   ! Sabinal River Basin
      REAL*8 A_SH,B_SH; PARAMETER (A_SH=-2.776139,B_SH=5.36413)     ! Seco-Hondo River Basin
      REAL*8 A_HS,B_HS; PARAMETER (A_HS=1.,B_HS=0.)          ! Helotes-Salado River Basin
      REAL*8 A_CB,B_CB; PARAMETER (A_CB=-1.840337,B_CB=4.449842)    ! Cibolo River Basin
      REAL*8 DC_MAX; PARAMETER(DC_MAX=20.0)              ! Dry Comal River Basin
      REAL*8 BC_MAX; PARAMETER(BC_MAX=1.331)             ! Blanco River Basin

C     Adjacent area ratios
      REAL*8 AR_FO; PARAMETER(AR_FO=1.+90./691.)
      REAL*8 AR_SL; PARAMETER(AR_SL=1.+24./241.)
      REAL*8 AR_SH; PARAMETER(AR_SH=1.+164./317.)

C     Flow interpolation ratio
      REAL*8 CB2CB; PARAMETER (CB2CB=274./68.4)   ! Interpolate Cibolo @Selma from Cibolo @Boerne
      REAL*8 CB2HS; PARAMETER (CB2HS=314./68.4)   ! Interpolate Cibolo @Helotes-Salado from Cibolo
                                                  ! @Boerne
      REAL*8 GL2DC; PARAMETER (GL2DC=130./82.)    ! Interpolate Dry Comal from Guadalupe
      REAL*8 GL2YK; PARAMETER (GL2YK=92./82.)     ! Interpolate York-Purgetory-Sink from Guadalupe

C     General form of loss-ratio function
      LOSSRAT(A,B) = MIN(ONE, MAX(ZERO, (LOG(INFLOW)-B)/A) )    ! 0 ≤ Ratio ≤ 1

      IF (INFLOW==ZERO) THEN
         LOSS = ZERO
      ELSE
         SELECT CASE (IBASIN)

C        Nueces River Basin
         CASE (1)
            TERM1 = MIN(ONE, 37.5902/(927.6461-WATLEV))
            TERM1 = ONE - MAX(ZERO, TERM1)**3.6814
            LOSS = MIN(NU_MAX, TERM1*INFLOW)

C        FRIO River Basin
         CASE (2)
            TERM1 = LOSSRAT(A_FO,B_FO)*INFLOW
            LOSS = TERM1*AR_FO

C        Sabinal River Basin
         CASE (3)
            TERM1 = LOSSRAT(A_SL,B_SL)
            IF (TERM1 > 0.9 .AND. WATLEV < 824.0)
     1         TERM1 = ONE-4.6786/(870.0-WATLEV)
            LOSS = TERM1*INFLOW*AR_SL

C        Seco-Hondo River Basin
         CASE (4)
            TERM1 = LOSSRAT(A_SH,B_SH)*INFLOW
            LOSS = TERM1*AR_SH

C        Medina River Basin
         CASE (5)
            ! INFLOW = current reservoir level
            ! WATLEV = previous reservoir level
            TERM1 = 263.8/(INFLOW-1.533)-1.0
            TERM1 = 2.996498972/TERM1**0.294548213
            IF (WATLEV>=INFLOW) THEN
               TERM1 = TERM1 + 1.0
```

```
            ELSE
                TERM1 = TERM1 + 3.380840153
            END IF
            LOSS = TERM1

C       Helotes-Salado River Basin
        CASE (6)
            !INFLOW = Cibolo flow @Helotes-Salado
            TERM1 = LOSSRAT(A_CB,B_CB)*INFLOW       !Use Cibolo loss rate
            LOSS = TERM1

C       Cibolo-Dry Comal River Basin
        CASE (7)
            !INFLOW = Cibolo flow @Selma (no loss)
            !WATLEV = Guadalupe flow @Dry Comal
            TERM1 = LOSSRAT(A_CB,B_CB)*INFLOW       !Cibolo creek loss
            LOSS = TERM1 + MIN(DC_MAX,WATLEV)       !+Dry Comal creek loss

C       Blanco River Basin
        CASE (8)
            !WATLEV = Guadalupe flow @York-Purgetory-Sink-Alligator
            TERM1 = MIN(BC_MAX,INFLOW)          !Blanco river loss
            LOSS = TERM1 + MIN(DC_MAX,WATLEV)       !+York creek loss

        CASE DEFAULT   ! Required in MacFORTRAN for stand alone code
                       ! otherwise will cause error in linker
        END SELECT
    END IF

    RETURN
    END
```

FILE: matrix.f

```
C****************************************
C   Matrix addition: C = A + B
C****************************************
    SUBROUTINE MADD(A,B,C,NR,NC)
    IMPLICIT NONE
    INTEGER NR,NC,I,J
    REAL*8 A(NR,NC),B(NR,NC),C(NR,NC)
    DO I = 1,NR
    DO J = 1,NC
       C(I,J) = A(I,J) + B(I,J)
    END DO
    END DO
    RETURN
    END


C****************************************
C   Matrix subtraction: C = A - B
C****************************************
    SUBROUTINE MSUBT(A,B,C,NR,NC)
    IMPLICIT NONE
    INTEGER NR,NC,I,J
    REAL*8 A(NR,NC),B(NR,NC),C(NR,NC)
    DO I = 1,NR
    DO J = 1,NC
       C(I,J) = A(I,J) - B(I,J)
    END DO
```

```
    END DO
    RETURN
    END


C**************************************
C   Matrix unity: I
C**************************************
    SUBROUTINE MUNIT(A,NA)
    IMPLICIT NONE
    INTEGER NA,I,J
    REAL*8 A(NA,NA)
    DO I = 1,NA
    DO J = 1,NA
       A(I,J) = 0.0
    END DO
       A(I,I) = 1.0
    END DO
    RETURN
    END


C**************************************
C   Matrix multiplication C = AB'
C**************************************
    SUBROUTINE MMULTRAN(A,B,C,NA,NB,NC)
    IMPLICIT NONE
    INTEGER NA,NB,NC,I,J,K
    REAL*8 A(NA,NB),B(NC,NB),C(NA,NC)

C   FIND C(NA,NC) = A(NA,NB) * B'(NC,NB)
    DO I=1,NA
    DO J=1,NC
        C(I,J) = 0.
        DO K=1,NB
            C(I,J) = C(I,J)+A(I,K)*B(J,K)
        END DO
    END DO
    END DO
    RETURN
    END


C**************************************
C   Matrix inversion
C**************************************
    SUBROUTINE MINVER(A,AI,na)
    IMPLICIT NONE
    INTEGER*4 na
    REAL*8 A(na,na),AI(na,na)
    INTEGER*4 ip(200),j
C-----------------------------------------------------------------------
C Subroutine to invert a matrix use gauss0 which solve system of
C simultaneous equation by LU factorization
C-----------------------------------------------------------------------
    call MUNIT(AI,na)
    call gauss0(A,AI(1,1),na,ip,0)
    do (j=2,na)
    call gauss1(A,AI(1,j),na,ip,1)
    repeat
    end

    SUBROUTINE gauss0(A,b,na,ip,op)
    IMPLICIT NONE
```

```
    INTEGER*4 na
    REAL*8 A(na,na),b(*),sum
    INTEGER*4 ip(na,2),p,i,j,k,k1,k2,op,irow,ipivot
C-----------------------------------------------------------------------
C Subroutine to solve system of simultaneous equation
C by LU factorization
C Upon return A contain LU and b contain solution
C U include diagonal elements where L contain only lower triangle
C elements.  Note that diagonal elements of L is 1
C ip(i,1) is the pivot sequence for row i
C op is the option, specify .t. if this is the only one time solution
C of matrix A, otherwise M(na-1)*...*M(2)*M(1) will be computed for
C later use of A with different b vector
C-----------------------------------------------------------------------

C initialize pivot sequence
    do (i = 1,na)
       ip(i,1) = 0
    repeat

C Fill up multipliers M(1),M(2),...,M(na-1) and compute U
    do (k = 1,na-1)           ! M(1),M(2),...,M(na-1)
      p = ipivot(A,k,na,ip)       ! Find the next row to pivot
      do (i = 1,na)           ! row of M(k)
        if (ip(i,1) == 0) then    ! not yet pivot?
          A(i,k)=-A(i,k)/A(p,k)   ! negative multipliers
          do (j = k+1,na)         ! column of U
       A(i,j) = A(i,k)*A(p,j)+A(i,j)
          repeat
          b(i) = A(i,k)*b(p)+b(i)
        endif
      repeat
    repeat
    do (i = 1,na)
      if (ip(i,1) == 0) then
        ip(i,1) = na
        exit
      endif
    repeat
    if (op == 0) then
C Compute M = M(na-1)*...*M(3)*M(2)*M(1) for later use to solve
C for difference b vector but the same A matrix
    do (k = 2,na-1)       ! M(2) to M(na-1)
      k1 = irow(k,na,ip)
      do (j = 1,k-1)      ! column to be updated (previous M)
        do ( k2 = k+1,na)         ! row to be updated
          i = irow(k2,na,ip)
          A(i,j) = A(i,k)*A(k1,j)+A(i,j)
        repeat
      repeat
    repeat
    endif
    goto 10

    entry gauss1(A,b,na,ip,op)
C Apply M to the new b vector
    do (k = na,2,-1)
      i = irow(k,na,ip)
      do (j = 1,k-1)
        b(i) = b(i)+A(i,j)*b(irow(j,na,ip))
      repeat
    repeat
C Solve for X
10  i = irow(na,na,ip)
```

```
    b(i) = b(i)/A(i,na)
    do (k = na-1,1,-1)
      i = irow(k,na,ip)
      sum = 0.0
      do (j = k+1,na)
        sum = sum+A(i,j)*b(irow(j,na,ip))
      repeat
      b(i) = (b(i)-sum)/A(i,k)
    repeat
C Put solution inplace
    do (i = 1,na)
      ip(i,2) = ip(i,1)
    repeat
    call sort(ip(1,2),b,na)
    end

    SUBROUTINE sort(ip,b,na)
    IMPLICIT NONE
    INTEGER*4 ip(*),na,iptemp,i,k
    REAL*8 b(*),btemp
    do (k=1,na-1)
      btemp = b(k); iptemp = ip(k)
      do (i = k,na)
        if (ip(i) == k) exit
      repeat
      b(k) = b(i); ip(k) = ip(i)
      b(i) = btemp; ip(i) = iptemp
    repeat
    end

    INTEGER*4 FUNCTION ipivot(A,j,na,ip)
    IMPLICIT NONE
    INTEGER*4 na,ip(*),i,j
    REAL*8 A(na,na),AA,AMX
C Function row index of the next pivot sequence in the part of a column
C It also fillup ip array of the determined sequence
    AMX = 0.0
    do (i=1,na)
      if (ip(i) == 0) then
        AA = dabs(A(i,j))
        if (AA > AMX) then
          ipivot = i
          AMX = AA
        endif
      endif
    repeat
    ip(ipivot) = j
C   if (AMX == 0.0) print *,'Matrix is singular'
    end

    INTEGER*4 FUNCTION irow(iseq,na,ip)
    INTEGER*4 iseq,ip(*)
C Function to return row index of iseq
    do (irow = 1,na)
      if (ip(irow) == iseq) exit
    repeat
    end
```

FILE: NLPgcomp.f

```
C**************************************
C  Macintosh toolbox include files
C**************************************
    GLOBAL DEFINE
        INCLUDE "Types.inc"
        INCLUDE "Memory.inc"
        INCLUDE "NLPmodel.inc"
    END


    SUBROUTINE GCOMP(G,X)
    IMPLICIT NONE
    REAL*8 G(*),X(*)
    INTEGER MYFLAG,N,NL,NT
    RECORD /NLPARR/ Z
    POINTER (p_Z,Z)
    INTEGER*4 ICODE,BITSET,BITCLR,BITTST
    SAVE MYFLAG,N,NL,NT,p_Z

C   Initialize step
    IF (MYFLAG<>9999) THEN
C       Create a structure and use instead of COMMON block - Macintosh Fortran Only
        p_Z = NewPtr(VAL(SIZEOF(/NLPARR/)))

        Z.ICODE = 0
        Z.ICODE = BITSET(Z.ICODE,INIBIT)            ! Set initialize step code
        IF (X(1) .EQ. 9999.) THEN
            Z.ICODE = BITCLR(Z.ICODE,GRGBIT)        ! Call from NLPmodel
        ELSE
            Z.ICODE = BITSET(Z.ICODE,GRGBIT)        ! Set code for calling from GRG2
        END IF

C       Read input data
C       First read problem size info for determining array variable sizes
        OPEN (99,FILE='',STATUS='OLD')
        READ (99,*) N,NL,NT,ICODE
C       Set flag for input data echo and output types
        Z.ICODE = BITSET(Z.ICODE,ICODE)
        IF (BITTST(Z.ICODE,ECHBIT)) THEN
           WRITE (*,*) '---------- ECHO INPUT ----------'
           WRITE (*,'(4I5)') N,NL,NT,ICODE
        END IF

C       Initialize array addresses and allocate dynamic memory
        CALL IA_GCOMP(N,NL,NT,p_Z)

C       Initialize variables
        CALL IV_GCOMP(X(1),X(NL+1),X(NL+N+1),X(NL+2*N+1),X(NL+3*N+1),Z.TO,
    1       Z.LA,Z.LI,Z.AL,Z.BE,Z.WL,Z.FW,Z.PP,Z.WLC,Z.SFW,Z.DA,Z.WKAL,p_Z,N,NL,NT)

    ELSE
        Z.ICODE = BITCLR(Z.ICODE,INIBIT)     ! Clear initialize step code
    END IF

C   Call Discrete Lumped Parameter model
    CALL NX_STATE(X(1),X(NL+1),X(NL+N+1),X(NL+2*N+1),X(NL+3*N+1),
    1   Z.WLC,Z.FW,Z.PP,Z.RE,Z.U,Z.O,Z.WLA,p_Z,N,NL,NT)

C   Output - compute Obj Fn | Print results | Show graphical result
```

```
      CALL NLP_OUT(G,X,Z.WLC,Z.WL,Z.PP,Z.RE,Z.U,Z.O,Z.DA,N,NL,NT,p_Z)
      MYFLAG = 9999
      RETURN
      END


C**************************************
C   Initialize GCOMP routine
C**************************************
      SUBROUTINE IA_GCOMP(N,NL,NT,KZ)
      IMPLICIT NONE
      INTEGER*4 KZ,N,NL,NT
      RECORD /NLPARR/ Z
      POINTER (p_Z,Z)
      INTEGER*4 FPSIZE;     PARAMETER(FPSIZE = 8)     !Floating point word size
      INTEGER*4 ISIZE

      p_Z = KZ

C   Array variable addresses
C   Input-output matrices
      ISIZE = N*NT*FPSIZE
      Z.p_WL  = NewPtr(VAL(ISIZE))
      Z.p_FW  = NewPtr(VAL(ISIZE))
      Z.p_PP  = NewPtr(VAL(ISIZE))
      Z.p_RE  = NewPtr(VAL(ISIZE))
      Z.p_U       = NewPtr(VAL(ISIZE))
      Z.p_SFW     = NewPtr(VAL(ISIZE))

      ISIZE = NL*NT*FPSIZE
      Z.p_O       = NewPtr(VAL(ISIZE))

      ISIZE = N*(NT+1)*FPSIZE
      Z.p_WLC     = NewPtr(VAL(ISIZE))

C   Temporary hold system parameter
      Z.p_DA  = NewPtr(VAL(NT*9))       ! Date label 9 characters
      Z.p_LA  = NewPtr(VAL(N*4))        ! use 4 byte-integer
      Z.p_LI  = NewPtr(VAL(NL*2*4))        ! use 4 byte-integer
      Z.p_AL  = NewPtr(VAL(NL*FPSIZE))
      Z.p_BE  = NewPtr(VAL(NL*FPSIZE))
      Z.p_T       = NewPtr(VAL(NL*FPSIZE))
      Z.p_S       = NewPtr(VAL(N*FPSIZE))
      Z.p_TO  = NewPtr(VAL(N*FPSIZE))
      Z.p_WLA     = NewPtr(VAL(N*FPSIZE))

C   System equations matrices
      ISIZE = N*N*FPSIZE
      Z.p_A       = NewPtr(VAL(ISIZE))
      Z.p_B       = NewPtr(VAL(ISIZE))
      Z.p_P       = NewPtr(VAL(ISIZE))
      Z.p_Q       = NewPtr(VAL(ISIZE))
      Z.p_W       = NewPtr(VAL(ISIZE))

C   Kalman matrices
      Z.p_WKAL    = NewPtr(VAL(ISIZE*5))    !Five NxN matrices

      ISIZE = N*NL*FPSIZE
      Z.p_C       = NewPtr(VAL(ISIZE))
      Z.p_D       = NewPtr(VAL(ISIZE))

C   Working matrices
      Z.p_W1  = NewPtr(VAL(ISIZE))
      Z.p_W2  = NewPtr(VAL(ISIZE))
```

```
      RETURN
      END


      SUBROUTINE IV_GCOMP(T,S,XB,SN,SX0,XT,LA,LI,AL,BE,WL,FW,PP,WLC,SFW,DA,WKAL,KZ,N,NL,NT)
      IMPLICIT NONE
      INTEGER*4 KZ,N,NL,NT
      INTEGER*4 LA(N),LI(2,NL)
      REAL*8 T(NL),S(N),XB(N),XT(N),SN(N),SX0(N),AL(NL),BE(NL)
      REAL*8 WL(N,NT),FW(N,NT),PP(N,NT),WLC(N,NT+1),SFW(N,NT)
      REAL*8 WKAL(*)
      CHARACTER*9 DA(NT)
      RECORD /NLPARR/ Z
      POINTER (p_Z,Z)
      INTEGER*4 I,K,BITTST
      REAL*8 DUMMY
      CHARACTER*80 FMT

      p_Z = KZ

C     Read parameters
      IF (.NOT. BITTST(Z.ICODE,GRGBIT)) THEN
C        Read tanks' parameters
         DO I = 1,N
         READ (99,*) LA(I),        ! Tank label
     1          S(I),          ! Storage parameter
     2          XB(I),      ! Aquifer base under the basin
     2          XT(I),      ! Aquifer top under the basin
     3          SN(I),      ! Power of  storage term, for linear in S, SN = 1.
     4          SX0(I)      ! threshold elevation for storage
         IF (BITTST(Z.ICODE,ECHBIT)) WRITE (*,'(I5,5E15.7)') LA(I),S(I),XB(I),XT(I),SN(I),SX0(I)
         END DO
C        Read basin links
         DO K = 1,NL
         READ (99,*) LI(1,K),      ! Head basin index
     1          LI(2,K),       ! Tail basin index
     2          T(K),          ! Transmissivity parameter
     3          AL(K),      ! ALPHA parameter for boundary conditions
     4          BE(K)          ! BETA parameter for boundary conditions
         IF (BITTST(Z.ICODE,ECHBIT)) WRITE (*,'(2I5,3E15.7)') LI(1,K),LI(2,K),T(K),AL(K),BE(K)
         END DO

      ELSE                      ! call from GRG's GCOMP, some data will be ignored
         DO I = 1,N
         READ (99,*) LA(I),DUMMY,DUMMY,XT(I)
         IF (BITTST(Z.ICODE,ECHBIT)) WRITE (*,'(I5,5E15.7)') LA(I),S(I),XB(I),XT(I),SN(I),SX0(I)
         END DO
         DO K = 1,NL
         READ (99,*) LI(1,K),      ! Head basin index
     1          LI(2,K),       ! Tail basin index
     2          DUMMY,      ! Transmissivity parameter
     3          AL(K),      ! ALPHA parameter for boundary conditions
     4          BE(K)          ! BETA parameter for boundary conditions
         IF (BITTST(Z.ICODE,ECHBIT)) WRITE (*,'(2I5,3E15.7)') LI(1,K),LI(2,K),T(K),AL(K),BE(K)
         END DO
      END IF

C     Read water level and inflow
      READ (99,'(A80)') FMT
      DO K = 1,NT
        READ (99,FMT) DA(K),(WL(I,K),I=1,N),(FW(I,K),I=1,N),(PP(I,K),I=1,N),(SFW(I,K),I=1,N)
        IF (BITTST(Z.ICODE,ECHBIT)) WRITE (*,'(A9,30F8.3)')
     DA(K),(WL(I,K),I=1,N),(FW(I,K),I=1,N),(PP(I,K),I=1,N)
      END DO
```

```
C   Load initial state
    DO I = 1,N
       WLC(I,1) = WL(I,1)
    END DO

C   Read variances for Kalman filter
    IF (BITTST(Z.ICODE,KALBIT)) THEN
       DO I=1,N
        READ(99,*) (WKAL(K*N+I),K=0,3*N-1)
        IF (BITTST(Z.ICODE,ECHBIT)) WRITE(9,'(30E12.5)') (WKAL(K*N+I),K=0,3*N-1)
       END DO
    END IF
    RETURN
    END
```

FILE: Kalman.f

```
C****************************************
C   Macintosh toolbox include files
C****************************************
    GLOBAL DEFINE
        INCLUDE "NLPmodel.inc"
    END


C****************************************
C   KALMAN ALGORITHM - main subroutine
C****************************************
    SUBROUTINE KALMAN(WL,X,WKAL,KZ,N,K)
    IMPLICIT NONE
    INTEGER*4 KZ,N,K,I
    REAL*8 WL(N,*),X(N),WKAL(N*N,*)
    RECORD /NLPARR/ Z
    POINTER (p_Z,Z)
    INTEGER*4 BITTST
    REAL*8 XTEMP(10)
    SAVE XTEMP

    p_Z = KZ

    IF (BITTST(Z.ICODE,KALBIT)) THEN
       IF (K .EQ. 1) THEN
C       Kalman Filter - Initializing step
        DO I=1,N
           XTEMP(I) = X(I)
        END DO
        CALL KALUPD(WKAL(1,1),WKAL(1,2),WKAL(1,4),WKAL(1,5),N)
       ELSE
        DO I=1,N
           X(I-N) = XTEMP(I)
           XTEMP(I) = X(I)
        END DO
        CALL KALCOM(WL(1,K),X,Z.P,Z.Q,
    1      WKAL(1,1),WKAL(1,2),WKAL(1,3),WKAL(1,4),WKAL(1,5),N)
       END IF
    END IF
    RETURN
    END
```

```
C**************************************
C   KALUPD updating KALMAN weight
C**************************************
    SUBROUTINE KALUPD(VM,VV,WT,WK,N)
    IMPLICIT NONE
    INTEGER*4 N
    REAL*8 VM(N,N),VV(N,N),WT(N,N),WK(N,N)
    CALL MADD(VM,VV,WT,N,N)
    CALL MINVER(WT,WK,N)
    CALL MMULT(VM,WK,WT,N,N,N)
    RETURN
    END


C**************************************
C   Apply Kalman to the current state
C**************************************
    SUBROUTINE KALCOM(WL,X,P,Q,VM,VV,VW,WT,WK,N)
    IMPLICIT NONE
    INTEGER*4 N,I
    REAL*8 WL(N),X(N),P(N,N),Q(N,N)
    REAL*8 VM(N,N),VV(N,N),VW(N,N),WT(N,N),WK(N,N)

C   Kalman filtering
C   x(k+1) = P(k)x(k) + Qd(k)u(k) + Q(k)w(k);  w(k) is input noise
C   y(k) = C(k)x(k) + v(k);              v(k) is output noise
C   Current algorithm not updating  W = E[ww'] and V = E[vv']
C   C(k) = I
C   VM = E[xx']
C   VV = V (Covariance of output noise)
C   VW = W (Covariance of input noise)
C   WT = K (Kalman's weight)

C   First term for M
    CALL MUNIT(WK,N)          ! I         --> WK
    CALL MSUBT(WK,WT,WT,N,N)  ! (I-K)     --> WT
    CALL MMULT(WT,VM,WK,N,N,N)    ! (I-K).M  --> WK

C   WRITE(9,'(30(E12.5,1H,))')
C   1   (DSQRT(VM(I,I)),I=1,N),(DSQRT(WK(I,I)),I=1,N),(1.-WT(I,I),I=1,N)
    CALL MMULT(P,WK,WT,N,N,N)     ! P.(I-K)M     --> WT
    CALL MMULTRAN(WT,P,VM,N,N,N)     ! P(I-K)M.P'--> VM

C   Second term for M
    CALL MMULTRAN(VW,Q,WK,N,N,N)     ! W.Q' --> WK
    CALL MMULTPC(Q,WK,VM,N,N,N)  ! P(I-K)MP' + Q.WQ' --> VM

C   Updating Kalman weight
    CALL KALUPD(VM,VV,WT,WK,N)    ! M(M+V)'  --> WT

C   Can't change VM and WT from here on
C   CALL MSUBT(WL,X,WK,N,1)       ! y(k)-x(k)    --> WK
C   Need to check missing observations
    DO I = 1,N
       IF (WL(I) .NE. 0.0) THEN
        WK(I,1) = WL(I) - X(I)
       ELSE
        WK(I,1) = 0.0
       END IF
    END DO

    CALL MMULTPC(WT,WK,X,N,N,1)   ! x(k)+K[y(k)-x(k)]

    RETURN
```

```
    END
```

FILE: NLPout.f

```
C****************************************
C   Macintosh toolbox include files
C****************************************
    GLOBAL DEFINE
        INCLUDE "Types.inc"
        INCLUDE "Memory.inc"
        INCLUDE "QuickDraw.inc"
        INCLUDE "Windows.inc"
        INCLUDE "Resources.inc"
        INCLUDE "SysEqu.inc"
        INCLUDE "ToolUtils.inc"
        INCLUDE "SANE.inc"
        INCLUDE "NLPmodel.inc"

        STRUCTURE /NLPGPH/
            REAL*8 WLMAX
            REAL*8 WLMIN
            REAL*8 REMAX
            REAL*8 REMIN
            REAL*8 SCX
            REAL*8 SCY1
            REAL*8 SCY2
            RECORD /Rect/ rect(2)
            INTEGER*4 ICODE(2)
            RECORD /GrafPort/ GRPort
            POINTER (p_GRPort, GRPort)
            RECORD /Rect/ gr_rect
            RECORD /Picture/ axPict      !Axes object
            POINTER (p_axPict, axPict)
            POINTER (h_axPict, p_axPict)
        END STRUCTURE

        INTEGER*4 DForm0;     PARAMETER (DForm0=Z'01000000')

        INTEGER*4 redRGB;     PARAMETER (redRGB=35)
        INTEGER*4 greenRGB;   PARAMETER (greenRGB=185)
        INTEGER*4 blueRGB;    PARAMETER (blueRGB=210)
        INTEGER*4 magentaRGB;     PARAMETER (magentaRGB=30)
        INTEGER*4 yellowRGB;  PARAMETER (yellowRGB=5)
        INTEGER*4 cyanRGB;    PARAMETER (cyanRGB=180)
        INTEGER*4 orangeRGB;  PARAMETER (orangeRGB=23)
        INTEGER*4 ltblueRGB;  PARAMETER (ltblueRGB=198)
        INTEGER*4 purpleRGB;  PARAMETER (purpleRGB=104)
        INTEGER*4 brownRGB;   PARAMETER (brownRGB=17)
        INTEGER*4 dkgreenRGB;     PARAMETER (dkgreenRGB=231)
        INTEGER*4 grayRGB;    PARAMETER (grayRGB=250)
        INTEGER*4 ltgrayRGB;  PARAMETER (ltgrayRGB=249)
        INTEGER*4 dkwhiteRGB;     PARAMETER (dkwhiteRGB=248)
        INTEGER*4 blackRGB;   PARAMETER (blackRGB=255)
        INTEGER*4 whiteRGB;   PARAMETER (whiteRGB=0)
    END


    SUBROUTINE NLP_OUT(G,X,WLC,WL,PP,RE,U,O,DA,N,NL,NT,KZ)
    IMPLICIT NONE
    INTEGER*4 KZ,N,NL,NT
    RECORD /NLPARR/ Z
```

```
      POINTER (p_Z,Z)
      REAL*8 G(*),X(*),SFWOBJ
      REAL*8 WLC(N,NT+1),WL(N,NT),PP(N,NT),RE(N,NT),U(N,NT),O(NL,NT)
      CHARACTER*80 ALPHA
      CHARACTER*9 DA(NT)
      RECORD /NLPGPH/ GR
      POINTER (p_GR,GR)
      INTEGER*4 J,K,BITTST,I2COLOR

      RECORD /Picture/ wdPict
      POINTER (p_wdPict, wdPict)
      POINTER (h_wdPict, p_wdPict)

      INTEGER*1 thePat(0:7)
      SAVE ALPHA,p_GR

      p_Z = KZ

C     Compute objective function
C     G(1) = WLOBJ(WLC,WL,N,NT)
      G(1) = SFWOBJ(VAL(Z.p_SFW),O,N,NL,NT)

      IF (.NOT. BITTST(Z.ICODE,GPHBIT)) THEN

C        Hardcopy output
         WRITE(9,*)
         WRITE(9,*) '---------- LUMPED PARAMETER MODEL OUTPUT ----------'
         IF (BITTST(Z.ICODE,INIBIT)) READ(99,'(A)') ALPHA   !Read output format
         IF (BITTST(Z.ICODE,GRGBIT)) THEN
          WRITE(9,ALPHA) G(1),(X(J),J=1,NL),(X(J),J=NL+1,NL+N*4)
         ELSE
          DO K = 1, NT
             WRITE(9,ALPHA) DA(K),(WLC(J,K),J=1,N),(RE(J,K),J=1,N),(O(J,K),J=1,NL)
          END DO
         END IF

      ELSE

C        Show graphical output
         IF (BITTST(Z.ICODE,INIBIT)) THEN
            p_GR = NewPtr(VAL(SIZEOF(/NLPGPH/)))
            READ(99,'(B32,1x,B32)') GR.ICODE
             CALL NLP_INIGPH(WL,RE,O,N,NL,NT,KZ,p_GR)
         END IF

C        Plot output
         h_wdPict = GetWindowPic(VAL(GR.p_GRPort))
         IF (h_wdPict<>0) CALL KillPicture(VAL(h_wdPict))
         h_wdPict = OpenPicture(GR.gr_rect)
         CALL RGBBackColor(VAL(I2COLOR(grayRGB)))
         CALL EraseRect(GR.gr_rect)
         CALL DrawPicture(VAL(GR.h_axPict),GR.gr_rect)

C        Observed Water levels
         CALL NLP_GRAPH(WL,N,NT,GR.SCX,GR.SCY1,GR.WLMAX,GR.rect(1),GR.ICODE(1))
C        Computed Water levels
         CALL GetIndPattern(thePat,VAL2(sysPatListID),VAL2(4))
         CALL PenPat(thePat)
         CALL NLP_GRAPH(WLC,N,NT,GR.SCX,GR.SCY1,GR.WLMAX,GR.rect(1),GR.ICODE(1))
C        Computed Recharge or underflow
         IF (BITTST(Z.ICODE,RCHBIT)) THEN
            CALL NLP_GRAPH(RE,N,NT,GR.SCX,GR.SCY2,GR.REMAX,GR.rect(2),GR.ICODE(2))
         ELSE
            CALL NLP_GRAPH(O,NL,NT,GR.SCX,GR.SCY2,GR.REMAX,GR.rect(2),GR.ICODE(2))
```

```
C          Observed Springflow
              CALL PenNormal()
           CALL NLP_GRAPH(Z.SFW,N,NT,GR.SCX,GR.SCY2,GR.REMAX,GR.rect(2),GR.ICODE(2))
        END IF

        CALL PenNormal()
        CALL ClosePicture()
        CALL SetWindowPic(VAL(GR.p_GRPort),VAL(h_wdPict))
        CALL DrawPicture(VAL(h_wdPict),GR.gr_rect)
        CALL RGBForeColor(VAL(I2COLOR(whiteRGB)))

      END IF

      RETURN
      END


      SUBROUTINE NLP_INIGPH(WL,RE,O,N,NL,NT,KZ,KGR)
      IMPLICIT NONE
      INTEGER*4 N,NL,NT,KZ,KGR
      RECORD /NLPARR/ Z
      POINTER (p_Z,Z)
      RECORD /NLPGPH/ GR
      POINTER (p_GR,GR)
      REAL*8 WL(N,NT),RE(N,NT),O(NL,NT),YINT
      INTEGER*4 J,K,BITTST,IMASK

      RECORD /Point/ margin
      RECORD /QuickdrawGlobals/ QDGlobals
      POINTER (p_QDGlobals, QDGlobals)
      INTEGER*4 GETA5              ! Inline Function to return current A5
      INLINE(GETA5=z'2e8d')        !  68000: MOVE.L A5,(A7)

      p_Z = KZ
      p_GR = KGR

C     Determine scales
      GR.WLMAX = -1.E+15; GR.WLMIN = 1.E+15
      GR.REMAX = -1.E+15; GR.REMIN = 1.E+15
      DO K = 1,NT
         DO J = 1,N
            IF (.NOT. BITTST(GR.ICODE(1),IMASK(J))) CYCLE
          GR.WLMAX = MAX(GR.WLMAX,WL(J,K))
          IF (WL(J,K) .NE. 0.0) GR.WLMIN = MIN(GR.WLMIN,WL(J,K))    ! 0.0 is missing value
         END DO
         IF (BITTST(Z.ICODE,RCHBIT)) THEN
          DO J = 1,N
             IF (.NOT. BITTST(GR.ICODE(2),IMASK(J))) CYCLE
             GR.REMAX = MAX(GR.REMAX,RE(J,K))
             IF (RE(J,K) .GT. 0.0) GR.REMIN = MIN(GR.REMIN,RE(J,K))     ! 0.0 is missing value
          END DO
         ELSE
          DO J = 1,NL
             IF (.NOT. BITTST(GR.ICODE(2),IMASK(J))) CYCLE
             GR.REMAX = MAX(GR.REMAX,O(J,K))
             IF (O(J,K) .GT. 0.0) GR.REMIN = MIN(GR.REMIN,O(J,K))   ! 0.0 is missing value
          END DO
         END IF
      END DO
      YINT = (INT((GR.WLMAX-GR.WLMIN)/5./10.)+1)*10.     !5 intervals @ modulus of 10
      GR.WLMIN = INT(GR.WLMIN/YINT)*YINT
      GR.WLMAX = GR.WLMIN + YINT*5.
      YINT = (INT((GR.REMAX-GR.REMIN)/5./10.)+1)*10.     !5 intervals @ modulus of 10
      GR.REMIN = INT(GR.REMIN/YINT)*YINT
```

```
      GR.REMAX = GR.REMIN + YINT*5.

      GR.p_GRPort = LONG(GETA5())
      p_QDGlobals = GR.p_GRPort - SIZEOF(/QuickdrawGlobals/)
      GR.gr_rect = QDGlobals.screenBits.bounds
      margin.h = 40;     margin.v = 100
      GR.rect(1).top = 5
      GR.rect(1).left = 50
      GR.rect(2).left = GR.rect(1).left
      GR.SCX = REAL(GR.gr_rect.right-GR.gr_rect.left-2*margin.h)/NT
      GR.rect(2).top = 0.5*REAL(GR.gr_rect.bottom-GR.gr_rect.top-margin.v)
      GR.SCY1 = GR.rect(2).top/(GR.WLMAX-GR.WLMIN)
      GR.SCY2 = GR.rect(2).top/(GR.REMAX-GR.REMIN)
      GR.rect(2).top = GR.rect(1).top+GR.rect(2).top+15

      GR.rect(1).bottom = GR.rect(1).top+NINT((GR.WLMAX-GR.WLMIN)*GR.SCY1)
      GR.rect(1).right = GR.rect(1).left+NINT(NT*GR.SCX)
      GR.rect(2).bottom = GR.rect(2).top+NINT((GR.REMAX-GR.REMIN)*GR.SCY2)
      GR.rect(2).right = GR.rect(1).right

      CALL SetRect(GR.gr_rect,VAL2(-32767),VAL2(-32767),VAL2(32767),VAL2(32767))
      CALL DRWSCL(Z.DA,NT,p_GR)
      GR.p_GRPort = FrontWindow()

      RETURN
      END


      REAL*8 FUNCTION WLOBJ(WLC,WL,N,NT)
      IMPLICIT NONE
      INTEGER*4 N,NT,I,J,K
      REAL*8 WLC(N,NT+1),WL(N,NT),OBJ

C     Compute sum square errors of computed water levels
      OBJ = 0.0D0
      I = 0
      DO K = 1,NT
      DO J = 1,N
          IF (WL(J,K)<>0.0) THEN
              OBJ = OBJ + (WLC(J,K)-WL(J,K))**2
              I = I + 1
          END IF
      END DO
      END DO
      WLOBJ = SQRT(OBJ)/REAL(I)
      RETURN
      END


      REAL*8 FUNCTION SFWOBJ(SFW,O,N,NL,NT)
      IMPLICIT NONE
      INTEGER*4 N,NL,NT,K
      REAL*8 SFW(N,NT),O(NL,NT),OBJ
      INTEGER*2 COMAL,SAN_MARCOS
      PARAMETER (COMAL = 10, SAN_MARCOS = 11)

C     Compute sum square errors of computed water levels
      OBJ = 0.0D0
      DO K = 1,NT
          OBJ = OBJ + (SFW(1,K)-O(COMAL,K))**2
          OBJ = OBJ + (SFW(2,K)-O(SAN_MARCOS,K))**2
      END DO
      SFWOBJ = SQRT(OBJ)/REAL(NT*2)
      RETURN
```

```
END


SUBROUTINE NLP_GRAPH(Y,N,NT,SCX,SCY,YMAX,IAXIS,ICODE)
IMPLICIT NONE
INTEGER*4 N,NT
INTEGER*2 IAXIS(4)
REAL*8 Y(N,NT),SCX,SCY,YMAX
INTEGER*4 J,K,IX,IY,ICODE
INTEGER*4 I2COLOR,BITTST,IMASK

DO J = 1,N
   IF (.NOT. BITTST(ICODE,IMASK(J))) CYCLE
   SELECT CASE (MOD(J,12))
   CASE (1)
       CALL RGBForeColor(VAL(I2COLOR(redRGB)))
   CASE (2)
       CALL RGBForeColor(VAL(I2COLOR(greenRGB)))
   CASE (3)
       CALL RGBForeColor(VAL(I2COLOR(blueRGB)))
   CASE (4)
       CALL RGBForeColor(VAL(I2COLOR(magentaRGB)))
   CASE (5)
       CALL RGBForeColor(VAL(I2COLOR(yellowRGB)))
   CASE (6)
       CALL RGBForeColor(VAL(I2COLOR(cyanRGB)))
   CASE (7)
       CALL RGBForeColor(VAL(I2COLOR(orangeRGB)))
   CASE (8)
       CALL RGBForeColor(VAL(I2COLOR(ltblueRGB)))
   CASE (9)
       CALL RGBForeColor(VAL(I2COLOR(purpleRGB)))
   CASE (10)
       CALL RGBForeColor(VAL(I2COLOR(brownRGB)))
   CASE (11)
       CALL RGBForeColor(VAL(I2COLOR(dkgreenRGB)))
   CASE DEFAULT
       CALL RGBForeColor(VAL(I2COLOR(blackRGB)))
   END SELECT
   DO K = 1,NT
    IX = IAXIS(2)+NINT((K-1)*SCX)
    IY = IAXIS(1)+NINT((YMAX-Y(J,K))*SCY)
    IY = ISIGN(IY,MIN(32767,ABS(IY)))
    IF (K==1) THEN
        CALL MoveTo(VAL2(IX),VAL2(IY))
    ELSE IF (Y(J,K-1)==0.0 .OR. Y(J,K)==0.0) THEN
        CALL MoveTo(VAL2(IX),VAL2(IY))
    ELSE
        CALL LineTo(VAL2(IX),VAL2(IY))
    END IF
   END DO
END DO
RETURN
END


SUBROUTINE DRWSCL(DA,NT,KGR)
INTEGER*4 NT,KGR
IMPLICIT NONE
CHARACTER*9 DA(NT)
RECORD /NLPGPH/ GR
POINTER (p_GR,GR)
RECORD /Str255/ DStr,DAStr
INTEGER*2 IX,IY
```

```
      INTEGER*4 K,I2COLOR
      REAL*8 Y

      p_GR = KGR

      GR.h_axPict = OpenPicture(GR.gr_rect)
      CALL TextFont(VAL2(4))
      CALL TextSize(VAL2(9))
      CALL RGBForeColor(VAL(I2COLOR(whiteRGB)))

C     X-axis for water levels
      CALL MoveTo(VAL2(GR.rect(1).left),VAL2(GR.rect(1).bottom))
      CALL LineTo(VAL2(GR.rect(1).right),VAL2(GR.rect(1).bottom))
      DO K = 1,NT,12
          IX = GR.rect(1).left+NINT((K-1)*GR.SCX)
          CALL MoveTo(VAL2(IX),VAL2(GR.rect(1).bottom))
          CALL Line(VAL2(0),VAL2(3))                !Tick marks
          CALL RGBForeColor(VAL(I2COLOR(ltgrayRGB)))
          CALL MoveTo(VAL2(IX),VAL2(GR.rect(1).bottom))
          CALL LineTo(VAL2(IX),VAL2(GR.rect(1).top))  !Grid lines
          CALL RGBForeColor(VAL(I2COLOR(whiteRGB)))
      END DO
C     Y-axis for water levels
      CALL MoveTo(VAL2(GR.rect(1).left),VAL2(GR.rect(1).bottom))
      CALL LineTo(VAL2(GR.rect(1).left),VAL2(GR.rect(1).top))
      DO Y = GR.WLMIN,GR.WLMAX,(GR.WLMAX-GR.WLMIN)/5.
          IY = GR.rect(1).top+NINT((GR.WLMAX-Y)*GR.SCY1)
          CALL MoveTo(VAL2(GR.rect(1).left),VAL2(IY))
          CALL Line(VAL2(-3),VAL2(0))               !Tick marks
          CALL RGBForeColor(VAL(I2COLOR(ltgrayRGB)))
          CALL MoveTo(VAL2(GR.rect(1).left),VAL2(IY))
          CALL LineTo(VAL2(GR.rect(1).right),VAL2(IY))   !Grid lines
          CALL RGBForeColor(VAL(I2COLOR(whiteRGB)))
          CALL Num2Str(VAL(DForm0),VAL(Y),DStr)
          CALL MoveTo(VAL2(GR.rect(1).left-StringWidth(DStr)-6),VAL2(IY+4))
          CALL DrawString(DStr)                     !Label
      END DO

C     X-axis for flows
      CALL MoveTo(VAL2(GR.rect(2).left),VAL2(GR.rect(2).bottom))
      CALL LineTo(VAL2(GR.rect(2).right),VAL2(GR.rect(2).bottom))
      DO K = 1,NT,12
          IX = GR.rect(2).left+NINT((K-1)*GR.SCX)
          CALL MoveTo(VAL2(IX),VAL2(GR.rect(2).bottom))
          CALL Line(VAL2(0),VAL2(3))                !Tick marks
          DAStr.chars = trim(DA(K))
          DAStr.len = len(trim(DA(K)))
          CALL MoveTo(VAL2(IX-StringWidth(DAStr)/2),VAL2(GR.rect(2).bottom+15))
          CALL DrawString(DAStr)                    !Label
          CALL RGBForeColor(VAL(I2COLOR(ltgrayRGB)))
          CALL MoveTo(VAL2(IX),VAL2(GR.rect(2).bottom))
          CALL LineTo(VAL2(IX),VAL2(GR.rect(2).top))  !Grid lines
          CALL RGBForeColor(VAL(I2COLOR(whiteRGB)))
      END DO
C     Y-axis for flows
      CALL MoveTo(VAL2(GR.rect(2).left),VAL2(GR.rect(2).bottom))
      CALL LineTo(VAL2(GR.rect(2).left),VAL2(GR.rect(2).top))
      DO Y = GR.REMIN,GR.REMAX,(GR.REMAX-GR.REMIN)/5.
          IY = GR.rect(2).top+NINT((GR.REMAX-Y)*GR.SCY2)
          CALL MoveTo(VAL2(GR.rect(2).left),VAL2(IY))
          CALL Line(VAL2(-3),VAL2(0))               !Tick marks
          CALL RGBForeColor(VAL(I2COLOR(ltgrayRGB)))
          CALL MoveTo(VAL2(GR.rect(2).left),VAL2(IY))
          CALL LineTo(VAL2(GR.rect(2).right),VAL2(IY))   !Grid lines
```

```
    CALL RGBForeColor(VAL(I2COLOR(whiteRGB)))
    CALL Num2Str(VAL(DForm0),VAL(Y),DStr)
    CALL MoveTo(VAL2(GR.rect(2).left-StringWidth(DStr)-6),VAL2(IY+4))
    CALL DrawString(DStr)                          !Label
END DO

CALL ClosePicture()

RETURN
END


INTEGER*4 FUNCTION I2COLOR(I)
IMPLICIT NONE
INTEGER*4 I
RECORD /ColorTable/ CTAB
RECORD /ColorSpec/ ctTAB
POINTER (p_CTAB,CTAB)
POINTER (h_CTAB,p_CTAB)
POINTER (p_ctTAB,ctTAB)

h_CTAB = RGetResource(VAL(Z'636C7574'),VAL2(8))    !System color table
p_ctTAB = LOC(CTAB.ctTable) + I*SIZEOF(/ColorSpec/)
I2COLOR = LOC(ctTAB.rgb)
RETURN
END


INTEGER*4 FUNCTION IMASK(J)
IMPLICIT NONE
INTEGER*4 J

IF (J .LE. 1) THEN    !avoid sign bit
    IMASK = Z'80000000'
ELSE IF (J .GT. 32) THEN !
    IMASK = Z'00000000'
ELSE
    IMASK = 2**(32-J)
END IF

RETURN
END
```