DISSERTATION

EVALUATING LEAST ABSOLUTE DEVIATION REGRESSION AS AN INVERSE MODEL IN GROUNDWATER FLOW CALIBRATION

Submitted by

John Matthew Huddleston

Department of Geosciences

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2004

COLORADO STATE UNIVERSITY

March 24, 2004

WE HEREBY RECOMMEND THAT THE DISSERTATION PREPARED UNDER OUR SUPERVISION BY JOHN MATTHEW HUDDLESTON ENTITLED EVALUATING LEAST ABSOLUTE DEVIATION REGESSION AS AN INVERSE MODEL IN GROUNDWATER FLOW CALIBRATION BE ACCEPTED AS FULFILLING IN PART REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY.

Committee on Graduate Work

Advisor

Department Head

ABSTRACT OF DISSERTATION

EVALUATING LEAST ABSOLUTE DEVIATION REGRESSION AS AN INVERSE MODEL IN GROUNDWATER FLOW CALIBRATION

The automated inverse procedure built into the US Geological Survey MODFLOW groundwater model works well with single layer geologic systems; however, it does not work well with multi-layer systems. This dissertation provides the Least sum Absolute Deviation regression and Expectation Maximization procedures. These new FORTRAN procedures were added to the parameter estimation source code of the MODFLOW groundwater computer model. The resulting inverse MODFLOW model can now calculate the hydraulic conductivity for multi-layer systems. The model was applied to Walnut Creek watershed in Iowa. Hydraulic conductivity parameters computed with the inverse watershed model using the Least sum Absolute Deviation regression and the Expectation Maximization procedures agree with published data. A forward groundwater model of the larger encompassing Upper Skunk River basin in Iowa was created. The hydraulic conductivity parameters computed by both Walnut Creek inverse models were used to create two forward basin models. Using the hydraulic conductivity from the Least sum Absolute Deviation regression and Expectation Maximization methods on the Upper Skunk river basin produced model heads for Walnut Creek that were within two meters of observed heads. The MODFLOW model predicted a confined aquifer system while the Least sum Absolute Deviation regression and Expectation Maximization model predicted an unconfined system. Literature documents Walnut Creek an an unconfined system. This modified MODFLOW model is a better tool for modeling multi-layer groundwater aquifers.

John Matthew Huddleston
Department of Geosciences
Colorado State University
Fort Collins, CO 80523
Summer, 2004

This author is grateful

to

the U. S. Geological Survey

and to

the USDA National Soil Tilth Laboratory

for detailed groundwater and precipitation data on Walnut Creek

to

the State of Iowa Department of Natural Resources

for Geographic Information System data and images

and

with special thanks

to

Freeman Minson Smith

and

my wife Paula

who have been supportive through this time.

TABLE OF CONTENTS

CHAPTER 1
GENERAL INTRODUCTION

CHAPTER 2
STATISTICS

CHAPTER 3
GROUNDWATER

CHAPTER 4
WATERSHED AND BASIN DESCRIPTIONS

CHAPTER 5
INVERSE GROUNDWATER MODELING OF WALNUT CREEK

CHAPTER 9

CHAPTER 10
APPENDICES

CHAPTER 1

GENERAL INTRODUCTION

1.0 INTRODUCTION

Groundwater inverse modeling determines the parameters to represent a given field in a modeler designed situation. There is a recent innovation in the US Geological Survey MODFLOW model that will allow a modeler to determine the forward modeling parameters for a groundwater model in an automated computational procedure that is set up correctly (Harbaugh et al., 2000). Correctly means correct geology, hydrography, hypsography, and computer model parameters. The computer optimal function is based on least squares Gauss-Newton-Levenberg-Marquardt procedure. In this dissertation, the computer procedures were changed. The regression procedure was replaced with a Least sum Absolute Deviation procedure and an intelligent procedure based on Expectation Maximization will compute the new forward parameters.

The automated procedure computes the desired parameters by either one of two means, minimizing the forward parameter or minimizing the objective function. The best procedure is for the iterations to stop the computer model when the forward parameters have converged to a desired tolerance. A parameter tolerance is a small difference between the current and the previously estimated parameter. When the last difference of the forward parameter is smaller than the modeler given tolerance, the computer procedure stops. When the last difference from the desired forward parameter is greater than the tolerance, the next step is to observe the objective function differences. The objective function is established to minimize the differences over the entire area of concern. While observation of the objective function may help to solve

global differences, it may not help the solution as a whole; in fact, it will force the modeler to re-examine the whole model.

This dissertation provides for a new MODFLOW Least sum Absolute Deviation (LAD) regression methodology for groundwater inverse modeling. Convergence to the Least sum Absolute Deviation solution optimizes the forward parameters using an Expectation Maximization procedure.

1.1 OVERVIEW

In 1856, Henri Darcy observed that the volume of flow through a porous medium when flow is laminar is directly proportional to the head loss and inversely proportional to the length of the flow path. Although empirical, this is now known as Darcy's law. The coefficient of proportionality is K, the saturated hydraulic conductivity. To estimate field hydraulic conductivities the calculated or measured porosity, velocity, and hydraulic gradient are required. Porosity is measured in place whereas both velocity and gradient measurements require at least two observation locations. The problem of determining the parameters to be used to represent a given field situation in a selected subsurface flow or transport model is referred to as the inverse problem.

1.2 DEFINITION OF THE PROBLEM

Numerical groundwater flow models require the storage coefficient (storativity), specific storage, and hydraulic conductivity parameters to characterize the aquifer. Usually prior knowledge of the values of the parameters is limited and they must be quantified by the calibration, which uses observations of hydraulic heads.

Trial and error approaches estimate the parameters by experienced modelers. Because the number and combinations of parameter adjustments are not bounded, "trial and error" is time consuming. In addition, the combination of parameters to fit a solution is nearly infinite. In the end, the modeler recognizes that the model fits the observed conditions but the input parameters may be one of many sets in their model to generate output that fits the observed data. The trial and error approach uses multiple executions of the forward model. The process is an inverse procedure, albeit carried out manually based on the skill of the practitioner.

Automated inverse models identify unknown model parameters by using mathematical optimization techniques. The most important one is minimizing the model forward parameter. The second important optimization of the inverse model is to minimize the discrepancies between the observed values and the model calculated values. This is carried out over the area of concern based on a geographically defined domain within which a predefined objective function operates. In an inverse groundwater flow model the objective function analyzes the difference in heads.

Inverse modeling is a two step process as the inverse model is run to calculate the parameters and then a forward model is run with the new parameters, the differences in the heads are calculated, and the cycle continues until either the forward parameter reaches a pre-defined minimum tolerance or the objective function criterion is met. When the forward parameter cannot be minimized and the objective function does not converge to a minimum tolerance, the process can be terminated after a user specified number of cycles.

1.3 OBJECTIVE

In this dissertation, there are three objectives. First, a groundwater model of approximately 5000 hectares (approximately 20 square miles) Walnut Creek watershed in Iowa will be created. The USGS MODFLOW model least squares inverse procedure will be used to create the inverse model.

The second objective is to revise the USGS MODFLOW groundwater model to replace the current least squares regression procedure with a Least sum Absolute Deviation regression procedure and a forward predicting Expectation Maximization procedure. The Expectation Maximization procedure is a mathematical solution based on matrix procedures to be defined in this dissertation.

The third objective is to integrate the results of Objectives 1 and 2 with geologic and hydrologic data into a forward model of the larger Upper Skunk River basin in Iowa to evaluate the groundwater within the Walnut Creek watershed. The problems of applying results of parameter determination from a small area to a larger encompassing area will be discussed. The Skunk River basin is approximately 156,000 hectares (approximately 600 square miles). Both the Walnut Creek watershed and the Upper Skunk River basin are to be modeled using geographic information system (GIS) data. The issues related to GIS modeling will be solved using ESRI, ET GeoWizards, and compiled C software.

1.4 HISTORICAL PERSPECTIVE OF REGRESSION

Least squares methods date back to 1805 with the publication of Legendre's work on the subject. A half century earlier though, Boscovich (Sheynin, 1973)

examined the ellipticity of the earth. Newton (Sheynin, 1973) and others had suggested that the earth's rotation could be expected to make it bulge at the equator with a corresponding flattening at the poles. Boscovich suggested an estimate by minimizing the sum of absolute errors subject to the constraint that the fitted lines pass through the centroid and the errors sum to zero. For each point designated by an x,y pair, the slope b_i is calculated as

$$b_i = \frac{y_i - \bar{y}}{x_i - \bar{x}} \quad \text{.....1.4.1}$$

Each slope is associated with a weight $w_i = |x_i - \bar{x}|$. Find the smallest j such that

$$\sum_{i=1}^{j} w_i = \frac{1}{2}\sum_{i=1}^{n} w_i \quad \text{.....1.4.2}$$

The Boscovich estimator was studied in detail by Laplace in 1789 and later in his monumental work Traite de Mechanique Celeste. Boscovich's proposal, which Laplace later called the "method of situation" is a blend of mean and median ideas; in effect, the slope parameter b is estimated as the median and the intercept is estimated as a mean. Edgeworth revived these ideas in 1888 in his early work on index numbers and weighted averages. The median, he argued, could easily be superior to the mean; he discarded the Boscovich-Laplace constraint that the residuals sum to zero and proposed instead to minimize the sum of absolute residuals in both intercept and slope parameters, calling it a "double median" method and noted that it could be extended to a "plural median". A geometric algorithm was given for the bivariate case; however, it was not until the advent of linear programming that a simple and computationally efficient procedure would solve this problem. (Sheynin, 1973)

1.5 DATA AVAILABILITY FOR GROUNDWATER MODELING

Data preparation for groundwater modeling can be secured from one or more sources: 1) transmissivity measurements; 2) hydraulic head measurements; 3) tracer concentrations in wells from tracer tests; and 4) geologic information on the nature and characteristics of the formation.

Large scale conductivity trends which extend over thousands of meters tend to be governed by regional geological processes which act over long time periods. Smaller scale fluctuations about these trends are likely to be the result of more transient processes which are associated with shorter geologic time processes such as deposition from small streams which flow during unusually wet streams.

Regional groundwater models, such as the modeling of the Upper Skunk River basin in Iowa, are characterized by large dimensionality and a large number of parameters: spatially distributed hydraulic conductivity, variable bedrock topology, storage coefficients, transmissivities, and dispersion coefficients. In practice the required distribution of system parameters is difficult to obtain. Direct measurements of parameters through in-situ measurement or laboratory tests of samples is generally not economical, and often not meaningful, since the samples tend to reflect local conditions due to variations at a scale much smaller than the scale of discretization of the model. An indirect method for inferring the system parameters is through the study of a system response to given excitations such as pumping tests. This customary way of estimating parameters depends on several simplifications about the system and gives a lumped measure of the system response at the location of the test.

The uncertainty associated with prediction of piezometric head in an aquifer is due to multiple factors: model structure of equations and spatial and temporal discretization; parameter uncertainty in space; boundary conditions; source and sink terms; the amount and location of well pumpage; initial conditions uncertainty; and measurement error.

This dissertation will apply the groundwater parameters calculated from an inverse model of a 5000 hectare watershed assuming two layers to a 156,000 hectare basin assuming similar geologic layers. The depth to bedrock is acquired using Iowa Department of Natural Resources geographic information system data hosted on the ftp.igsb.uiowa.edu website.

1.6 LAYOUT OF THIS DISSERTATION

Chapter 2 includes a description of statistics, regression, and Expectation Maximization. Chapter 3 includes the derivation of the groundwater equation. Chapter 4 lays out the geology, watershed and basin characteristics of Walnut Creek watershed and the Upper Skunk River basin in Iowa. Chapter 5 describes the inverse groundwater modeling of Walnut Creek using the MODFLOW model. Chapter 6 describes the inverse modeling of Walnut Creek using the modified MODFLOW model with Least sum Absolute Deviation and Expectation Maximization procedures. Chapter 7 contains the Upper Skunk River basin modeling. Chapter 8 contains the analysis, results, summary, and conclusions of the research. Chapter 9 contains the references for this study.

Chapter 10 contains the appendices. Section 1 includes well location and elevation data. Section 2 contains the list of head observations at the wells. Section 3 contains a listing of the parameter estimation, least squares Gauss-Newton, Least sum Absolute Deviation, and Expectation Maximization FORTRAN source code. Section 4 contains a list of the observed and model heads for three different top depths of five, six, and seven meters. Section 5 is a contains a list of the observed and model head differences for three different top depths of five, six, and seven meters as well as a plot of these differences. Section 6 is a listing of the Exp2Shp C source code to convert ArgusONE export files into shapefiles. Section 7 is a listing of the C source code to intersect river hydrography with contour hypsography to produce prescribed head data. Section 8 is a description of the Iowa Department of Natural Resources Geographic Information System data.

CHAPTER 2
STATISTICS

2.0 INTRODUCTION

2.1 STATISTICAL ANALYSIS

2.1.1 Parametric statistics

Uncertainty in the subsurface distribution of hydrogeologic properties can be broken into two components: geologic uncertainty and parameter uncertainty. Geologic uncertainty refers to the uncertainty in the location of the aquifer units and aquitards, as well as the uncertainty in the boundary conditions.

Parameter uncertainty denotes the uncertainty in the values of the hydrogeologic parameters throughout the spatial domain. Inferring properties about parameters fall into two categories: parametric and nonparametric. In parametric statistics, the mean is the statistic used to indicate average value of a population or sample. If the mean is combined with the standard deviation, then the pair of numbers indicates both the central tendency of the group of numbers and the spread. A large standard deviation reflects a large spread in the data; that is, the numbers are diverse and far apart. A small standard deviation reflects a tightness of the data. A plot of the data in a histogram creates a graph that looks like the well-known bell-shaped curve if it is normally distributed.

If the data fit a normal distribution the mean and the standard deviation can be combined to provide confidence intervals on the population. This information is often

used to create a range of values in which you might expect future sampled data to appear. More precisely,

$$\bar{X} - z_{\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right) \leq \mu \leq \bar{X} \leq \bar{X} + z_{\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right) \quad \text{.....2.1.1}$$

in which $\bar{X}$ is the sample mean, $z_{\alpha/2}$ are values of random variables having a standard normal distribution and cutting off $(1-\gamma)$ and $(1-\gamma/2)$ percent in the tail of the distribution; α is the level of significance and equals $(1-\gamma)$; and σ is the standard deviation of the population. This confidence interval represents a means of providing a range of values in which the true value can be expected to lie.

To employ parametric statistical tests, the data must be on an interval scale, continuous, and normally distributed.

2.1.2 Nonparametric Statistics

Whether using conditional simulation or stochastic simulation, model developers have the choice of a parametric or non-parametric approach. A parametric approach has model parameters determine the distribution function. Nonparametric statistics are often called distribution free; that is, there is no assumption about the underlying shape of the data. A nonparametric approach is a collection of generalized linear regression techniques for minimizing an estimation variance defined from a prior model for a covariance.

Nonparametric analysis includes such techniques as simple kriging, ordinary kriging, and indicatory kriging, as well as a variety of simulation techniques and

annealing with observed data. Nonparametric analysis will not be considered in this research study; it is mentioned here to differentiate between analyses.

2.1.3 Parametric Statistical Parameters

A parametric statistical sample contains single valued estimates such as the mean, variance, correlation coefficient, or a regression coefficient. These single value estimates represent the best estimate of the population values. It sometimes happens that one measurement in a series of measurements appears to disagree significantly with all the others. A legitimate measurement may deviate significantly from other measurements of the same quantity and may be rejected. Data rejection is controversial and, although based on some criterion, it is ultimately a subjective decision.

The hypothesis test gives an answer to whether or not an estimate meets a particular standard or criterion. If the sample size is less than 20, the t-test is appropriate, otherwise an analysis of variance is used. If the data has been grouped into 1-2 groups, the t-test is appropriate, otherwise an analysis of variance is used. If there is one dependent variable, the t-test is appropriate, otherwise an analysis of variance is used.

Confidence intervals reflect the precision of the sample estimate and additionally provide a probability statement about the likeliness of correctness of the data.

Correlation analysis provides a means of drawing inferences about the strength of the relationship between two or more variables. The correlation coefficient is used

as a measure of the goodness of fit between the prediction equation and the data sample. Relationships can be linear or nonlinear. The Pearson product moment correlation test is universally accepted for parametric data.

2.1.4 Mean's Connection with Optimization

Given a sample of single valued data a function and its first derivative

$$f(x) = \sum_{i=1}^{m} (x - b_i)^2 \quad \text{.....2.1.2}$$

$$f'(x) = \sum_{i=1}^{m} 2(x - b_i) = 2\sum_{i=1}^{m} x - 2\sum_{i=1}^{m} b_i \quad \text{.....2.1.3}$$

set the first derivative to 0 to find its minimum value and divide both sides by $\frac{1}{2m}$

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x = \frac{1}{m}\sum_{i=1}^{m} b_i \quad \text{.....2.1.4}$$

and in the limit that $m \to \infty$, $\bar{x}$ is a minimum.

2.1.5 Median's Connection with Optimization

Given a sample s of single valued data a function and its first derivative

$$f(x) = \sum_{i=1}^{m} |x - b_i| \quad \text{.....2.1.5}$$

$$f'(x) = \sum_{i=1}^{m} \operatorname{sgn}(x - b_i) \text{ where } \operatorname{sgn}(x) = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \quad \text{.....2.1.6}$$

equals (the number of b_i's smaller than x) – (the number of b_i's larger than x).

2.1.6 Statistical Optimization

Statistical optimization differs somewhat from the theorems provided for finding the minima and maxima of an explicit function. For the explicit function, there are only two elements, the function and the unknown(s). For statistical optimization, the function to be optimized is called the objective function, which is an explicit mathematical function that describes what is considered to be the optimal solution. Second, there is a mathematical model that relates a random variable, called the criterion or dependent variable, to a vector of unknowns and a vector of predictor variables, called independent variables. The predictor variables usually have a causal relationship with the criterion variable. The third element of statistical optimization is the data set. The data set consists of measured values of the criterion variable and the predictor variable(s).

2.1.7 Methods of Optimization

There are three general categories that can be used to classify optimization techniques: analytical, numerical, and subjective. Analytical optimization uses analytical calculus in deriving the unknowns from the objective function. Analytic solutions provide a direct and exact solution for simple model structures. However, they do not necessarily guarantee optimality since the vanishing differential may be a local minimum, maximum, valley, ridge, or saddle point.

Numerical optimization evaluates coefficients using computational techniques such as regression. Numerical optimization techniques include matrix and linear

programming approaches. In problems that can be solved using a matrix approach to least absolute deviation or least squares, there is a single value that minimizes the sums.

Subjective optimization is a trial and error process that relies heavily on the user's experience.

2.1.8 Matrix Approach to Least Absolute Deviation Regression

The error, or residual, is defined here as the difference between the predicted and measured value of the criterion value. As such, a positive residual indicates over prediction. Using a Manhattan distance model, $\|x\|_1 = \sum_i |x_i|$ and given a system Ax = b, the least absolute deviation regression estimate that minimizes the error $\hat{x} = \text{argmin}_x \|b - Ax\|_1$ is shown as follows:

$$f(x) = \|b - Ax\| = \sum_i \left| b_i - \sum_j a_{ij} x_j \right| \quad \text{.....2.1.7}$$

differentiate f(x) and set it to 0 to find the minimum,

$$\frac{\partial f}{\partial x_k}(\hat{x}) = \sum_{i=1}^{m} \frac{b_i - \sum_j a_{ij}\hat{x}_j}{\left| b_i - \sum_j a_{ij}\hat{x}_j \right|}(-a_{ik}) = 0, \quad k=1,2,...,n \quad \text{.....2.1.8}$$

rearranging

$$\sum_i \frac{a_{ik} b_i}{e_i(\hat{x})} = \sum_i \sum_j \frac{a_{ik} a_{ij} \hat{x}_j}{e(\hat{x})}, \quad k=1,2,...,n \quad \text{.....2.1.9}$$

or in matrix notation

$$A^T E(\hat{x})b = A^T E(\hat{x})A\hat{x}, \text{ where } E(\hat{x}) = \text{Diag}(e(\hat{x}))^{-1} \quad \text{.....2.1.10}$$

then assuming $A^T E(\hat{x})A$ is invertible

$$\hat{x} = \left(A^T E(\hat{x})A\right)^{-1} A^T E(\hat{x})b \quad \text{.....2.1.11}$$

This is an implicit equation that can be solved iteratively,

$$\hat{x}^0 = 0 \quad \text{.....2.1.12a}$$

$$\hat{x}^1 = \left(A^T E(\hat{x}^0)A\right)^{-1} A^T E(\hat{x}^0)b \quad \text{.....2.1.12b}$$

$$\hat{x}^2 = \left(A^T E(\hat{x}^1)A\right)^{-1} A^T E(\hat{x}^1)b \quad \text{.....2.1.12c}$$

$$\hat{x}^{k+1} = \left(A^T E(\hat{x}^k)A\right)^{-1} A^T E(\hat{x}^k)b \quad \text{.....2.1.12d}$$

2.1.9 Matrix Approach to Least Squares Regression

Using an Euclidean distance model, $\|x\|_2 = \left(\sum_i x_i^2\right)^{\frac{1}{2}}$ and given the system

$Ax = b$, the least squares regression estimate that minimizes the error

$\bar{x} = \text{argmin}_x \|b - Ax\|_2^2$ is shown as follows:

$$f(x) = \|b - Ax\|_2^2 = \sum_i \left(b_i - \sum_j a_{ij}x_j\right)^2 \quad \text{.....2.1.13}$$

differentiate f(x) and set it to 0 to solve for the minimum

$$\frac{\partial f}{\partial x}(\bar{x}) = \sum_i 2\left(b_i - \sum_j a_{ij}\bar{x}_j\right)\left(-a_{ik}\right) = 0, \quad k=1,2,...,n \quad \text{.....2.1.14}$$

rearranging

$$\sum_i a_{ik} b_i = \sum_i \sum_j a_{ik} a_{ij} \bar{x}_j, \quad k = 1,2,...,n \quad \text{.....2.1.15}$$

in matrix notation

$$A^T b = A^T A \bar{x} \quad \text{.....2.1.16}$$

assuming $A^T A$ is invertible

$$\bar{x} = \left(A^T A\right)^{-1} A^T b \quad \text{.....2.1.17}$$

2.1.10 Criteria for Evaluation of Regression Models

The following criteria can be assessed in evaluating a regression model: 1) the correlation coefficient; 2) the standard error of estimate; 3) the F statistic for the analysis of variance; 4) the rationality of the coefficients and the relative importance of the predictor variable; and 5) the degree to which the underlying assumptions of the regression model are met. The correlation coefficient is an index of the degree of association between the measured and predicted variables.

2.2 LEAST ABSOLUTE DEVIATION LITERATURE REVIEW

2.2.1 Parameter Evaluation

In parametric regression models, the least squares estimator is usually used for parameter estimates. If the error term is distributed as normal, then the least squares estimator is a maximum likelihood estimator and attains minimum variance within unbiased estimators. One or more outliers may cause a large error in the least squares estimator. This occurs in the case of fatter tail distributions (skewed) of the error term.

In such cases, more robust estimators, such as the least absolute deviation estimator, are desirable. Outliers and skew were examined in the papers by: Cade and Richards (1996); Foss, Myrtveit, and Strensrud (2001); Tasker and Granato (2000); Huber (1973); and Huddleston and Shafer (1987).

Cade and Richards (1996) examined permutation testing that is sampling without replacement. They found that least squared regression estimates provided a gain in power when error distributions are not approximated well by the normal distribution or when a small number of outliers unduly influence least squares regression. They further found that least absolute deviation estimates are median unbiased and are maximum likelihood estimates when the errors are independent identically distributed.

Foss, Myrtviet, and Strensrud (2001) initially proposed that the least absolute deviation estimator would be superior to least squares estimator in software engineering applications. Their results were not conclusive.

Tasker and Granato (2000) classify and analyze water quantity and quality data for highway and urban runoff and state that robust regression methods, such as least absolute deviation, are insensitive to the effects of outliers in data and are useful for detecting outliers by accentuating observations with large residuals. Tasker and Granato state the following: 1) the least squares estimator is sensitive to outliers because it gives relatively greater weight to large residuals; 2) the least absolute deviation estimator is resistant to outliers but may give too much weight to small residuals; 3) the use of least absolute deviation is more efficient with respect to the presence of outliers and skewed distributions.

Huber (1973) proposed a compromise estimator that is robust yet relatively efficient if data are normal. It weights small residuals as least squares estimator and large residuals as a least absolute deviation estimator.

Huddleston and Shafer (1987) examined snow survey data used in prediction of streamflow runoff and determined that when skew and kurtosis were used to determine the probability model, forecast errors could be reduced from 25 to 5 percent.

Mielke and Berry (2001) devote a chapter of their book on permutation methods to least squares and least absolute deviation regressions. They state that fat-tailed distributions involve an abundance of extreme values and least squares gives disproportionate weight to such values whereas least absolute deviation regression is virtually unaffected by the presence of a few extreme values.

2.2.2 Confidence Intervals

Tasker and Granato (2000) state that one problem with robust methods is an apparent lack of agreement among authorities on how best to construct confidence intervals for the parameter estimates. Cade and Richards (1996) reiterate an assertion by Rao (1988) that greater use of least absolute deviation regression continues to be limited by a lack of acceptable procedures for testing hypotheses about alternative regression models.

2.2.3 Application to Linear Programming

Dougherty and Smith (1966) proposed the use of a simple trend surface method in which the sum of absolute values of deviations is minimized. Because of the

difficulty of handling absolute values with traditional mathematical tools, they proposed that a linear programming approach be used. The suggested method had many potential advantages and had some theoretical basis for the geophysical data with which they were concerned.

2.3 LEAST SQUARES USE IN GROUNDWATER MODELING

2.3.1 Groundwater Parameters

The spatially variable parameters of most interest in groundwater inverse applications include hydraulic conductivity (K) for unconfined aquifers or transmissivity for confined aquifers (transmissivity T=KD, where D is the thickness of the confined aquifer), storage coefficients (S), groundwater recharge and discharge, fluxes and piezometric heads on designated boundaries, and chemical rate coefficients. All of these parameters are uncertain, but some may have a greater effect on predictions than others in any given application. Usually it is not advisable to include recharge as a fitting parameter but rather to estimate recharge based on precipitation, evapotranspiration, infiltration, and surface runoff calculations.

2.3.2 Least Squares Objective Function

The parameter values are elements in a parameter vector defined as:

$$b = \begin{matrix} b_1 \\ b_2 \\ \vdots \\ b_{N_p} \end{matrix} = \begin{bmatrix} b_1 & b_2 & \cdots & b_{N_p} \end{bmatrix}^T \quad \text{.....2.3.1}$$

where the superscript T means the transpose of the vector or matrix. The parameter vector can include unknown values such as hydraulic conductivity and transmissivity. Assume that we also have a set of field operations (values) to be used for model calibration. If the number of observations is ND, the observation vector can be shown as:

$$Y = \left[Y_1 \quad Y_2 \quad \cdots \quad Y_{ND} \right]^T \qquad 2.3.2$$

The observations can include hydraulic head data, flux data, or prior information about the value of one or more of the parameters. Corresponding to each observed value is a value computed with the model. This model value is symbolized by $Y_i(b)$ for the i^{th} observed value. If Y_i is the i^{th} observed value such as hydraulic head at a certain time and location, then $Y_i(b)$ is the model computed value at that time and location. Just as the set of observed values are elements in the observation vector Y, the set of corresponding model computed values are elements in the model function vector defined as:

$$Y(b) = \left[Y_1(b) \quad Y_2(b) \quad \cdots \quad Y_{ND}(b) \right]^T \qquad 2.3.3$$

Y_p is a prior estimate of the i^{th} model parameter value, and $Y_p(b)$ is equal to the actual value of b_l in the model (i.e. $Y_p(b) = b_l$). An example is a prior estimate of a recharge value available from root zone water balance considerations. Another example is a prior estimate of a zonal transmissivity available from pumping tests. The set of prior estimates are shown as:

$$Y_p(b) = \left[Y_{p_1}(b) \quad Y_{p_2}(b) \quad \cdots \quad Y_{p_{NPR}}(b) \right]^T \qquad 2.3.4$$

In order to develop an inverse procedure, it is necessary to define a measure of the difference between the set of observations used for model calibration, Y, and the corresponding set of values computed by the model, Y(b). One measure that is often used is the sum of squared differences:

$$S(b) = \sum_{t=1}^{ND} \left[Y_t - Y_t(b) \right]^2 \quad \text{.....2.3.5}$$

The MODFLOW inverse procedure uses a weighted measure equation as follows:

$$S(b) = \sum_{t=1}^{ND} \omega_t \left[Y_t - Y_t(b) \right]^2 + \sum_{p=1}^{NPR} \omega_p \left[Y_p - Y_p(b) \right]^2 \quad \text{.....2.3.6}$$

where ND is the number of observations and NPR is the number of prior information values. Parameter estimation is built into MODFLOW using the theory and methods that were pioneered in MODFLOWP and UCODE (Hill, 1992).

Weighting requires a full weight matrix (Hill, 1998, p. 7, eq. 2), where the diagonals of the weight matrix equal the observation error variances and the off-diagonals equal the covariances. A diagonal weight matrix is strictly valid only if the measurement errors are independent. The weight matrix capabilities of MODFLOW are different for hydraulic heads and for the other types of observations and for prior information. For hydraulic head observations, MODFLOW does not support a full weight matrix, but it does support differencing methods designed to accommodate commonly encountered error correlation. For all other types of observations and prior information, MODFLOW supports a full weight matrix.

2.3.3 Gauss-Newton Method

Consider that $\bar{b}$ is the parameter vector $\left[b_1 \quad b_2 \quad \cdots \quad b_{N_p} \right]^T$ that needs to be estimated. Gauss-Newton optimization method is specifically designed for minimizing the objective function that has the form of sum of square functions. The derivation of the algorithm (Sun, 1994) is as follows:

$$f_m(\bar{b}) = S(b) = \sum_{t=1}^{ND} \omega_t \left[Y_t - Y_t(b) \right]^2 + \sum_{p=1}^{NPR} \omega_p \left[Y_p - Y_p(b) \right]^2 \quad \text{.....2.3.7}$$

The first order derivatives of $Y(\bar{b})$ can be obtained:

$$\frac{\partial Y}{\partial b_i} = 2 \sum_{m=1}^{ND} f_m \frac{\partial f_m}{\partial b_i} \quad (i = 1,..,N_p) \quad \text{.....2.3.8}$$

where N_p is the number of parameters. The second derivatives are

$$\frac{\partial^2 Y}{\partial b_i \, \partial b_j} = 2 \sum_{m=1}^{ND} \left[\frac{\partial f_m}{\partial b_i} \frac{\partial f_m}{\partial b_{ji}} + f_m \frac{\partial^2 f_m}{\partial b_i \, \partial p_j} \right] \quad (i = 1,..,N_p; j = 1,..,N_p) \quad \text{....2.3.9}$$

In the above equation $f_m(\bar{b})$ is a residual, when $\bar{b}$ is not too far from the optimum value, it can be assumed that the value of $f_m(\bar{b})$ is small and the second order terms of the right-hand side can be ignored, thus

$$\frac{\partial^2 Y}{\partial b_i \partial b_j} = 2 \sum_{m=1}^{ND} \frac{\partial f_m}{\partial b_i} \frac{\partial f_m}{\partial b_{ji}} \quad \text{.....2.3.10}$$

It is possible to define a new matrix

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1}{\partial b_1} & \frac{\partial f_1}{\partial b_2} & \cdots & \frac{\partial f_1}{\partial b_{N_p}} \\ \frac{\partial f_2}{\partial b_1} & \frac{\partial f_2}{\partial b_2} & \cdots & \frac{\partial f_2}{\partial b_{N_p}} \\ . & . & . & . \\ . & . & . & . \\ \frac{\partial f_{ND}}{\partial b_1} & \frac{\partial f_{ND}}{\partial b_2} & \cdots & \frac{\partial f_{ND}}{\partial b_N} \end{bmatrix} \quad \text{.....2.3.11}$$

which consists of derivatives of functions $f_1(\bar{b})$, $f_2(\bar{b})$, ..., $f_{Np}(\bar{b})$ with respect to the variation of each parameter components b_1, b_2,..., b_N. Usually ND is much larger than N_p, and thus **A** is not a square matrix. Using matrix **A** and considering $\frac{\partial Y}{\partial b_1}$, gradient ∇Y can now be represented in matrix form

$$\nabla Y = 2A^T f \quad \text{.....2.3.12}$$

where $\mathbf{f} = (f_1 \ f_2 \ \ldots \ f_{Np})^T$. Considering $\frac{\partial^2 Y}{\partial b_i \, \partial b_j}$ and matrix **A**, the Hessian matrix **G** can be approximated by

$$G \approx 2 A^T A \quad \text{.....2.3.13}$$

Now substituting the previous two equations into Newton's equation

$$b_{k+1} = b_k - G^{-1} g_k \quad \text{.....2.3.14}$$

which results in

$$b_{k+1} = b_k - \left(A_k^T A_k\right)^{-1} A_k^T f_k \quad \text{.....2.3.15}$$

where the subscript k indicates that $\mathbf{A}_k$ and $\mathbf{f}_k$ are evaluated at $\bar{p}_k$. This equation can also be written in the form

$$\left(A_k^T A_k\right)\Delta b_k = -A_k^T f_k \quad \text{.....2.3.16}$$

which is a linear system of equations where $A_k{}^T A_k$ is N by N square matrix and $A_k{}^T f_k$ is an N-dimensional vector and $\Delta b_k = b_{k+1} - b_k$ is the unknown vector, which can be solved by Gaussian elimination or decomposition methods.

2.3.4 Gauss-Newton-Levenberg-Marquardt Method

The Gauss-Newton method is different from the Newton's method in such a way that the Hessian matrix is calculated using only the first derivative (equation 2.3.10). In the generation of a Gauss-Newton sequence for inverse solution, several problems may be encountered. First, it is possible that the search sequence does not converge, i.e. $Y(b_{k+1}) > Y(b_k)$ for some k. Second, matrix $A_k{}^T A_k$ is near singular (elements very close to zero), and a solution cannot be obtained. Third, the displacement vector Δb_k is so large that parameter values are not in the admissible region any more.

In order to avoid the above mentioned difficulties, it is necessary to modify the Gauss-Newton algorithm. The Levenberg-Marquardt method is one way to guarantee that $E(b_{k+1}) < E(b_k)$. The method is called the Gauss-Newton-Levenberg-Marquardt (GNLM) method and the modification includes an additional term added to $A_k{}^T A_k$ to avoid the singularity:

$$\left(A_k^T A_k + \lambda_M I\right)\Delta b_k = -A_k^T f_k \quad \text{.....2.3.17}$$

where λ_M is a coefficient and I is the unit matrix. When $\lambda_M = 0$, the method reduces to Gauss-Newton and if λ_M tends to infinity, Δb_k turns into the so called steepest descent

direction and the size Δb_k tends to zero. Therefore, $Y(b_{k+1}) < Y(b_k)$ can always be expected by increasing the value of λ_M. The criteria for stopping the GNLM inverse method is when the square sum between computed and measured values is less than a tolerance parameter between 10^{-8} and 10^{-5} (Hill, 1998).

2.3.5 Quasi-Newton Updating of the Normal Equations

The USGS MODFLOW 2000 uses a more sophisticated procedure based on Dennis et al. (1981) for estimating the additional term. In the nomenclature of the USGS WR98-4005 Appendix B (Hill, 1998), substitute $A_k^T w A_k + R$ for $A_k^T w A_k$ where R is an estimate of the difference between $A_k^T w A_k$ and the Hessian matrix equation 2.3.10, i.e.

$$\lambda_M I = A_k^T w A - \frac{\partial^2 Y}{\partial b_i \partial b_j} \quad \text{.....2.3.18}$$

2.4 EXPECTATION MAXIMIZATION

2.4.1 An Expectation Maximization Algorithm for Linear Models

Given a linear regression model

$$y(i) = x(i)\,\beta + \sigma\left(\sqrt{2}v(i)\right)\varepsilon(i),\ i=1, ..., n \quad \text{.....2.4.1}$$

where $\sigma > 0$, x(i) and β are Kx1 vectors, the $\varepsilon(i)$ are unobserved independently, identically distributed standard normal random variables and v(i) are unobserved independently identically distributed positive random numbers.

The Expectation Maximization (EM) algorithm consists of iteratively executing two steps: the E (expectation) step and the M (maximization) step. The E step

constructs a proxy log-likelihood by calculating the conditional expectation of the "complete" data log-likelihood given the "incomplete" data while taking the current fit of the parameters to be the parameters of the conditional distribution. Upon defining the incomplete data to be y1, y2, …, yn while defining the complete data are (y1,v1), (y2,v2), …, (yn,vn), this step gives the proxy log-likelihood.

2.4.2 Matrix Approach to the Expectation Maximization Solution

The vector β represents the B vector in the MODFLOW program and contains the estimation parameters. In order to calculate β use the results matrix R, the Least sum Absolute Deviation regression coefficients X, and the weights W from the model to compute

$$\Delta = \left(X^T WX\right)^{-1} X^T WXR \quad \text{.....2.4.2}$$

The resulting update vector Δ is composed of a general weight and individual weights for each parameter. The estimation parameter vector β is updated as follows:

$$\beta(i+1) = \beta(i) + \Delta \quad \text{.....2.4.3}$$

When the sum of the changes in the forward parameters is less than a preset difference, or tolerance, the solution is said to converge based on the forward parameter.

CHAPTER 3

GROUNDWATER MODELING

3.0 INTRODUCTION

This chapter will present the equations that are used in the solution of the groundwater flow equation.

3.1 GROUNDWATER THEORY

3.1.1 Pressure Head

A well-known relationship from fluid mechanics is Bernoulli's equation (Todd, 1959) for steady, non-viscous, incompressible fluids

$$\frac{p}{\rho g} + Z + \frac{V_p^2}{2g} = \text{constant} = H \quad \text{.....3.1.1}$$

where V_p is pore velocity. In groundwater flow, Bernoulli's equation is modified to account for the loss of energy due to viscous resistance within the individual pores

$$\frac{p_A}{\rho g} + Z_A + \frac{V_{pA}^2}{2g} = \frac{p_B}{\rho g} + Z_B + \frac{V_{pB}^2}{2g} + \Delta H \quad \text{.....3.1.2}$$

where ΔH is the total head loss in energy per unit weight of fluid. In most groundwater problems, the velocity heads are so small that they may be neglected. The Bernoulli equation is shortened to

$$\frac{p_A}{\rho g} + Z_A = \frac{p_B}{\rho g} + Z_B + \Delta H \quad \text{.....3.1.3}$$

and the total head at any one point is a combination of pressure head plus elevation

$$H = \frac{p}{\rho g} + Z \text{ or } p = \rho g H + Z \rho g \quad \text{.....3.1.4}$$

3.1.2 Darcy Flow

In 1856, Henri Darcy, in France, observed that the volume of flow through a porous medium when flow is laminar is directly proportional to the head loss and inversely proportional to the length of the flow path. In the limit that $\Delta x \xrightarrow{\lim} 0$ the hydraulic gradient $-\frac{\Delta H}{\Delta x}$ can be expressed as $-\frac{dH}{dx}$. Darcy concluded that the flow rate Q was equal to

$$Q = -KA\frac{dH}{dx} = -KA\frac{d}{dx}\left[\frac{p}{\rho g} + Z\right] \quad \text{.....3.1.5}$$

where Q is the flow rate, A is the cross sectional area perpendicular to the flow, and K is the hydraulic conductivity. The hydraulic conductivity has been known in the past as the coefficient of permeability, the effective permeability, and seepage coefficient. K can be expressed in terms of an intrinsic permeability as:

$$K = \frac{k \rho g}{\mu} \quad \text{.....3.1.6}$$

where k is the intrinsic permeability, ρ is the density, g is gravity, and μ is the viscosity of the fluid. Darcy's velocity is given as:

$$q = \frac{Q}{A} = -K\frac{dH}{dx} = -\frac{k \rho g}{\mu}\frac{d}{dx}\left[\frac{p}{\rho g} + Z\right] \quad \text{.....3.1.7}$$

Darcy's velocity may be extended to three dimensions:

$$q_x = -K_x \frac{\partial H}{\partial x} \quad q_y = -K_y \frac{\partial H}{\partial y} \quad q_z = -K_z \frac{\partial H}{\partial z} \qquad \text{.....3.1.8}$$

3.1.3 Hydrostatic Pressure

Consider a porous medium, an aquifer, which is saturated with liquid under hydrostatic pressure. Aquifers under hydrostatic pressure exhibit both compressible and elastic characteristics. Any change in storage mass is related to the change in head by a term called specific storage S_s. The specific storage is the volume of water which a unit volume of the aquifer releases from storage because of expansion of water and compression of the aquifer under a unit decline in the average head. Next, consider mass conservation within a volume in space of $\Delta x, \Delta y, \Delta z$ dimensions:

$$\begin{aligned} &\rho\, q_x\big|_x \, \Delta y \Delta z \Delta t - \rho\, q_x\big|_{x+\Delta x} \, \Delta y \Delta z \Delta t \\ &+ \rho\, q_y\big|_y \, \Delta x \Delta z \Delta t - \rho\, q_y\big|_{y+\Delta y} \, \Delta x \Delta z \Delta t \\ &+ \rho\, q_z\big|_z \, \Delta x \Delta y \Delta t - \rho\, q_z\big|_{z+\Delta z} \, \Delta x \Delta y \Delta t = \\ &\quad \rho\, S_s \, \Delta x \Delta y \Delta z \left[H\big|_{t+\Delta t} - H\big|_t \right] \end{aligned} \qquad \text{.....3.1.9}$$

Divide through by $\Delta x \Delta y \Delta z \Delta t$ and taking the limits

$$\Delta x \xrightarrow{\lim} 0, \ \Delta y \xrightarrow{\lim} 0, \ \Delta z \xrightarrow{\lim} 0, \ \Delta t \xrightarrow{\lim} 0 \qquad \text{.....3.1.10}$$

results in:

$$-\frac{\partial(\rho\, q_x)}{\partial x} - \frac{\partial(\rho\, q_y)}{\partial y} - \frac{\partial(\rho\, q_z)}{\partial z} = \rho\, S_s \frac{\partial H}{\partial t} \qquad \text{.....3.1.11}$$

Using the chain rule to break up the density and velocity components results in:

$$q_x \frac{\partial \rho}{\partial x} + q_y \frac{\partial \rho}{\partial y} + q_z \frac{\partial \rho}{\partial z} + \rho \frac{\partial q_x}{\partial x} + \rho \frac{\partial q_y}{\partial y} + \rho \frac{\partial q_z}{\partial z} = -\rho S_s \frac{\partial H}{\partial t} \quad \text{....3.1.12}$$

3.1.4 Energy Equation

Using the definition from above $p = \rho g H + Z \rho g$ take the partial differential

$$\frac{\partial p}{\partial x} = \rho g \frac{\partial H}{\partial x} + \frac{p}{\rho} \frac{\partial \rho}{\partial x} \quad \text{.....3.1.13a}$$

$$\frac{\partial p}{\partial y} = \rho g \frac{\partial H}{\partial y} + \frac{p}{\rho} \frac{\partial \rho}{\partial y} \quad \text{.....3.1.13b}$$

$$\frac{\partial p}{\partial z} = \rho g \frac{\partial H}{\partial z} + \frac{p}{\rho} \frac{\partial \rho}{\partial z} \quad \text{.....3.1.13c}$$

Now defining compressibility β of water as the reciprocal of its bulk modulus of elasticity

$$\beta = \left(\frac{1}{\rho}\right)\left(\frac{\partial \rho}{\partial p}\right) \quad \text{.....3.1.14}$$

$$\left(\frac{\partial \rho}{\partial p}\right) = \rho \beta \quad \text{.....3.1.15}$$

also, by definition

$$d\rho = \frac{\partial \rho}{\partial p} dp \quad \text{.....3.1.16}$$

dividing through by dx

$$\frac{d\rho}{dx} = \frac{\partial \rho}{\partial p} \frac{dp}{dx} = \rho \beta \frac{dp}{dx} \quad \text{.....3.1.17a}$$

similarly for the y and z directions

$$\frac{d\rho}{dy} = \rho\beta\frac{dp}{dy} \quad \text{.....3.1.17b}$$

$$\frac{d\rho}{dz} = \rho\beta\frac{dp}{dz} \quad \text{.....3.1.17c}$$

substituting into the equation for ∇p

$$\frac{d\rho}{dx} = \rho^2 g\beta\frac{dH}{dx} + \beta p\frac{\partial\rho}{\partial x} \quad \text{.....3.1.18}$$

$$\frac{\partial\rho}{\partial x} = \left[\frac{1}{1-\beta p}\right]\rho^2 g\beta\frac{dH}{dx} = \rho^2 g\beta\frac{\partial H}{\partial x} \quad \text{.....3.1.19a}$$

$$\frac{\partial\rho}{\partial y} = \left[\frac{1}{1-\beta p}\right]\rho^2 g\beta\frac{dH}{dy} = \rho^2 g\beta\frac{\partial H}{\partial y} \quad \text{.....3.1.19b}$$

$$\frac{\partial\rho}{\partial z} = \left[\frac{1}{1-\beta p}\right]\rho^2 g\beta\left[\frac{dH}{dz} - 1\right] = \rho^2 g\beta\left[\frac{\partial H}{\partial y} - 1\right] \quad \text{....3.1.19c}$$

from Darcy's law

$$q_x = -K_x\frac{\partial H}{\partial x} \quad q_y = -K_y\frac{\partial H}{\partial y} \quad q_z = -K_z\frac{\partial H}{\partial z} \quad \text{.....3.1.20abc}$$

substitute into the preceding equation, q_x, q_y, q_z, $\frac{\partial\rho}{\partial x}$, $\frac{\partial\rho}{\partial y}$, $\frac{\partial\rho}{\partial z}$

$$\rho^2 g\beta\left[K_x\left(\frac{\partial H}{\partial x}\right)^2 + K_y\left(\frac{\partial H}{\partial y}\right)^2 + K_z\left(\frac{\partial H}{\partial z}\right)^2 - K_z\left(\frac{\partial H}{\partial z}\right)\right] + \rho\frac{\partial}{\partial x}\left(K_x\frac{\partial H}{\partial x}\right) + \rho\frac{\partial}{\partial y}\left(K_y\frac{\partial H}{\partial y}\right) + \rho\frac{\partial}{\partial z}\left(K_z\frac{\partial H}{\partial z}\right) = \rho S_s\frac{\partial H}{\partial t} \quad \text{....3.1.21}$$

Considering that $\beta = 4.637 \times 10^{-10}$ m^2/N or 3.3×10^{-6} in^2 / lb and that at low angles of flow $\left(\frac{\partial H}{\partial z} << 1\right)$ the first bracket term may be neglected. The following equation is obtained:

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial H}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial H}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial H}{\partial z}\right) = S_s \frac{\partial H}{\partial t} \quad \text{.....3.1.22}$$

This equation applies to anisotropic, heterogeneous, unsteady state for unconfined and confined groundwater flow. H is the observed data and K is the parameter to be defined by median, mean, standard deviation, and properties of probabilities.

3.2 INVERSE MODELING

3.2.1 Inverse Methodology

An inverse method can be characterized by: 1) the way it describes spatial parameter variability; 2) the forward equation it uses to relate parameters to measurements; 3) the performance criterion it uses to define calibrated parameter estimates; and 4) the solution technique it uses to find these estimates. Although all four factors are important, inverse algorithms differ most significantly in their approaches to parameterization. Parameterization deserves special consideration because it has a strong influence on the configuration of an inverse problem and on the physical plausibility of its solution.

3.2.2 Inverse Groundwater Methodology

Groundwater inverse methods have been reviewed by McLaughlin and Townley (1996), Eppstein and Dougherty (1996), and Keidser and Rosbjerg (1991). These inverse methods can be characterized as follows: inverse direct, inverse indirect, stochastic, and Monte Carlo.

The inverse direct method finds hydraulic conductivity or transmissivity within each model element that makes the model fit the hydraulic head at every node point of the model. It is assumed that the true hydraulic head is known in every node point.

This assumption is crude because measurements of head are scattered over a relatively small number of wells within the study area. Head interpolation from the uncertain measurements is therefore necessary to compute the hydraulic head at all node points.

The inverse indirect method estimates a relatively small number of parameter values that produce a best fit of model-computed parameters to observations.

The stochastic method generates parameters using statistical functions that are assumed to describe the spatial distribution the parameter within the studied area. The estimated parameters are assumed to produce the best fit of model computed parameters to measured parameters.

The Monte Carlo method generates a large number of probable parameter realizations to compute the parameters at observation points. The realizations that produce the best fit of computed parameters to measured parameters are selected to be the best estimate of the parameters. Best fit for hydraulic head is defined as the sum which minimizes the differences between computed and observed heads.

3.2.3 Parameters in Groundwater Models

The groundwater flow equation determines fluid flow in an aquifer. The hydraulic conductivity K and/or the Transmissivity, T=KB where B varies spatially as the aquifer thickness, and the storage coefficient S, which represents the volume of water released from storage per unit area of aquifer per unit decline in head, are the major parameters required for water or other fluid flow within an aquifer.

CHAPTER 4

WATERSHED AND BASIN DESCRIPTIONS

4.0 INTRODUCTION

The study area references for the geology and hydrogeology in this dissertation are: the Geological Society of Iowa Guidebook 58 (Simpkins, 1993); Iowa Department of Natural Resources Geologic Survey Bureau Guidebook Series No. 20 (Simpkins, 1996); the State of Iowa Department of Natural Resources web pages (www.iowadnr.com); the Iowa Geological Survey Open File Reports 82-8 WRD for Boone County and 82-85 WRD for Story County (Thompson, 1982); and the USDA ARS Walnut Creek Watershed Research Protocol Report (Sauer and Hatfield, 1994).

4.1 WALNUT CREEK

4.1.1 Landforms

The Walnut Creek watershed in north central Iowa is 5175.8 hectares (19.98 square miles). Walnut Creek watershed is primarily an agricultural watershed within the low-relief landscape of the Des Moines Lobe. (Figure 4.1)

Three major ice advances occurred in Iowa during the late Wisconsinan time (14,000 to 12,000 B.P.): the Bemis, the Altamont, and the Algona. Walnut Creek lies within the most southern Bemis advance in the lower Des Moines Lobe. The manner in which the glacier dropped its cargo produced low-relief landscapes punctuated by distinct ridges marking the limits of major ice advances. The flat undulating surface of the glacial till produced "swell and swale" shapes with

unconnected drainages. The unconnected depressions are often called potholes and they collect water from the local landscape forming wetlands. Water levels in small capture basins within these wetlands rise and fall depending upon snowmelt and the long-term variable rainfall.

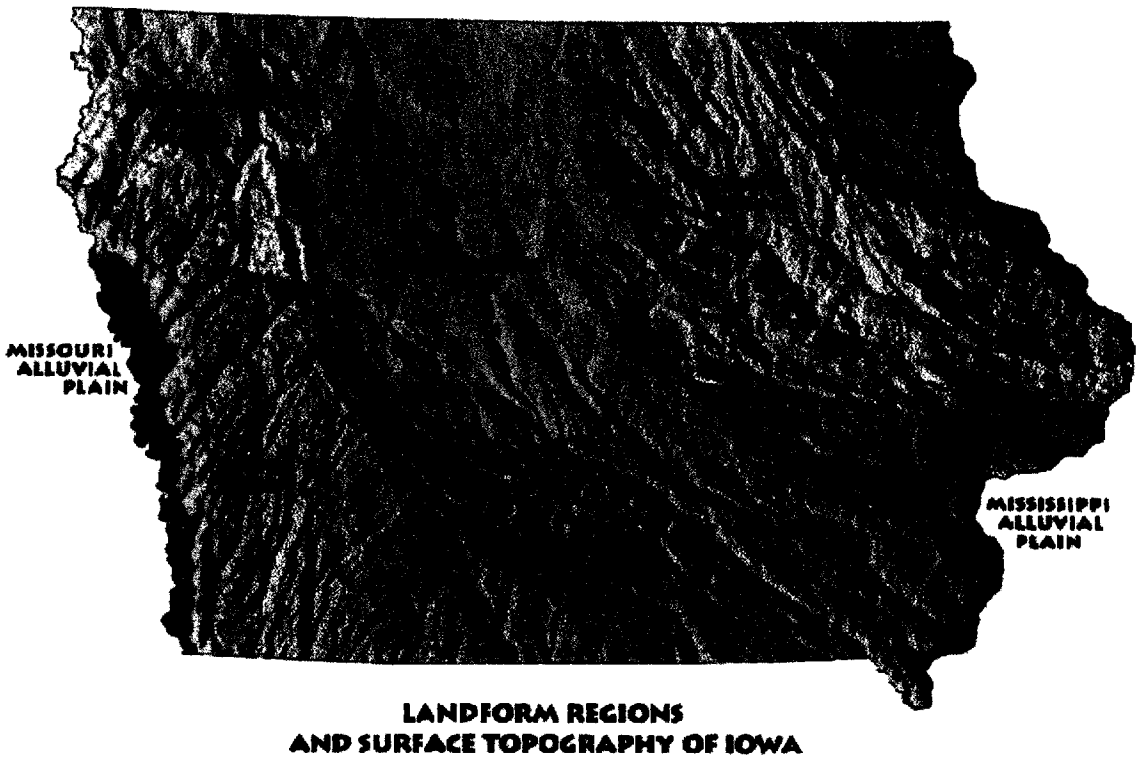


Figure 4.1. Landforms within the state of Iowa (Iowa DNR)

The low-relief landscape also produces a water table that is close to the surface. The shallow aquifer often contributes to the water of pothole wetlands. Low spots contain poorly drained, dark-colored soils indicating sites of previously ponded water and abundant organic accumulations. These low spots are good for wetland habitat; however, incomplete surface drainage is an impediment to agricultural productivity and native wetlands were drained to make way for agriculture lands. These agricultural lands have tile lines laid 1 to 2 meters below the surface to enable

water once contained by pothole wetlands to be transported to surface drainage networks. The uplands of Walnut Creek have a network of drainage tile. Figure 4.2 details this network where the dashed lines are tile drainage lines (Sauer and Hatfield, 1994).

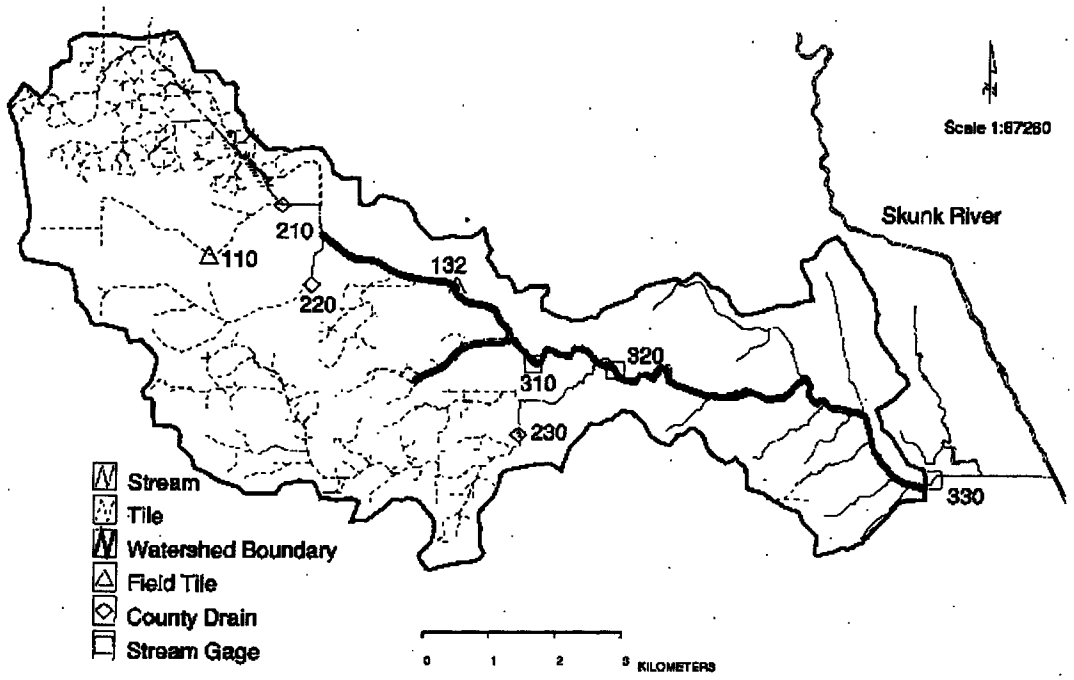


Figure 4.2 Tile Drainage Lines within Walnut Creek (Sauer and Hatfield,1994)

4.1.2 Soils

Soils in Walnut Creek watershed were formed in calcareous glacial till deposited within the Des Moines Lobe during Wisconsinan glaciation. The Clarion-Nicollet-Canisteo soil association characterizes the soils within the watershed and consists of well-drained Clarion and Lester soils on higher or sloping areas, somewhat poorly-drained Nicollet soils on convex side slopes, Canisteo and Webster soils on poorly drained low areas and drainage ways, and very poorly drained

Okoboji and Harps soils in closed depressional areas. (Andrews and Dideriksen, 1981) and (Dewitt, 1984)

Canisteo is a poorly drained, moderately permeable soil found on nearly level swales on uplands. The soil is classified as a fine-loamy, mixed, mesic Typic Haplaquoll and was formed in calcareous till or local alluvium under water tolerant grasses. The surface soil is black, silty clay loam to clay loam and ranges from 35 to 60 cm in thickness. The subsoil ranges in texture from loam to clay loam. The subsolum has a texture or loam or clay loam. The available water capacity is high.

Clarion soil is classified as a fine-loamy, mixed, mesic Typic Hapludoll. This is a well-drained soil found on gently to strongly sloping knolls and side slopes with slopes ranging from 2 to 14 percent. It was formed in till under prairie vegetation. A typical soil profile consists of a dark surface layer of loam 25 to 45 cm thick. The thickness of the subsolum ranges from 60 to 100 cm with the subsoil being a brown loam to clay loam. This moderately permeable soil has a high available water capacity.

Harps is classified as a fine-loamy, mesic Typic Calciaquoll. It is nearly level, 0 to 2 percent, poorly drained calcareous soil on upland flats surrounding closed depressional areas. The surface soil is typically black loam or clay loam and is 25 to 55 cm thick. The subsoil is from 75 to 135 cm thick, gray or dark gray in color, and loam in texture. The subsolum is a mottled gray loam. The permeability is moderate and available water capacity is high.

Lester is a gently to moderately sloping, 2 to 9 percent slopes, well drained soil formed in glacial till on convex upland knolls and ridge tops. It is classified as fine-loamy, mixed, mesic Mollic Hapludalf and was formed under alternating forest and prairie vegetation. Permeability is moderate and available water capacity is high. The subsurface soil is typically a shallow, less than 20 cm, very dark grayish brown sandy loam to clay loam. The subsoil is a brown to yellowish brown, mottled clay loam, 60 to 120 cm thick. The subsolum is a brown, mottled loam and clay loam.

Nicollet is a somewhat poorly drained, moderately permeable soil formed on uplands from loamy glacial till under prairie vegetation. Slopes range from 1 to 3 percent and are usually slightly convex with low relief. Nicollet is classified as a fine-loamy, mixed, mesic Aquic Hapludoll. The surface soil is black and very dark gray loam to clay loam or silt loam in sand content and is about 27 cm thick. The subsoil is a dark grayish brown loam or clay loam and is about 50 cm thick. The subsolum is a grayish brown and olive gray loam. Nicollet soils usually occupy the landscape between the Clarion soils that are upslope and the Canisteo or Webster soils down slope. The available water capacity is high.

Okoboji is classified as a fine, montmorillonitic, mesic Cumulic Haplaquoll and was formed in depressional areas on uplands from local sediments from surrounding uplands. This very poorly drained, moderately slow permeability soil is subject to periodic ponding and has slopes of 0 to 1 percent. The soil surface is typically black and ranges in texture from mucky silt loam to silty clay loam and is 60 to 90 cm thick. The subsoil is a heavy silt clay loam or silty clay and extends to a

depth of 100 to 150 cm. The subsolum is silt loam to silty clay loam and may contain sand lenses in places. Okoboji soils have a high available water capacity.

Webster is a nearly level, 0 to 2 percent, poorly drained soil formed on upland flats and drainageways from loamy glacial till and glacial sediments. This moderately permeable soil is classified as a fine-loamy, mixed, mesic Typic Haplaquoll. The surface soil is typically 35 to 60 cm thick and very dark gray clay loam or silty clay loam. The very dark gray and olive gray subsoil extends from 60 to 120 cm in thickness and has a texture of clay loam or silty clay loam high in sand content. The permeability is moderate and water holding capacity is high.

4.1.3 Geology

The geology of Walnut Creek has two components, bedrock and surficial sediments. The bedrock aquifers are the St. Peter-Jordan aquifer, Silurian-Devonian aquifer, and the Mississippian aquifer. The unconsolidated materials that overlie the bedrock include sand and gravel aquifers and loess and till. These layers provide the parent material for the soils in the Des Moines Lobe landform region and a buffer between the surface activities and the ground water resources.

The hydrogeologic setting of most of the Walnut Creek watershed consists of Paleozoic bedrock units and bedrock valley sand and gravel deposits overlain by Pleistocene till, loess, colluviums, and paleosol units that are effective confining units. Where Walnut Creek flows out of the till uplands, it traverses the modern flood plain of the South Skunk River underlain by Holocene sand and gravel and modern fine-grained flood deposits. Paleozoic units that subcrop beneath the Quaternary deposits include

Pennsylvanian shale and sandstone and Mississippian limestone. These units strike northwest to southeast and dip to the southwest into the Forest City structural basin.

The bedrock topography intersects mostly Mississippian-age rocks beneath Walnut Creek but Pennsylvanian rocks form the bedrock surface in much of the area around Walnut Creek. The depth to bedrock may be as much as 115 m. The major Mississippian aquifers are found in the Gilmore City and Hampton formations that yield generally less than 400 L/min although domestic supplies of less than 40 L/min can be found in less productive units.

Bedrock valleys form an extensive network in central Iowa where they often contain sand and gravel aquifers. Two such valleys, the Jordan and Skunk, underlie Walnut Creek watershed. These valleys were likely pre-glacial stream valleys that later served as meltwater channels during the Pre-Illinoian glaciations and, in some areas, late Wisconsinan glaciations. Although these buried valleys contain coarse sand and gravel in many areas there is evidence that, in the Walnut Creek area, the Skunk Valley contains Pre-Illinoian till separating two units of sand and gravel. Quaternary stratigraphic units comprise the dominant water-bearing strata. The youngest unit consists of alluvial deposits of the DeForest Formation that occupy areas immediately adjacent to the creek and have hydrologic significance downstream from the till uplands.

4.1.4 Topography

Walnut Creek is located within the South Skunk River basin hydrologic unit code (HUC) 07080105. On the west is the Middle Des Moines (HUC 07100004), to

the north is Boone River (HUC 07100005), and to the east are three basins: Upper Iowa (HUC 07080207), Middle Iowa (HUC 07080208), and North Skunk (HUC 07080106).

4.1.5 Relief

Low relief swell and swale topography, oriented southwest to northeast, with an average relief of several meters characterizes the till within the watershed. Topography in the western portion of the watershed is relatively level with only minor variations in relief around the depressional areas. There is more relief in the eastern portion of the watershed. The eastern half of the watershed has increased surface relief due to the head cutting of Walnut Creek before Walnut Creek discharges into the Skunk River.

4.1.6 Stream and Bedrock Profile

Using data from the Iowa Department of Natural Resources and the Agricultural Research Service National Soil Tilth Laboratory Walnut Creek report, data along the Walnut Creek streambed and Wisconsinan and Illinoian bedrock profiles is presented in tabular and cross section view. Figure 4.3 contains a plot of the profiles. The top profile is the streambed. The middle profile is boundary between the oxidized and unoxidized late Wisconsinan till. The bottom profile is the top of the pre-Illinoian till layer.

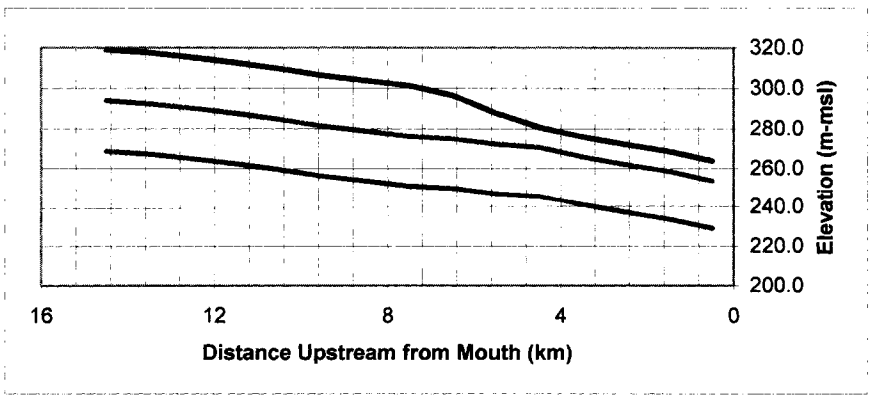


Figure 4.3 Walnut Creek Geologic Profiles (Sauer and Hatfield,1994)

4.2 UPPER SKUNK RIVER BASIN

4.2.1 Upper Skunk Basin

The basin draining into the Skunk River to the confluence with the Walnut Creek watershed is 155,998 hectares (or 602.32 square miles). The Walnut Creek watershed is located in the lower western section of the Upper Skunk.

4.2.2 Upper Skunk Geology

The geological layers of central Iowa include combinations of glacial drift, shale, limestone, and sandstone. Table 4.1 contains a description of the geology and hydrogeology. (Thompson, 1982)

Age	Rock Unit	Description	Thickness (m)	Hydro-geologic Unit	Water Bearing Characteristics (L/m)
Quaternary	Alluvium	Sand gravel silt clay	0-120	Surficial aquifer	Fair-large (37-1890)
	Glacial Drift	Till &scattered			Low (< 37)
	Buried channel	Sand gravel silt clay			Small to large
Pennsylvanian	Marmaton Group	Shale and limestone	0-65	Aquiclude	Low yields
	Cherokee Group	Shale clay silt sandstone			
Mississippian	Meramec Series	Sandy Limestone	30-90	Aquifer	Fair to low
	Osage Series	Limestone and			
	Kinderhook series	Limestone and Dolostone			
Devonian	Maple Hill shale	Shale and limestone	45-75	Aquiclude	None
	Cedar Valley Limestone	Limestone and Dolostone	150-166	Aquifer	Fair to low
Silurian	Undifferentiated	Dolostone	27-38	Aquifer	Low
Ordovician	Maquoketa	Shale and Dolostone	304-327	Aquiclude	None
	Galena	Dolostone		Aquiclude	Low
	Decorah	Limestone		Aquiclude	None
	St. Peter	Sandstone		Aquifer	Fair
	Prairie du Chien	Dolostone		Aquifer	High (> 1890)
Cambrian	Jordan Sandstone	Sandstone		Aquitard	Low
	St. Lawrence	Dolostone		Dresbach	High to Low
Precambrian	St. Lawrence	Dolostone		Base	None

Table 4.1 Geology and Hydrogeology in Central Iowa (Thompson, 1982)

The late Wisconsinan till in central Iowa overlies a Wisconsinan (Peoria) loess unit deposited between 14,000 and 17,000 years B.P. and consists primarily of silt and some clay. Figure 4.4 displays a transect of the Wisconsinan layers.

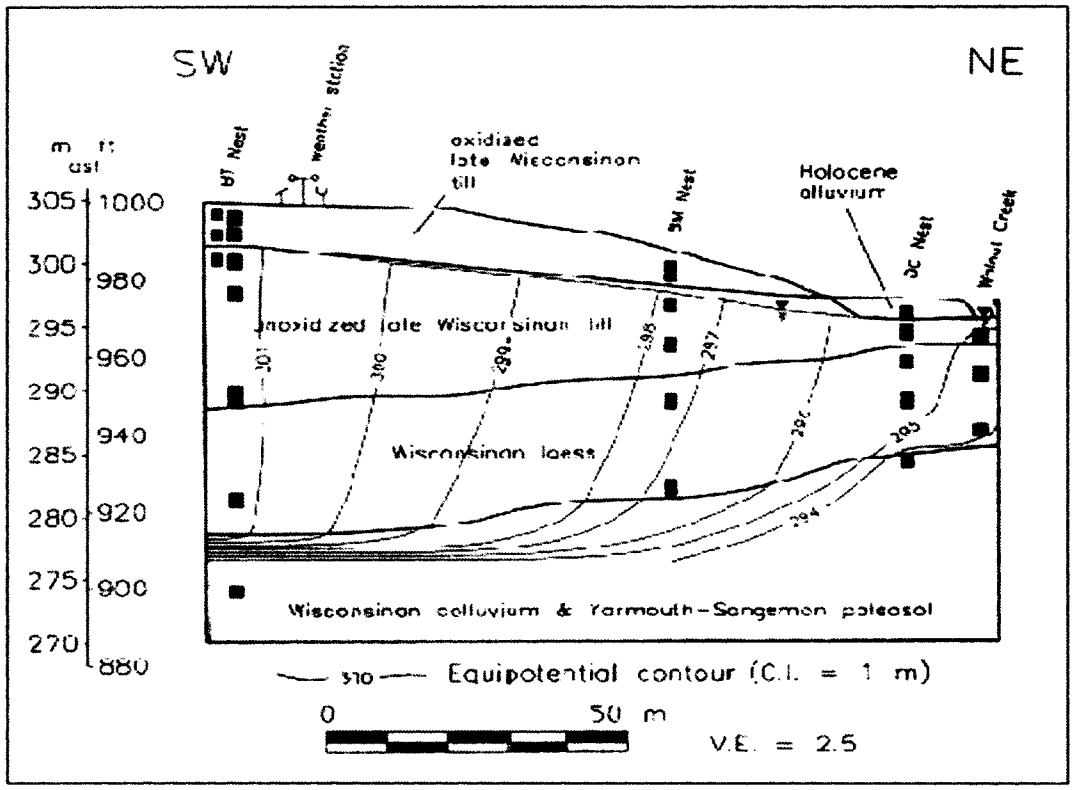


Figure 4.4 Wisconsinan geologic layers in central Iowa (Iowa DNR)

Hydraulic conductivity values of the oxidized Wisconsinan till are on the order of 10^{-4} to 10^{-5} m/s. Unoxidized Wisconsinan till and loess comprise a greater thickness (approximately 80 meters) and K values are much lower on the order of 10^{-8} to 10^{-9} m/s. Pre-Illinoian till underlies the Wisconsinan till. The K values of this glacial sequence are less than the Wisconsinan units above and approach 10^{-11} m/s. While there is only an order of magnitude of 10^{-3} between each layer, there is an order of magnitude of 10^{-6} between the top oxidized Wisconsinan till and the pre-Illinoian till.

4.2.3 Upper Skunk Bedrock Topology

The bedrock topology of the Upper Skunk River basin varies by as much as 115 meters in thickness. (Figure 4.5)

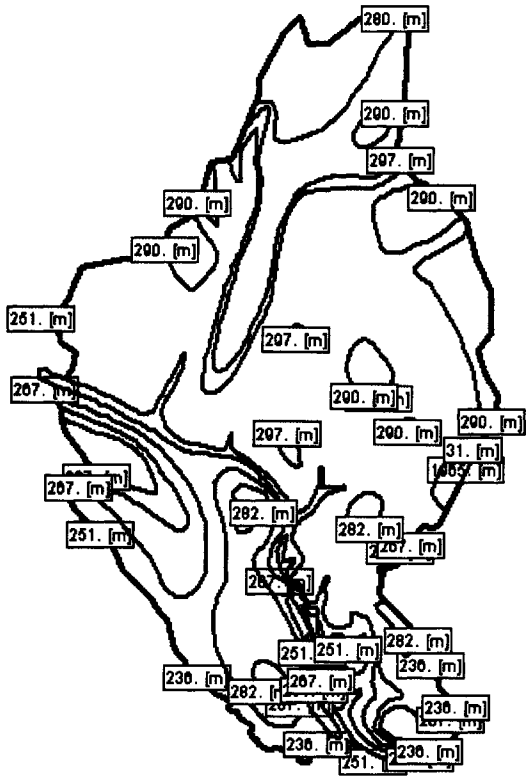


Figure 4.5 Upper Skunk River basin bedrock topology meters-msl (Iowa DNR)

4.3 SUMMARY OF MODELING CRITERIA

Two geologic layers make up both the Walnut Creek watershed and the Upper Skunk River basin. These top two layers are late Wisconsinan till (oxidized and unoxidized) and will be modeled for both inverse Walnut Creek and forward Upper Skunk basin models. The layer below the late Wisconsinan till is a Pre-Illinoian till which is a confining layer because of its low hydraulic conductivity ($K < 1E\text{-}11$ m/s). Research indicates that the top layer has a higher hydraulic conductivity (K ranges

between 1E-04 m/s and 1E-05 m/s) than the 2nd layer (K ranges between 1E-08 m/s and 1E-09 m/s) thus indicating an unconfined system.

The thickness of the top oxidized layer varies from two to nine meters. Most of the two meter thicknesses are along the rivers where the cutting action of the water has reduced this surface thickness. The thickness of the top layer varies from four to nine meters through the majority of the watershed and basin. Three different model configurations will be made in the Walnut Creek inverse modeling, one with a five meter thickness, a second with a six meter thickness, and a third with a seven meter thickness.

CHAPTER 5
INVERSE GROUNDWATER MODELING OF WALNUT CREEK

5.0 INTRODUCTION

The US Geological Survey (USGS) MODFLOW 2000 groundwater model parameter estimation process uses a least squares Gauss-Newton (LSG) optimization method for minimizing the objective function and for calculating parameters in an inverse procedure. The Gauss-Newton optimization method is specifically designed for minimizing a sum of squares objective functions. The Gauss-Newton method is different from the Newton's method since the Hessian matrix is calculated using only the first derivatives. In the generation of a Gauss-Newton sequence for inverse solution, several problems may be encountered. First, it is possible that the search sequence does not converge, the solution matrix is near singular (elements very close to zero), or the displacement vector is so large that parameter values are no longer in the admissible region. In order to avoid these difficulties, the modified Gauss-Newton algorithm was developed. The modified Gauss-Newton method includes an additional term to avoid the singularity.

5.1 ESTABLISHING THE INVERSE MODEL

5.1.1 Inverse MODFLOW Modeling

USGS supports version 1.12.01 of MODFLOW as of October 3, 2003. The forward model is a combination of gradual descent and other relaxation methods for solving the groundwater and transport equations. MODFLOWP (Hill, 1992) with roots from UCODE (Poeter and Hill, 1997) was incorporated into MODFLOW as an inverse

model to predict independent variables. Equation 3.1.22 of the groundwater equation can be rearranged to equate head drawdown as a function of time, parameters. and gradients.

$$h_{t+1} - h_t = \frac{1}{S}\left[\frac{\partial}{\partial x}\left(K_x \frac{\partial H}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial H}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z \frac{\partial H}{\partial z}\right)\right]dt + \frac{Q}{S}dt \quad \text{.....5.1.1}$$

Observed data are available for heads (h). The equation's hydraulic conductivity (K) parameters will be estimated by the inverse procedure.

There are four interfaces to MODFLOW: 1) the basic distributed by USGS which allows for data entry and data verification; 2) Visual MODFLOW; 3) Groundwater Modeling System; and 4) ArgusONE. The ArgusONE interface will be used in this study.

5.1.2 Walnut Creek Watershed

Walnut Creek is an intensively monitored watershed due to the US Department of Agriculture-Agricultural Research Service Management Systems Evaluation Area (USDA-ARS MSEA) and the US Environmental Protection Agency Midwest Agriculture Surface/Subsurface Transport and Effects Research (MASTER) programs. The USDA-ARS National Soil Tilth Laboratory (NSTL) oversees the site. The data collected includes: 1) tracking land cover and land unit changes; 2) stream gauging for discharge and suspended sediment; 3) bio-monitoring of aquatic macro-invertebrates and fish; 4) surface water quality monitoring for nitrate; pesticides, BOD, and fecal coliform bacteria; and 5) groundwater quality for pesticides, water chemistry and hydrologic monitoring.

Numerous documents are available including USGS WRI 95-4109. This report, written by R.C. Buchmiller (1995), contains groundwater level maps and well logs. Data for the years 1991 through 1999 are available in GIS and tabular format.

5.2 WALNUT CREEK DATA

5.2.1 Rainfall

Figures 5.1 and 5.2 show accumulated rainfall from Walnut Creek watershed rain gages 701 and 702. A regression analysis of the cumulative precipitation total from the years 1991 through 1998 indicates an average rate between 2.48E-05 and 2.66E-05 mm/s. Figure 5.3 shows a series of linear variations over shorter periods of time within these eight years. Table 5.1 shows the time in seconds and the analysis in the 8+ years. The R^2 value for these linear periods equals or exceeds 0.97 in all but two periods where the R^2 are 0.80 and 0.90. There was a low accumulation of rainfall during these two periods.

5.2.2 Recharge Assessment

There are 13 time periods of recharge applied to the groundwater model. The first is a steady state period equal in rate to the one-tenth of the eight-year average (2.66E-05 mm/s) and equal in duration to 563 days, which is the average lag time from the end of a linear period of rainfall to the response in the aquifer. This initial setting had two purposes: one, to initialize the model itself and fill the cells with moisture; and two, to bring the model time in step with the 563 day time lapse for the well response.

The next 12 time steps varied in duration and recharge and were exactly equal to the length of the linear periods of rainfall and were either one-tenth or two-tenths of the rainfall.

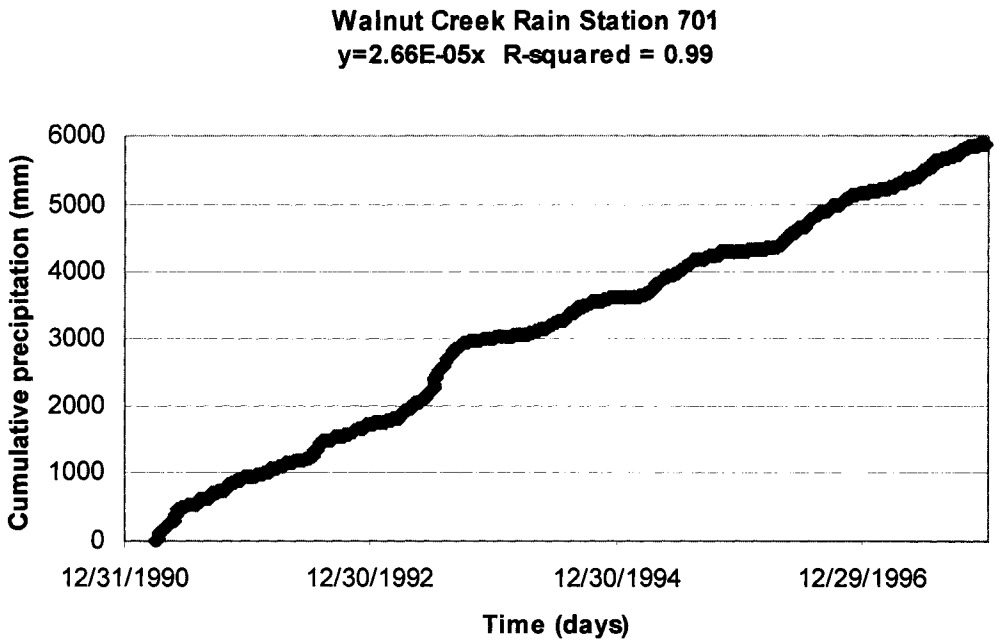


Figure 5.1 Walnut Creek Rainfall for Rain Gaging Station 701

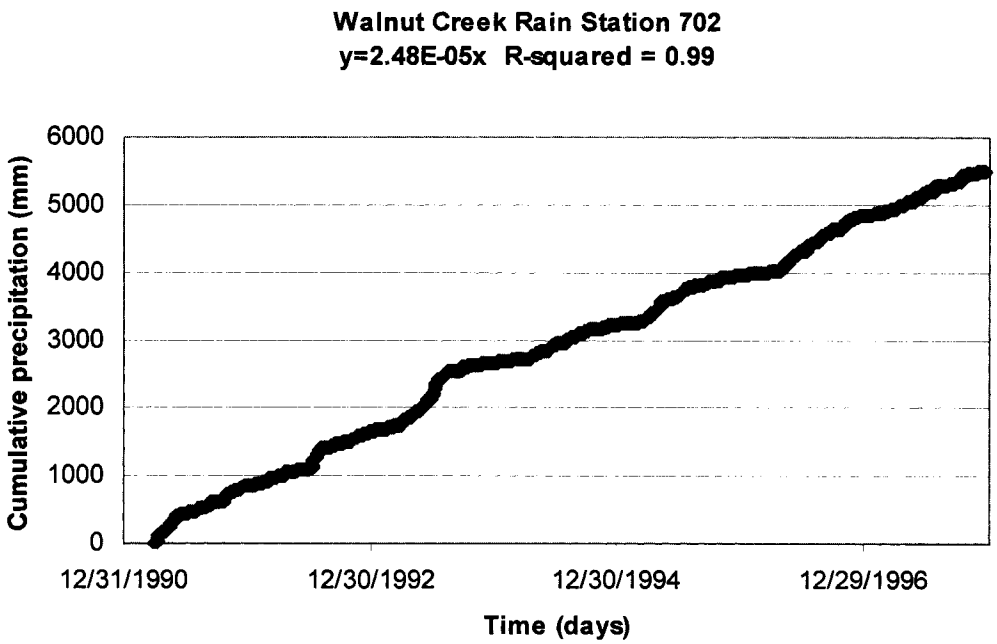


Figure 5.2 Walnut Creek Rainfall for Rain Gaging Station 702

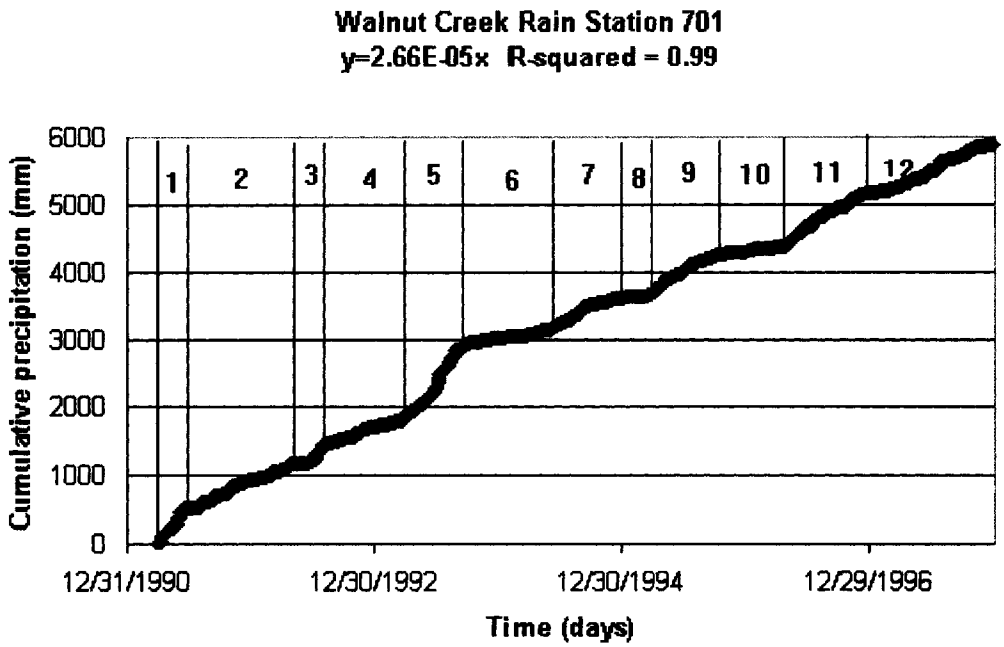


Figure 5.3 Twelve Time Periods of Linear Rainfall

Linear Period	Well Time (seconds)	Rain Beg. (seconds).	Rain End (seconds)	Lag time (seconds)	Lag (days)	Precip. Rate (mm/s)	R^2	Recharge (m/s)
1	10972800	8380800	14256000			7.241E-05	0.97	7.241E-09
2	19526400	14342400	46137600			2.386E-05	0.99	2.386E-09
3	101088000	46224000	57801600	43286400	501	3.382E-05	0.92	3.382E-09
4	114566400	57888000	69984000	44582400	516	1.637E-05	0.97	1.637E-09
5	137548800	70070400	86313600	51235200	593	7.381E-05	0.98	7.381E-09
6	149558400	86400000	103420800	46137600	534	7.879E-06	0.98	7.879E-10
7	173145600	103507200	124070400	49075200	568	2.852E-05	0.98	2.852E-09
8	191980800	124156800	134611200	57369600	664	7.159E-06	0.90	7.159E-10
9	196214400	134697600	150249600	45964800	532	3.248E-05	0.98	3.248E-09
10	214272000	150336000	159235200	55036800	637	6.302E-06	0.80	6.302E-10
11	231724800	159321600	186537600	45187200	523	3.352E-05	0.98	3.352E-09
12	244339200	186624000	220665600	23673600	274	2.534E-05	0.98	2.534E-09
Avg.				48652800	563			3.012E-09

Table 5.1 Regression Analysis of Precipitation Data

5.2.3 Well Analysis

Shallow ground water was observed with a series of NSTL piezometers. These piezometers range from 1 to 15 m in thickness. The position of each well within the watershed was located with a GPS to obtain the location coordinates. Mean sea level elevation for each well was determined by level line surveys. These positions were entered into the geographic information system data base for the watershed.

Figure 5.4 is a plot of one of the 113 wells within Walnut Creek.

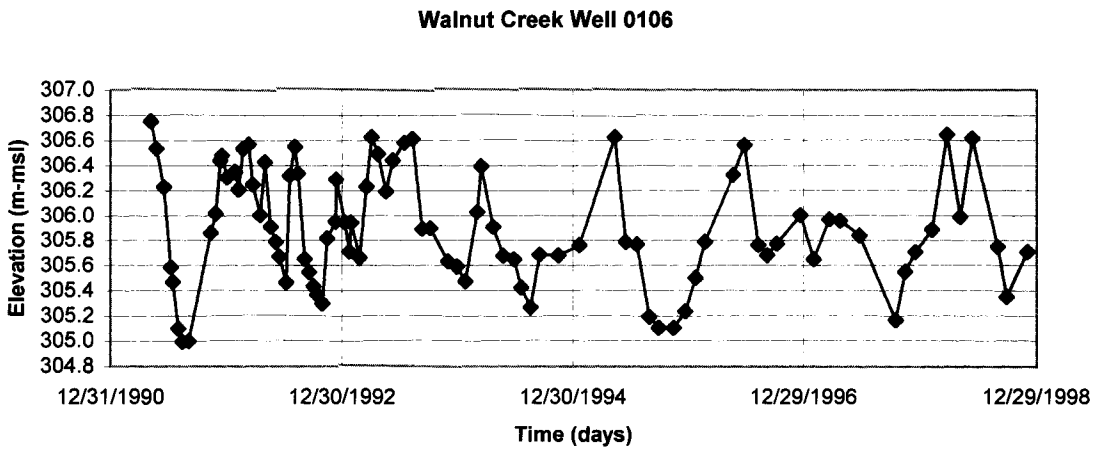


Figure 5.4 Groundwater Elevations for Walnut Creek Well 0106 over an eight year period from 1991 to 1998

Column two of Table 5.1 indicates the time in seconds to each of the peaks for well WC0106. Columns four and five indicate the lag time, in seconds and days respectively, from the end of the linear period of rain to the peak of the well time. The well data and rain data together indicate that there is an average of 563 response days from the end of a linear period of precipitation recharge to the response of the aquifer to this recharge. This varies from a high of 664 to a low of 501 days with a standard deviation of 57 days.

5.2.4 Initial Hydraulic Conductivity Values

Simpkins and Burkart (1996) report that the upper oxidized late Wisconsinan geologic layer has a hydraulic conductivity of 1E-04 to 1E-05 m/s and the second unoxidized late Wisconsinan till has a hydraulic conductivity of 1E-08 to 1E-09 m/s.

5.2.5 Thickness of Layers

There are two layers of late Wisconsinan till within the borders of the Walnut Creek watershed. The top layer is oxidized and varies from two meters to eight meters in thickness throughout the watershed. The second layer is unoxidized and varies in thickness to 80 meters. The second layer extended from the top layer bottom elevation to the bedrock bottom elevation.

5.3 MODELING DATA

There are two interfaces within the ArgusONE groundwater model. One interface is used to control the import, export, and modification of the geospatial and layer data. This software interface comes with ArgusONE when it is installed. The second interface is used to control the MODFLOW model parameters. This second software interface is available with the download of the MODFLOW graphical user interface (mfgui) from the USGS website. The mfgui software also includes MODFLOW and other USGS programs.

5.3.1 ArgusONE Geospatial Interface

The ArgusONE interface allows for use of a variety of geospatial formats to define the domain and as data input such as the hypsography contour data for the surface

and bedrock topologies. Using the National Soil Tilth Laboratory hypsography surface elevation contour data, the surface contour shapefile was entered into the interface as an image. Using the contours as an image, the watershed was delineated as a domain. A grid of 600 meters was used to "tile" the watershed. The contours were imported into the top elevation of layer one as a data layer. Similarly, the bottom elevation of the second layer was populated with data using the Iowa hypsography bedrock topology shapefile.

There were 113 sets of well data within the Walnut Creek watershed. (Section 10.1 of the appendices lists the well station's coordinates and elevation information.) Well data were entered into a layer by importing an EXP ArgusONE format data file. The first column is the name, the second and third are the UTM X and Y coordinates, and the remaining columns are the head data at the specified time periods. Section 10.2 contains the well observations in meters.

Using the layers window, the elevation of the bottom of the top unit was entered as an expression. This expression was the elevation of the top of unit one minus five, six, or seven meters. In this way the thickness of the top layer was set to five, six, or seven meters. The elevation of the top of the second unit was also entered as an expression using the layers window. This expression was the elevation of the bottom of the top unit. In this way, the thickness of the bottom layer was equal to the difference between the surface elevation and the bedrock elevation minus the five, six, or seven meters top thickness.

The recharge was entered into the groundwater model using the layer interface in ArgusONE as shown in Figure 5.5. One-tenth of the rainfall from rain gage 701 was used as the input recharge.

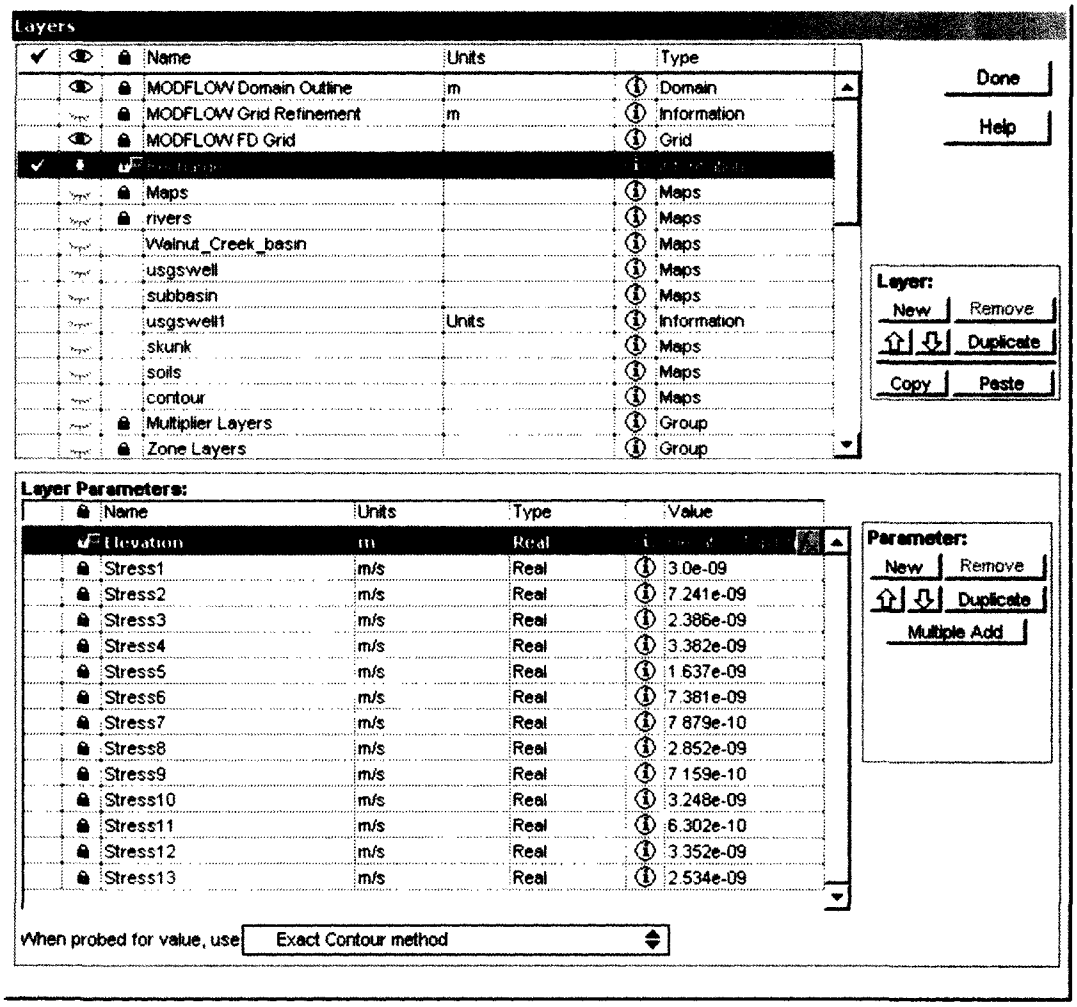


Figure 5.5 ArgusONE interface for Entering Recharge into the Walnut Creek Groundwater Model (National Soil Tilth Laboratory Rain Gaging Station 701)

The prescribed heads are elevations of the stream at specific locations. This data can be determined by observing where the stream crosses a contour line. The river hydrography data does not include elevation within the data. It was necessary to create a shapefile of these locations with the elevations for import into ArgusONE. The National Soil Tilth Laboratory (NSTL) hypsography contour and streambed data were intersected to identify the prescribed heads. Figure 5.6 shows the location and elevation of the prescribed heads.

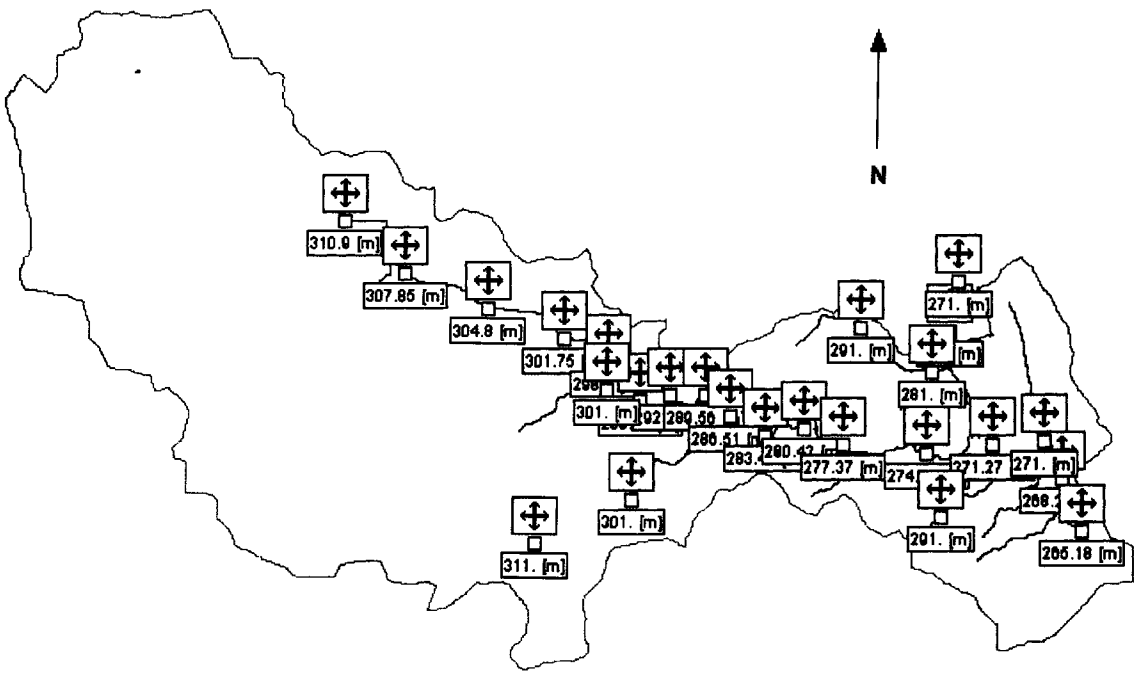


Figure 5.6 Prescribed Heads for Walnut Creek watershed

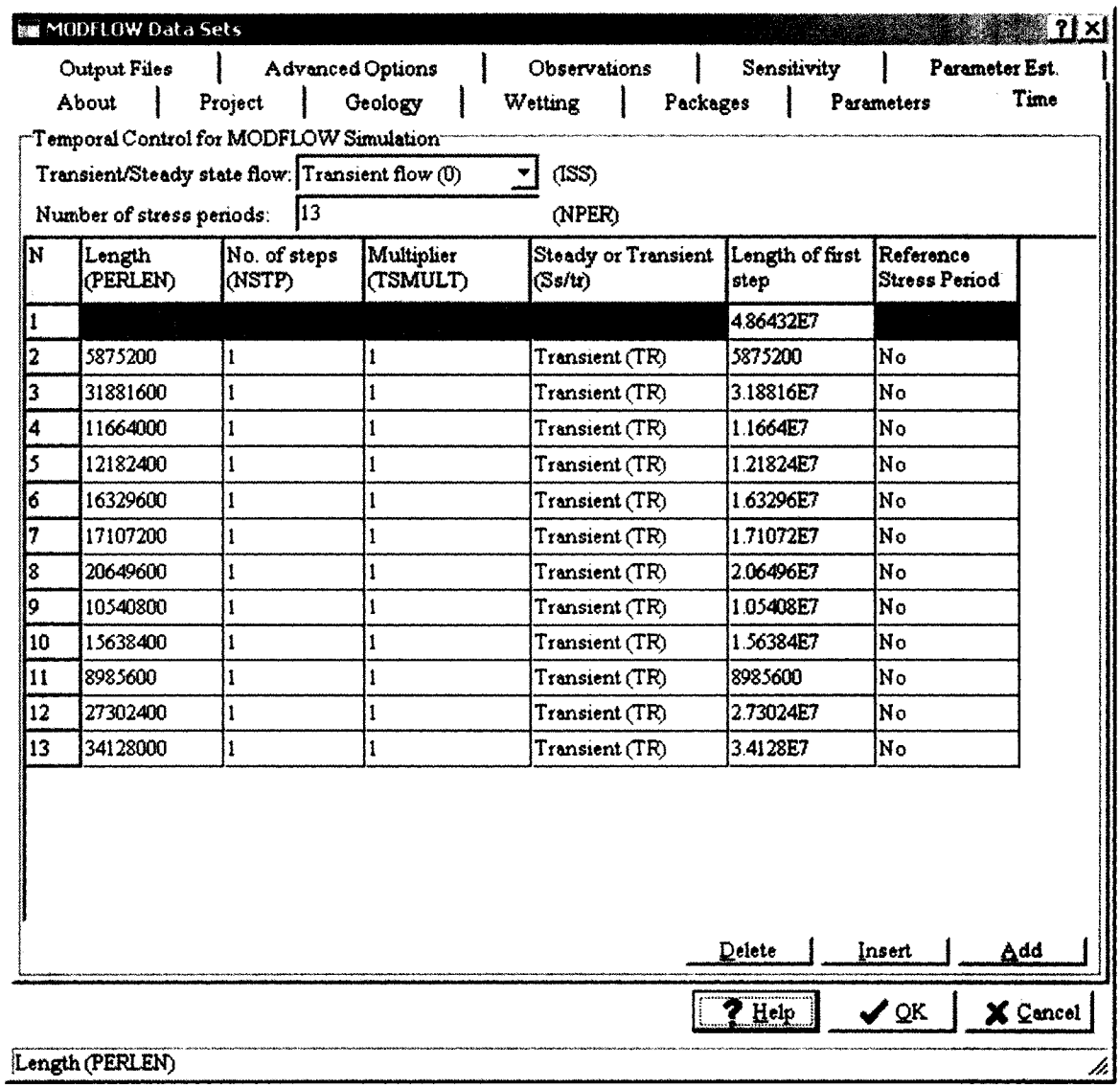
5.3.2 ArgusONE Model Interface

The US Geological Survey MODFLOW graphical user interface plug-in extension (PIE) was used to enter the data for geology, time periods, and options for running the MODFLOW model.

The model times are in seconds. The time for the first period of steady state recharge is equal to 563 days. After this period, there are 12 periods of recharge equal in duration to the 12 linear periods of rainfall. (Figure 5.7)

The geology tab was used to set both the number of layers as well as the type. The output run results shown here are for a convertible layer.

The sensitivity tab is used to set the initial hydraulic conductivity values for each layer as well as the range of acceptable hydraulic conductivity values for the inverse model. (Table 5.2)



N	Length (PERLEN)	No. of steps (NSTP)	Multiplier (TSMULT)	Steady or Transient (Ss/tr)	Length of first step	Reference Stress Period
1					4.86432E7	
2	5875200	1	1	Transient (TR)	5875200	No
3	31881600	1	1	Transient (TR)	3.18816E7	No
4	11664000	1	1	Transient (TR)	1.1664E7	No
5	12182400	1	1	Transient (TR)	1.21824E7	No
6	16329600	1	1	Transient (TR)	1.63296E7	No
7	17107200	1	1	Transient (TR)	1.71072E7	No
8	20649600	1	1	Transient (TR)	2.06496E7	No
9	10540800	1	1	Transient (TR)	1.05408E7	No
10	15638400	1	1	Transient (TR)	1.56384E7	No
11	8985600	1	1	Transient (TR)	8985600	No
12	27302400	1	1	Transient (TR)	2.73024E7	No
13	34128000	1	1	Transient (TR)	3.4128E7	No

Figure 5.7 Periods of Timing for the Walnut Creek Groundwater Modflow Model

Layer	Starting K (m/s)	Lower K (m/s)	Upper K (m/s)
Top (1^{st})	4E-8	3E-5	1E+4
Bottom (2^{nd})	8E-5	7E-20	1E+2

Table 5.2 Initial Hydraulic Conductivity Values

5.4 MODEL RESULTS

Running of the ArgusONE system is a two phase process as the data are first rendered and then passed to the MODFLOW PIE that creates the input MODFLOW program files. The second phase runs the MODFLOW program. Model runs were made with five, six, and seven meter thickness for the top layers.

5.4.1 Hydraulic Conductivity

The MODFLOW inverse groundwater model for Walnut Creek calculated hydraulic conductivity values for the three layers as shown in Table 5.3.

Thickness (m)	Literature Top Layer (m/s)	Literature 2nd Layer (m/s)	Top Layer K (m/s)	2nd Layer K (m/s)
5	1E-04-1E-05	1E-08-1E-09	3.83E-09	1.59E-04
6	1E-04-1E-05	1E-08-1E-09	5.93E-09	1.63E-04
7	1E-04-1E-05	1E-08-1E-09	7.84E-09	1.67E-04

Table 5.3 Resulting Hydraulic Conductivity Values

The aquifer layers were modeled as convertible. The "Thickness" column indicates the top layer thickness in meters. The last two columns are the hydraulic conductivity results from the MODFLOW inverse model.

5.4.2 Convergence

The inverse MODFLOW model for Walnut Creek would not converge to FINAL VALUES in the LSG file unless the initial values for the hydraulic conductivities were relatively close to what the model expected. The starting values were 4E-08 (m/s) for the top layer and 8E-05 (m/s) for the second layer.

5.5 SUMMARY OF MODEL RESULTS

Analysis of the data reveled that there were 12 linear time periods of rainfall and that these 12 periods of recharge were directly related to the 12 peaks in the well data.

Doubling the recharge essentially doubled the hydraulic conductivity values using the Least Squares Gauss-Newton inverse procedure.

Using the published hydraulic conductivity values for the two layers as initial starting values would not make the model converge. The MODFLOW inverse procedure required starting values that indicated a confining aquifer system. These starting values had to be close to the final convergence values. A manual process of changing these starting values was necessary until the model finally converged.

The MODFLOW inverse procedure resulted in a confined aquifer system. This was the result even when the two layers were modeled as convertible.

The hydraulic conductivity values predicted by MODFLOW did not agree with literature. (Table 5.4) MODFLOW results indicate a confined aquifer system that does not agree with published results. (Simpkins, 1993 and Simpkins and Burkart, 1996)

Layer	Literature (m/s)	MODFLOW (m/s)
Top (1^{st})	1E-04-1E-05	5.9E-09
Bottom (2^{nd})	1E-08-1E-09	1.6E-04

Table 5.4 Comparison Hydraulic Conductivity Values

CHAPTER 6

INVERSE GROUNDWATER MODELING WITH LEAST SUM ABSOLUTE DEVIATION AND EXPECTATION MAXIMIZATION

6.0 INTRODUCTION

The Least sum Absolute Deviation (LAD) regression and Expectation Maximization models were incorporated into the USGS MODFLOW model by modifying the MODFLOW FORTRAN PES1GAU1 subroutine module. The LAD procedure regresses the heads in an objective function. The Expectation Maximization (EM) procedure computes the new hydraulic conductivity parameters using the LAD regression coefficients, the observed heads, and the results from the LAD model. The LAD-EM inverse procedure was applied to Walnut Creek watershed in Iowa.

6.1 PROGRAMMING

The MODFLOW FORTRAN source code consists of multiple files. The main module controls the program execution. The source code that controls the inverse modeling of the groundwater flow equation is located in the two modules PES1BAS6 and PES1GAU1. These two "PES" FORTRAN modules contain the parameter estimation source code. Only the PES1GAU1 module has been modified in this study. The FORTRAN source code is listed in Chapter 10 Section 3.

6.1.1 PES1GAU1 MODFLOW Module

The PES1GAU1 MODFLOW FORTRAN module contains the parameter estimation sensitivity and Least Squares Gauss-Newton (LSG) source code. Least sum Absolute Deviation and Expectation Maximization (LAD-EM) FORTRAN source code

was added to this file. This author's contributions are shown in **bold** font. A FORTRAN 95 compiler is required as this code performs memory allocation.

The USGS standard for subroutine naming specifies that subroutines beginning with 'S' are to be called internally within a FORTRAN file. For example, if they are called from another subroutine outside of pes1gau1.f, such as pes1bas6.f, then they do not begin with an 'S'. Subroutines end with two to three characters to designate their function. The main subroutine is PES1GAU1. Table 6.1 contains descriptions of the PES1GAU1 subroutines in the MODFLOW pes1gau1.f FORTRAN source code.

Line	SUBROUTINE	DESCRIPTION
4	PES1GAU1AP	Main entry from MODFLOW into this module
275	SPES1GAU1	Original PES1GAU1 functionality
916	SPES1GAU1DM	Adjusts DMAX to allow extreme values to recover
934	SPES1GAU1IN	Calculates the inverse scaled coefficient matrix
972	SPES1GAU1PR	Print parameter values called from pes1bas6
1139	SPES1GAU1QN	Compute quasi-Newton component of C matrix
1302	SPES1GAU1SL	Compute parameter step lengths with Marquardt
1388	SPES1GAU1PF	Add components to the matrix and the vector
1425	SPES1GAU1CN	Check for parameter values <=0 that should be >0

Table 6.1 PES1GAU1 Subroutines within PES1GAU1.F Source

The MODFLOW pes1gau1.f subroutine PES1GAU1 was separated into SPES1GAU1 in order to insert the LAD-EM subroutines.

6.1.2 Least Absolute Deviation

The USGS standard for subroutine naming was adopted for the insertion of the LAD subroutines. The main subroutine is PES1LAD1. Other LAD subroutines begin with 'S' because they are called internally by PES1LAD1 or other PES1LAD1 subroutines. Table 6.2 contains descriptions of the PES1LAD1 subroutines in the MODFLOW pes1gau1.f FORTRAN source code. The lines numbers are taken from the listing in Chapter 10 Section 3.

Line	SUBROUTINE	DESCRIPTION
295	PES1LAD1	The main entry into the LAD function
1482	SPES1LAD1LT	Main control for LAD regression
1585	SPES1LAD1PR	Print LAD parameter values
1615	SPES1LAD1L1	The L1 normalization solution function
1835	SPES1LAD1CL	Assigns values into V1 from the computed values
1852	SPES1LAD1AL	Multivariate randomized block permutation (MRBP)
1880	SPES1LAD1CA	Calculates the MRBP
1984	SPES1LAD1DT	Calculates distances between data points
2087	SPES1LAD1PV	Calculates value probability <= observed value

Table 6.2 LAD Subroutines within the PES1GAU1.F Source

6.1.3 Expectation Maximization

The LAD regression calculates the parametric statistical values based on the differences between the observed heads and the model predicted heads. This calculation is not sufficient to adjust the dependent variables for convergence. It is necessary to calculate a delta change to be applied to the dependent variables. An Expectation Maximization (EM) procedure was added to the pes1gau.f subroutines to provide this functionality. See Chapter 2 Section 4 for theoretical information.

The USGS standard for subroutine naming was also adopted for the insertion of the EM subroutines. The main subroutine is SPES1LAD1EM. Table 6.3 contains the descriptions of the one EM subroutine and three matrix subroutines in the MODFLOW pes1gau1.f FORTRAN source code.

Line	SUBROUTINE	DESCRIPTION
2189	SPES1LAD1EM	Calculates the expectation maximization
2306	SPES1LAD1MUL	Matrix multiplication
2332	SPES1LAD1TRA	Matrix transposition
2347	SPES1LAD1INV	Matrix inversion

Table 6.3 EM Subroutines within the PES1GAU1.F Source

The MODFLOW main program deletes a "MODFLOW.lad" file at startup time if it exists. Upon entry into PES1GAU1AP, data required for the EM process is retrieved from the MODFLOW.lad file. The new code reads the previous "results" data from the files which includes the columns: weight, first run output, second run output, and the last column which is the observed data. At the end of the columns of data, the last calculation of the LAD regression coefficients is listed and is read into the program. After the LAD analysis has been completed the MODFLOW.lad data file is re-created with the weight in the first column, the third column is copied into the second column, the fourth column is copied into the third column, and the last column contains the observed data. Other than the first column of weights and the last column of observed data, the number of model run columns equals the number of parameters to estimate. The last lines of the file are the newly predicted LAD regression coefficients.

6.1.4 Convergence Criteria

There are two conditions for convergence of the LAD-EM solution. When the sum of the differences between previously computed forward parameters and the current forward parameters is less than the tolerance then one condition is met. When the sum of the differences between the previously computed model heads and the current models heads is less than the tolerance then both conditions have been met and LAD convergence criteria is satisfied. See lines 422 through 439 in Chapter 10 Section 3 for the execution of the tolerance logic. The tolerance is set within the MODFLOW model interface. It is not hard coded in the FORTRAN to a pre-set value. The ArgusONE PIE interface "Edit Project Info Parameter Estimation" tab allows the modeler to adjust the tolerance.

6.2 RESULTS

The same Walnut Creek ArgusONE groundwater model discussed in Chapter 5 was used for the LAD-EM analysis. At runtime, the model path was changed from the USGS mf2k.1_12.exe to the mf2k_lad.exe path. Three model runs were performed for varying thickness from five to seven meters. A listing of model and observed data is shown in Chapter 10 Section 4.

The literature and LAD-EM and MODFLOW inverse groundwater model calculated hydraulic conductivity values for the three layers are shown in Table 6.4.

Thickness (m)	Literature Top Layer (m/s)	LAD-EM Top Layer (m/s)	MODFLOW Top Layer (m/s)	Literature 2nd Layer (m/s)	LAD-EM 2nd Layer (m/s)	MODFLOW 2nd Layer (m/s)
5	1E-04-1E-05	2.57E-04	3.83E-09	1E-08-1E-09	1.08E-05	1.59E-04
6	1E-04-1E-05	2.72E-04	5.93E-09	1E-08-1E-09	1.00E-10	1.63E-04
7	1E-04-1E-05	6.84E-04	7.84E-09	1E-08-1E-09	1.00E-10	1.67E-04

Table 6.4 Comparison of Hydraulic Conductivity Values

The aquifer layers were modeled as convertible. The "Thickness" column indicates the top layer thickness. The top layer ranges from two to nine meters. The 2nd and 5th columns are the hydraulic conductivity estimates from literature. The 3rd and 6th columns are the hydraulic conductivity results from the LAD-EM modified MODFLOW inverse model. The 4th and 7th columns are the hydraulic conductivity results from the MODFLOW mf2k model version 1_12. Chapter 10 Section 4 lists the model and observed heads and it appears that the seven meter thickness has a best fit of model to observed heads. Chapter 10 Section 5 lists the differences and contains plots of these differences. Figures 6.1 and 6.2 show the groundwater model head contours for a thickness of seven meters in the MODFLOW and LAD-EM groundwater models.

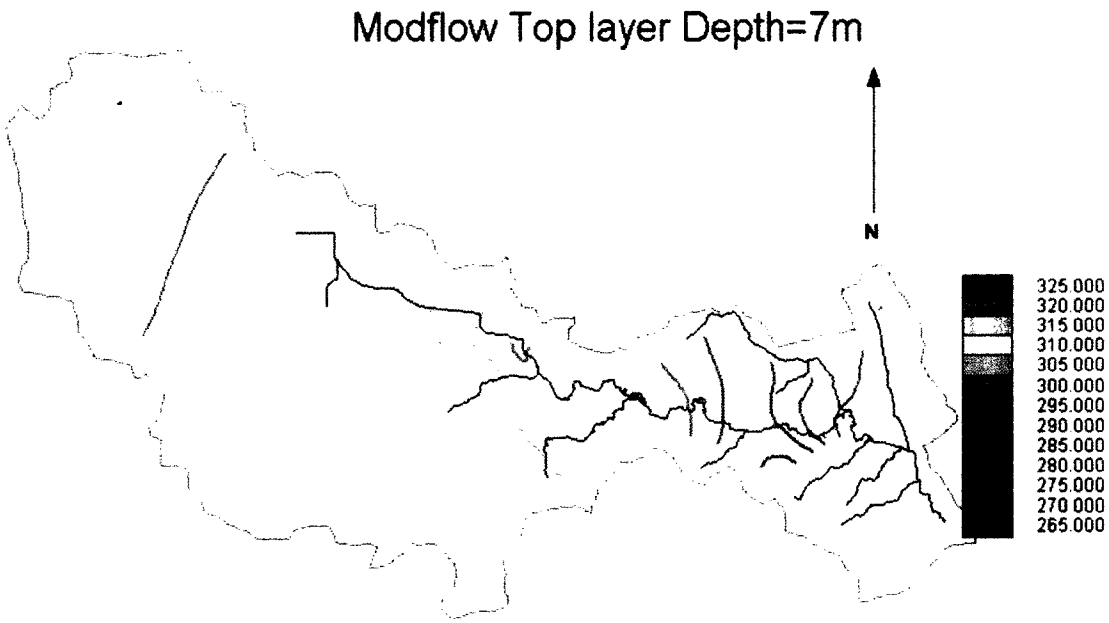


Figure 6.1 Walnut Creek Groundwater Head Contour Map for TOLERANCE=0.003, Thickness=7 meters using MODFLOW

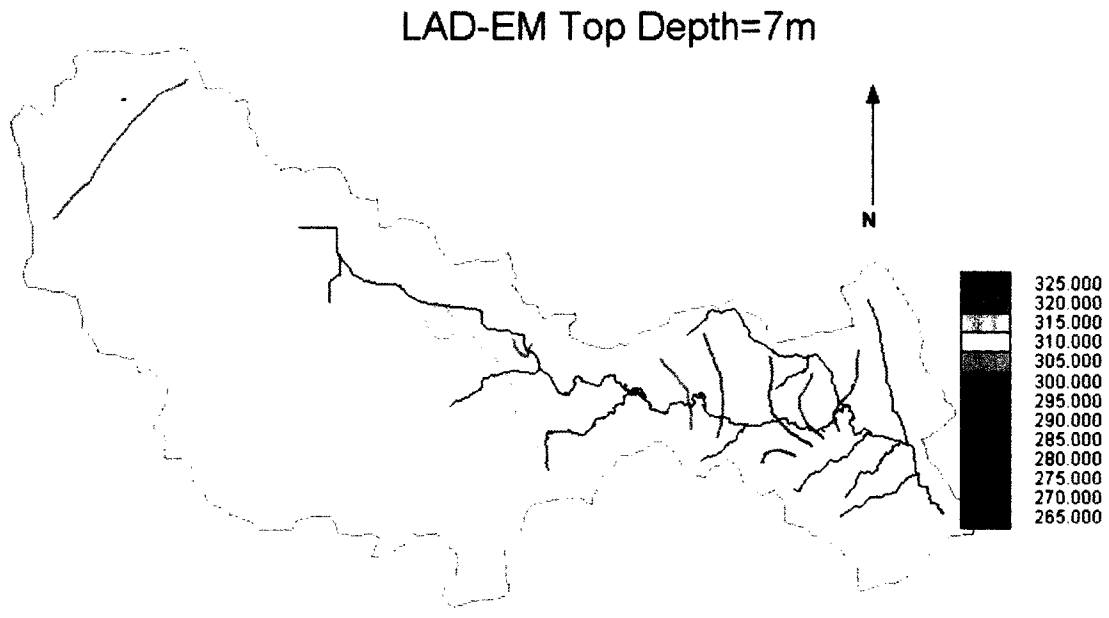


Figure 6.2 Walnut Creek Groundwater Head Contour Map for TOLERANCE=0.003, Thickness=7 meters using LAD-EM

6.3 SUMMARY OF LAD MODEL RESULTS

The LAD-EM modified MODFLOW inverse procedure did not require starting both layers as confined in order to converge.

The LAD-EM modified MODFLOW inverse procedure did not require starting values close to the final convergence values. Convergence was extremely quick due to the Expectation Maximization procedure that calculated the new parameters to be used in the forward model. No manual operation was required since convergence occurred with reasonable starting values for hydraulic conductivity.

The LAD-EM modified MODFLOW inverse procedure resulted in an unconfined aquifer system when the layers were selected as convertible and confined when it was modeled as confined.

The final hydraulic conductivity values predicted by LAD-EM modified MODFLOW for the seven meter thickness unconfined model agreed well with literature.

	Literature	MODFLOW	LAD-EM
Top Layer	1.0E-04-1.0E-05	7.8E-09	6.8E-04
2nd Layer	1.0E-08-1.0E-09	1.7E-04	1.0E-10

Table 6.5 Comparisons of Hydraulic Conductivities

The Expectation Maximization procedure is not sensitive to large differences in heads and is an effective means to back calculate the forward parameter. This author recommends that this be made an option for inverse modeling with the MODFLOW inverse regression procedure.

CHAPTER 7

GROUNDWATER MODELING OF THE UPPER SKUNK BASIN

7.0 BASIN GROUNDWATER MODELING

7.1 ESTABLISHING THE FORWARD MODEL

7.1.1 MODFLOW Modeling

The ArgusONE geospatial interface and USGS MODFLOW Plug-In Extension is used to create a forward groundwater model of the Upper Skunk River basin of 155,998 hectares (or 602.32 square miles). A grid of 1965 meters was used to discretize the basin. The recharge will be the 13 linear periods of recharge over an eight-year period that was used in the Walnut Creek model. There will be two geologic layers: one, an upper oxidized late Wisconsinan till; and two, a lower unoxidized late Wisconsinan till. The area fits within Universal Transverse Mercator Zone 15. It is centrally located within the Des Moines Lobe.

7.1.2 Creating the Datasets

The steps for data creation required manipulation of geographic information system (GIS) data. The Iowa Department of Natural Resources (DNR) hypsography contour shapefiles for Boone, Story, and Hamilton counties were imported as an image. Chapter 10 Section 8 describes Iowa GIS data. Using the contours as a visual guide, the boundary outline (domain) for the basin was delineated. This ArgusONE domain outline was then exported to a file. The export file resembles the Arc/INFO Generate (GEN) format. The format is so close to the GEN format that the public domain GEN2SHP program was modified to convert the ArgusONE

export format file into a shapefile. Chapter 10 Section 6 has the listing of the converted program "Exp2Shp". The exported domain file was converted to a polygon shapefile using Exp2Shp. This author's contributions are highlighted in **bold**.

The next objective was to intersect the recently created domain shapefile with the Iowa DNR hypsography and hydrography layers. The domain shapefile and the Iowa DNR hypsography and hydrography layers were imported into the ESRI ArcMAP program. Using the GeoProcessing Wizard under Tools in ArcMAP and selecting "Intersect two layers", the rivers "water" layer of the hydrography data was intersected with the domain to produce a rivers layer for the Upper Skunk. Similarly, the hypsography elevation contour layer data was intersected with the domain to produce a contour layer for the Upper Skunk.

The river layer set has river length and location information but not elevation. The elevation of the river is needed as input prescribed heads. The next step then was to intersect the river and contour data. This is not a straight forward process since the river and contour GIS data are polylines. GIS polylines consist of multiple lines (arcs) strung together during the digitization. They are not areal and the ArcMAP intersection process is not available for polylines. A different procedure had to be devised in order to intersect the river and contour polylines.

The three county data sets for hypsography contours (topo08, topo40, and topo85) were merged using the "Merge layers together" feature in ArcMAP. The process was repeated for the three hydrography layers (rivers08, rivers40, and rivers85). Using the GeoProcessing Wizard under Tools in ArcMAP the

"Dissolve features based on an attribute" was selected to resolve all the features into the same contour.

The river and contour data were exported into a GEN file using ArcMAP and a plug-in called ET GeoWizards 8.6. In order to individually cross the river with the contour data, the arcs had to be separated. A C program was written to separate all of the river polylines into separate arcs and intersect them. The logic was not trivial but others had worked it out at www.faqs.org. Chapter 10 Section 7 contains the listing of the C program by this author.

The two files were merged to identify the elevation of the river at the intersection with the contour. This GEN format data file was then converted back into a shapefile using GEN2SHP and imported into ArgusONE as a data layer for the prescribed heads. Figure 7.1 shows the locations of the intersections.

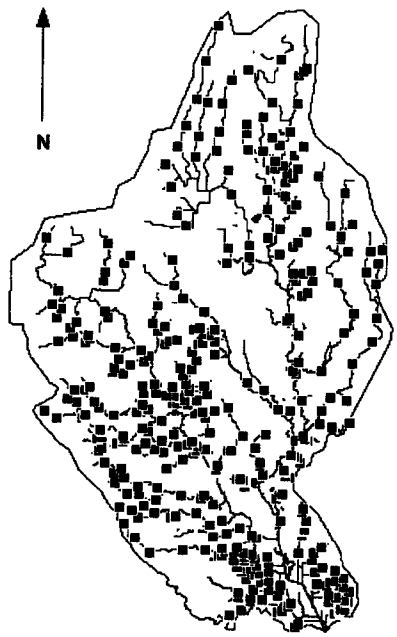


Figure 7.1 Prescribed heads in the Upper Sunk river basin In Iowa

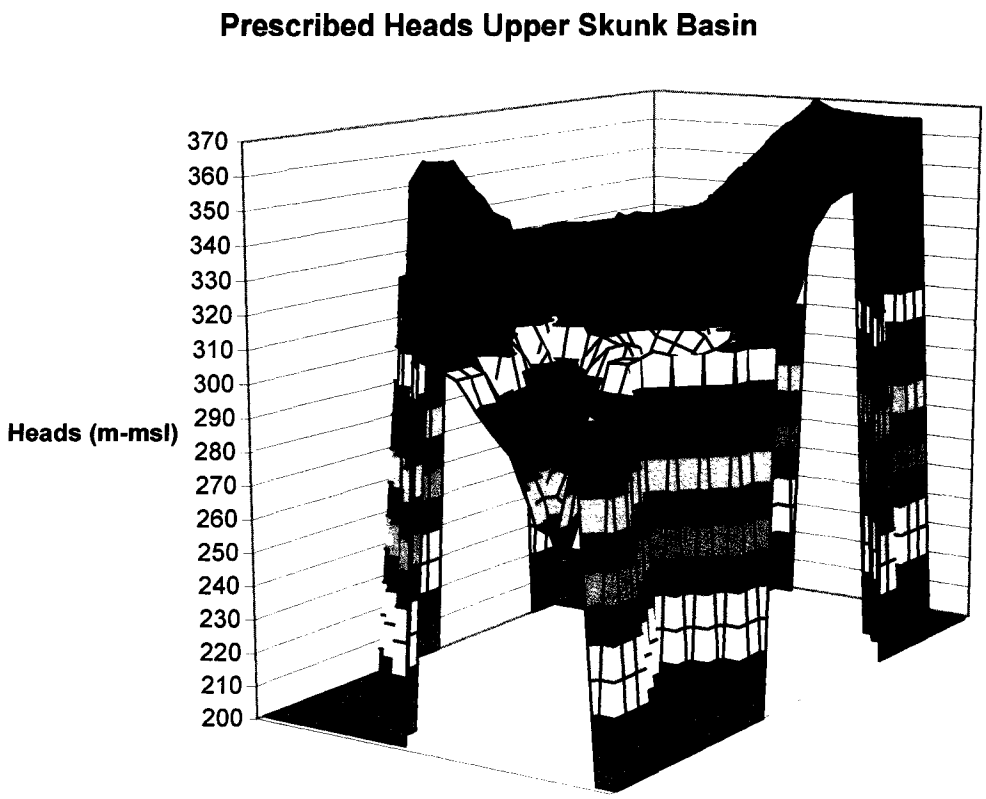


Figure 7.2 Elevations of the prescribed heads in the Upper Skunk in Iowa

Figure 7.2 shows the elevations of the prescribed heads within the Upper Skunk river basin.

The Iowa DNR hypsography bedrock topology was intersected with the domain shapefile. The new shapefile was imported into ArgusONE as a data layer for the bottom elevation of the unoxidized late Wisconsinan till. Figure 7.3 shows the bedrock contours.

The Upper Skunk hypsography elevation contour layer was imported into a data layer for the top elevation of the oxidized late Wisconsinan till. Figure 7.4 shows the image of the Upper Skunk surface contours in Iowa.

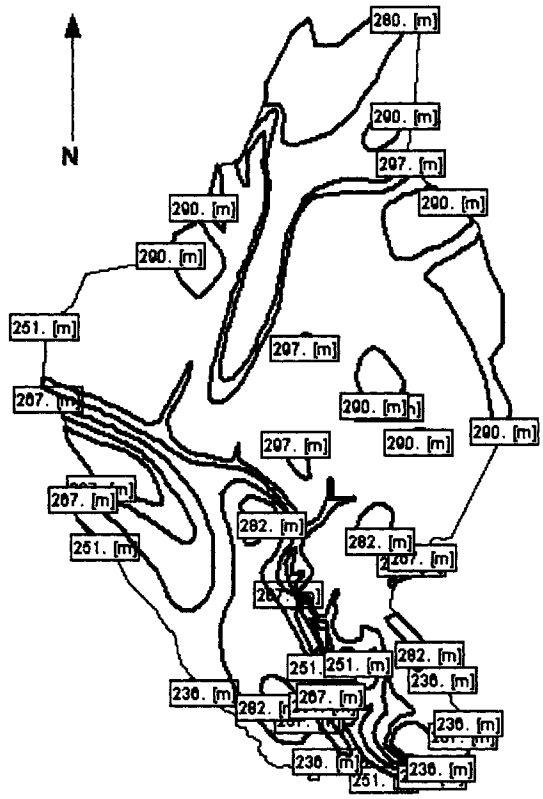


Figure 7.3 – Upper Skunk DNR bedrock contour topology (m-msl) in Iowa

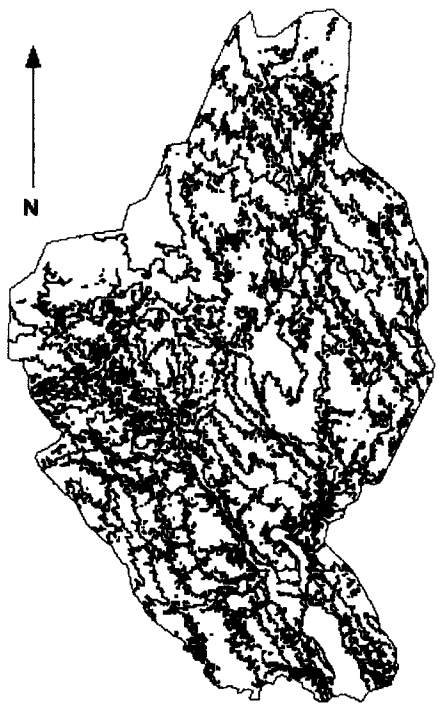


Figure 7.4 – Upper Skunk DNR surface contour topology image in Iowa

7.2 UPPER SKUNK GROUNDWATER MODEL DATA

7.2.1 Time Frame

An eight year period identical to the Walnut Creek model duration was used in the forward groundwater model of the Upper Skunk basin. Figure 5.5 shows the time in seconds for each of the recharge periods.

7.2.2 Thickness of the Geologic Layers

The Upper Skunk basin and the Walnut Creek watershed both lie squarely in the middle of the Des Moines Lobe. Chapter 4 describes this geology. For this reason it is reasonable to assume similar geological properties between the two areas.

The geologic information as discussed in Chapter 4 indicates that the late Wisconsinan oxidized layer varies in thickness from one to eight meters. Most of the one to three meters is immediately surrounding the rivers. Away from the rivers the late Wisconsinan oxidized layer varies from four to nine meters. A seven meter top thickness will be used to compare the two different sets of hydraulic conductivities.

Given the surface and bedrock hypsography and the top thickness, the model automatically computes the thickness of the bottom unoxidized layer by subtracting the thickness from the surface elevation. In this way each layer thickness is computed for each grid cell.

7.2.3 Recharge

An initial period of recharge was set to the average of recharge $3x10^{-5}$ mm/s over the eight-year period. This initial recharge had one purpose, that is, to initialize the model by filling the cells with moisture. Recharge was two-tenths the rainfall rates for the 12 linear periods. Figures 5.7 and 5.5 show the ArgusONE Modflow Graphical User Interface Plug-In Extension (USGS mfgui40s) screens for setting the time periods and recharge rates respectively.

7.2.4 Prescribed heads

The horizontal accuracy of the hypsography data from Iowa DNR is 52 meters. When the ArgusONE first rendered the data there were points at which the river data was below the surface. Each of the offending points had to be adjusted and were set to a meter above the surface. Figures 7.1 and 7.2 show a 2D and 3D representation of the prescribed heads. Section 7.1.2 describes how the layer was created.

7.2.5 Contour Data

GIS data from Iowa Department of Natural Resources was used as the basis for the surface and the bedrock contours. Surface elevation contours are divided into individual county files. The counties of Boone (08), Story (85), and Hamilton (40) were merged together into one shape. Bedrock topology was available from one file "brtopo" for the entire state. See the discussion in Section 7.1.2 for how the surface and bedrock elevation layers were created.

7.3 UPPER SKUNK GROUNDWATER MODELING RESULTS

7.3.1 Groundwater Map

The inverse modeled hydraulic conductivity values from both MODFLOW and LAD-EM modified MODFLOW were used to create groundwater maps of the Upper Skunk basin. (Figures 7.5 and 7.6) At the upper end of the basin the two different inverse groundwater model parameters have a distinct difference on the resulting heads. The MODFLOW hydraulic conductivities predict an elevation of 360 meters (msl) whereas the LAD-EM hydraulic conductivities predict a head elevation of 400 meters (msl).

The Walnut Creek watershed is shown in black outline in the lower western portion of the Upper Skunk river basin. Within the Walnut Creek watershed area, the groundwater elevations extend from 260 to 310 meters (msl) using the MODFLOW calculated hydraulic conductivities (Chapter 5). Using the LAD-EM calculated hydraulic conductivities (Chapter 6), the groundwater heads extend from 260 to 330 meters (msl).

7.3.2 Groundwater Flow

The upper section of the Upper Skunk river basin drains southward. The lower section drains into the center; that is, the lower western section drains eastward and the lower eastern section drains westward. Walnut Creek is in the lower western section and groundwater flow is from west to east.

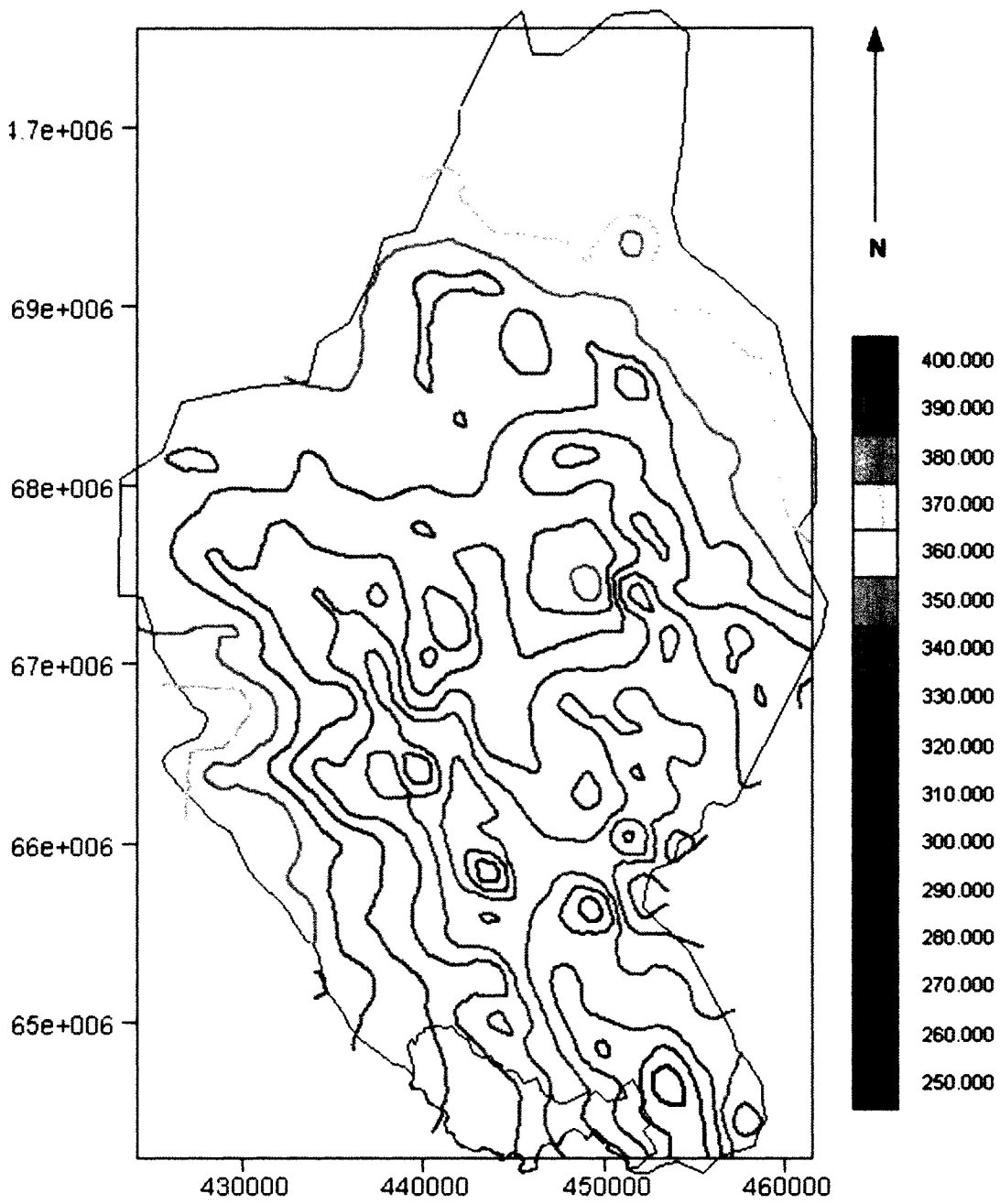


Figure 7.5 Modflow Top Thickness = 7 m Top K = 7.8E-9 Bottom K = 1.7E-4

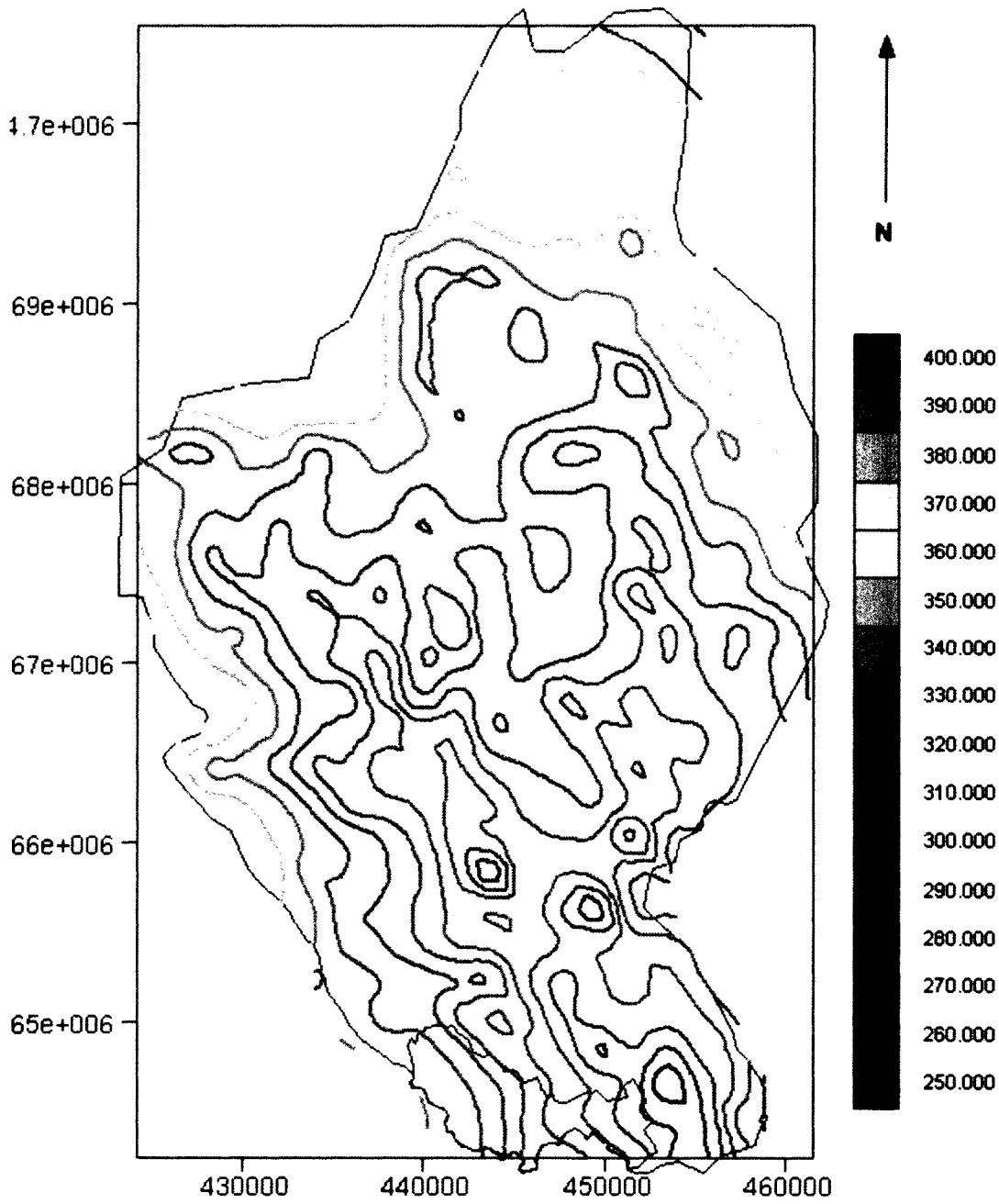


Figure 7.6 LAD-EM Top Thickness = 7 m Top K = 6.84E-04 Bottom K = 1E-10

7.4 SUMMARY OF THE UPPER SKUNK GROUNDWATER MODELING

Using the timing and recharge rates from Walnut Creek as input to the Upper Skunk groundwater model was required in this study in order to compare results with the Walnut Creek groundwater model. An alternate recommended procedure for basin analysis is to analyze all the raingages in the basin, analyze the rainfall for similar linear variations, and apply these as areal distributions throughout the basin.

The groundwater model cells needed to be filled with moisture in a steady state period in order for the 12 linear variations in recharge to produce a groundwater map of the basin. The steady state time period of 563 days was a rainfall impulse used to fill the cells of the Upper Skunk with moisture. Since the basin is 30 times the size of the watershed it is not expected that the well responses to rainfall impulses would have been the same. However, this is not an inverse model of the basin, rather a forward model, and any significant length of time, such as 365 days, could have been used to initialize the Upper Skunk basin groundwater model cells.

Accuracy of the NAD27 GIS data used in the basin model is not as good as the heavily scrutinized GIS data used in the Walnut Creek model. The USDA ARS National Soil Tilth Laboratory used NAD83 data. The Iowa Department of Natural Resources uses NAD27 data with a lesser degree of accuracy. The horizontal accuracy is important to groundwater modeling in order to place the rivers at the exact location of the prescribed heads. The grid discretization was 1965 meters and the GIS accuracy is 52 meters at its worst (15 meters at its best). The total effect is a 2.6% accuracy (0.8% at its best) for the grid size. The vertical accuracy is important in order to define the elevation of the prescribed heads. The total relief of the basin

is 115 meters. Vertical errors of 10 meters in the GIS data can cause serious modeling errors. The ArgusONE interface to MODFLOW will catch these errors and force the modeler to edit the data by hand for the questionable data.

The heads produced by using hydraulic conductivities computed from the LAD-EM groundwater model are more consistent with the observed data. The system was modeled as an unconfined system and hydraulic conductivities were within two meters of heads specified in the literature.

The west to east drainage is consistent with the Walnut Creek groundwater model.

CHAPTER 8
SUMMARY

8.0 INTRODUCTION

This study resulted in three products: 1) the creation of an inverse groundwater model within the Walnut Creek watershed in Iowa; 2) an adaptation of the USGS MODFLOW 2000 groundwater model to include an alternative inverse procedure using the Least sum Absolute Deviation regression and Expectation Maximization (LAD-EM) methodologies; and 3) creation of a regional groundwater model for the Upper Skunk River basin in upper central Iowa based on the results from the inverse modeling of Walnut Creek.

8.1 ANALYSIS

There were two observations regarding the input data for the Walnut Creek inverse model. One, it was observed that there were 12 different linear variations of rainfall accumulation during an eight year period of study; and two, it was observed that there was a direct connection between these 12 linear periods and the 12 well head elevations. These two observations made it possible to determine the periods of time and the recharge rate for the model execution.

An in-depth Monte Carlo analysis would have been required to determine the optimal periods of recharge had it not been for the realization of the linear periods of rainfall and the direct response of the wells to the rainfall.

While doubling the recharge essentially doubled the resulting hydraulic conductivity, the values produced were still in the same range. The MODFLOW model predicted a confined system and the Least sum Absolute Deviation Expectation Maximization (LAD-EM) MODFLOW model predicted an unconfined system.

NAD83 data is superior in accuracy to NAD27 geographic information system data. Both horizontal positional accuracy and the vertical accuracy of the hypsography data are essential to larger basin modeling. The ArgusONE interface is a good tool for resolving discrepancies.

Analysis of rainfall and well data make it possible to architect the time frame and recharge rates for groundwater models. Geographic information system data made it easy to define the geology for groundwater models but accuracy is an issue for calculating the elevations of the prescribed heads

8.2 RESULTS

An inverse groundwater model of the Walnut Creek watershed in Iowa was created. Geographic Information System data was used as the basis to define the geology. Analysis of the rainfall and well data made it possible to architect the groundwater model time frame and recharge rates. Observed well data were identified by its geospatial coordinates and entered into the MODFLOW model using the ArgusONE import tool. The architecture, the geology, and the observed heads made it possible to successfully create the inverse groundwater model. The

MODFLOW model was not successful at calculating hydraulic conductivities for a multi-layer geologic system.

Adding the Least sum Absolute Deviation and Expectation Maximization procedures to the inverse parameter estimation FORTRAN code created a new version of the MODFLOW computer model. This improved model was less sensitive to outliers in the data. The model calculated hydraulic conductivities that were nearly identical to published values.

Modeling at the basin level required large areas of hypsography and hydrography data in order to calculate the elevation of the prescribed heads. A program was written to separate the entire river and contour polylines of the geographic information system data, intersect them, and identify the elevation of the river at the point of intersection.

A groundwater model of the Upper Skunk river basin in Iowa was created. The Upper Skunk River basin groundwater model is a reasonable estimate of basin recharge and groundwater flow. The regional approach to groundwater modeling produced a head flow map oriented in the same direction as Walnut Creek. The heads produced by the basin groundwater model using the LAD-EM model hydraulic conductivities were 20 meters higher in the upland Walnut Creek area than those using the unmodified MODFLOW model hydraulic conductivites. The LAD-EM model hydraulic conductivity parameters produced results that were within two meters to those documented by USGS (Buchmiller, 1995).

8.3 CONCLUSION

Inverse modeling of a multi-layer geologic groundwater system with the least squares Gaussian MODFLOW procedure resulted in prediction of hydraulic conductivity values that indicated a confining layer on top. Further, convergence to a solution required starting values that were relatively close to the final convergence values. Hand manipulation of starting values was required.

Inverse modeling of a multi-layer geologic groundwater system using the LAD-EM modified MODFLOW model resulted in an unconfined aquifer system where final hydraulic conductivities values were nearly identical to those published in the literature. Further, initial values did not have to be close to final values for convergence to occur.

The Expectation Maximization procedure can be used with the Least sum Absolute Deviation regression method to predict the forward parameters. The Least sum Absolute Deviation method is a fast and accurate way to create inverse groundwater models. The Least sum Absolute Deviation does not require solutions close to the ultimate converged solution.

The MODFLOW model is now a much improved model to perform inverse groundwater modeling for two reasons: one, it can take any initial estimate of the hydraulic conductivity; and two, it works with multi-layered systems.

It is this author's recommendation that the US Geological Survey inspect this new improved MODFLOW source code and evaluate it on other multi-layer groundwater systems.

CHAPTER 9

REFERENCES

1. Andrews, W.F. and R.O. Dideriksen, US Department of Agriculture Soil Conservation Service, Soil Survey of Boone County, Iowa. 1981.

2. Barrodale, I. and F.D.K. Roberts. Algorithm 552: Solution of the Constrained L1 Linear Approximation Problem, ACM Transactions on Mathematical Software, 6, 1980. pp 231-235

3. Barrodale, I. and F.D.K. Roberts, An Efficient Algorithm for Discrete L2 Linear Approximation with Linear Constraints. SIAM J. Numer. Anal., v. 15, no. 3, 1978. pp. 603-611

4. Barrodale, I. and F.D.K. Roberts. An improved algorithm for discrete L1 linear approximation. SIAM J. Numer. Anal. 10(5), 1973. pp 839-848

5. Buchmiller, R.C., USGS 95-4109 I19.42/4:95-4109/DOC, "Ground-Water Levels and Flow at Selected Study Sites in the Walnut Creek Management System Evaluation Area, Boone and Story Counties, Iowa, 1991-1993", 1995.

6. Cade, B. S. and J.D. Richards, Permutation Tests for Least Absolute Deviation Regression, Biometrics, V. 52, I.3, Sept, 1996. pp 886-902

7. Cooley, R.L. and R.L. Naff, Regression Modeling of Ground-Water Flow, U.S. Geological Survey Techniques in Water Resources Investigations Chapter B4 Book 3, 1990. 232 p.

8. Dennis, J.E., D.M. Gay, and R.E.Welsch, An adaptive nonlinear least-squares algorithm, ACM Transactions of Mathematical Software, V. 7, No. 3, 1981. pp. 348-368

9. Dewitt, T.A., US Department of Agriculture Soil Conservation Service, Soil Survey of Story County, Iowa. 1984.

10. Dougherty, E.L. and S.T. Smith, The use of linear programming to filter digitized map data, Geophysics, V.31, 1966. pp. 253-259

11. Eppstein, M.J. and D.E. Dougherty, Simultaneous estimation of transmissivity values and zonation, WRR, Vol. 32, No. 11, 1996. pp. 3321-3336

12. Foss, T, I. Myrtveit, and E. Stensrud, A Comparison of LAD and OLS Regression for Effort Prediction of Software Projects, ESCOM Conference Ltd, August 21, 2001

13. Harbaugh, A.W., A computer program for calculating subregional water budgets using the results from the U.S. Geological Survey modular three-dimensional finite-difference ground-water flow model: U.S. Geological Survey Open-File Report 90-392, U.S. Geological Survey, 1990. 46 p.

14. Harbaugh, A.W., E.R. Banta, M.C. Hill, and M.G. McDonald, MODFLOW-2000, THE U.S. GEOLOGICAL SURVEY MODULAR GROUND-WATER MODEL-USER GUIDE TO MODULARIZATION CONCEPTS AND THE GROUND-WATER FLOW PROCESS, USGS, Reston, VA, 2000.

15. Hill, M.C., Preconditioned conjugate-gradient 2 (PCG2), a computer program for solving groundwater flow equation, U.S. Geological Survey Open File Report 90-4048, 1990. 43 p.

16. Hill, M.C., A Computer Program (MODFLOWP) for Estimating Parameters of a Transient, Three-Dimensional, Ground-Water Flow Model Using Nonlinear Regression, U.S. Geological Survey Open-File Report 91-484, 1992. 358 p.

17. Hill, M.C., Methods and guidelines for effective model calibration. U.S. Geological Survey Water-Resources investigations report 98-4005, 1998. 90 p.

18. Hill, M.C., E.R Banta, A.W. Harbaugh, and E.R. Anderman, Documentation of MODFLOW-2000, the U.S. Geological Survey modular ground-water model, User's guide to the Observation, Sensitivity, and Parameter-Estimation Process and three post-processing programs: U.S. Geological Survey Open-File Report 00-184, 2000. 209 p.

19. Huber, P.J., Robust regression—Asymtotics, conjectures, and Monte Carlo: Annals of Statistics, v.1, 1973. pp. 799–821

20. Huddleston, J.M. and B.A. Shafer, Streamflow Analysis of the Colorado River, AGU Infiltration Conference, Utah, 1987.

21. Keidser, A. and D. Rosbjerg, A comparison of four inverse approaches to groundwater flow and transport parameter identification, Water Resources Research, 27(9), September 1991. pp. 2219-2232

22. McDonald, M.G., and A.W. Harbaugh, A modular three-dimensional finite-difference ground-water flow model, Techniques of Water Resources Investigations 06-A1, United States Geological Survey, 1988.

23. McLaughlin, D. B. and L.R. Townley, A reassessment of the groundwater inverse problem, Water Resources Research, 32(4), 1996. pp. 1131-1161

24. McLaughlin, D. B. and E.F. Wood, A distributed parameter approach for evaluating the accuracy of groundwater model predictions, 1, Theory, Water Resources Research, 24(7), 1988a. pp. 1037-1047

25. McLaughlin, D. B. and E.F. Wood, A distributed parameter approach for evaluating the accuracy of groundwater model predictions, 2, Applications to Groundwater Flow, Water Resources Research, 24(7), 1988b. pp. 1048-1060

26. Mielke, P.W. and K. J. Berry, Permutation Methods A Distance Function Approach, Springer-Verlag, New York, 2001.

27. Phillips, R. F., Least absolute deviations estimation via the EM algorithm, Statistics and Computing, No. 12, 2002. pp. 281-285

28. Poeter, E. P. and M.C. Hill, Inverse Methods: A Necessary Next Step in Groundwater Modeling, Ground Water, v. 35, no. 2, 1997. pp. 250-260

29. Poeter, E.P. and M.C. Hill, Unrealistic Parameter Estimates in Inverse Modeling: A Problem or Benefit for Model Calibration, In Kovar, K., ed., 1996, proceedings of the IAHS International Conference on Calibration and Reliability in Groundwater Modeling (ModelCARE'96), Golden, Co., 1996.

30. Poeter, E.P. and S.A. McKenna, Reducing Uncertainty Associated With Ground-Water Flow and Transport Predictions, Ground Water, 33(6), 1995. pp. 899-904

31. Rao, C. R., Methodology based on the L_1-norm, in statistical inference, Sankhya, A 50, 1988. pp 289-313

32. Sauer, P.A. and J.L. Hatfield, Walnut Creek Watershed Research Protocol Report, 94-1, 1994.

33. Simpkins, W.W. coordinator., Water, Water, Everywhere..., Iowa State University, and Geological Society of Iowa, 57th Annual Tri-state Geological Field Conference, Guidebook Series No. 58, 1993. 139 p.

34. Simpkins, W.W. and M.R. Burkart, Hydrogeology and Water Quality of the Walnut Creek Watershed, Iowa Geological Survey Bureau Guidebook Series No. 20, 1996. 105 p.

35. Sheynin, O.B., R. J. Boscovich's work on probability, Archive for History of Exact Sciences, 9: 1973. pp. 306-324

36. Sun, N.Z., Inverse problems in groundwater modeling, Volume 6 of Theory and Applications of Transport in Porous Media, Kluwer Academic Publishers, 1994. 337p.

37. Sun, N.Z. and W.G. Yeh, A Stochastic Inverse Solution for Transient Groundwater Flow: Parameter Identification and Reliability Analysis, Water Resources Research, 28(12), 1992. p. 3269

38. Sun, N.Z. and W.G. Yeh, Coupled Inverse Problems in Groundwater Modeling, 1. Sensitivity Analysis and Parameter Identification, Water Resources., Vol. 26, No. 10, 1990. pp. 2507-2525

39. Tasker, G.D. and G.E. Granato, Statistical Approaches to Interpretation of Local, Regional, and National Highway-Runoff and Urban-Stormwater Data, USGS Open-File Report 00-491, 2000.

40. Thompson, C.A., Groundwater Resources, Boone County, Open File Report 82-8 WRD, Iowa Geological Survey, 1982.

41. Thompson, C.A., Groundwater Resources, Story County, Open File Report 82-85 WRD, Iowa Geological Survey, 1982.

42. Todd, D. K., Ground-Water Hydrology, Wiley, New York, 1959.

10.1 - Listing of Well Station UTM X,Y Location and Elevations in meters

Name [String]	X [Real]	Y [Real]	Height [Real]	Depth [Real]	Top_Elev [Real]	Bottom_Elev [Real]
WC0106	446295.69	4645878	3.10	7.50	308.0918	305.8058
WC0109	446296.69	4645879	2.60	10.00	307.9913	304.9433
WC0114	446298.91	4645879	2.90	15.00	308.0736	303.5016
WC0204	445911.47	4645914	2.00	5.00	305.1901	303.6661
WC0206	445911.97	4645913	1.80	7.50	305.1505	302.8645
WC0209	445912.75	4645912	1.90	10.00	305.1627	302.1147
WC0214	445913.25	4645911	1.80	15.00	305.0438	300.4718
WC0221	445914.03	4645910	2.70	22.50	305.1139	298.2559
WC0304	445976.41	4645879	1.65	5.00	305.7418	304.2178
WC0306	445977.88	4645879	3.10	7.00	305.8820	303.7484
WC0311	445978.72	4645879	2.60	12.50	305.6900	301.8800
WC0409	442580.16	4645837	2.75	10.50	316.9737	313.7733
WC0416	442580.44	4645836	2.65	17.50	316.9890	311.6550
WC0506	441796.34	4645930	2.65	7.50	317.0499	314.7639
WC0514	441795.06	4645930	2.55	15.00	316.9950	312.4230
WC0609	443458.03	4645712	2.70	10.50	313.8282	310.6278
WC0619	443458.28	4645711	2.70	20.50	313.8800	307.6316
WC0706	442153.38	4646219	2.25	7.50	316.5988	314.3128
WC0709	442153.53	4646220	2.15	10.50	316.6140	313.4136
WC0714	442153.63	4646221	2.25	15.50	316.6262	311.9018
WC0721	442153.59	4646222	2.35	22.50	316.6933	309.8353
WC0814	442111.44	4645108	2.30	15.50	318.1533	313.4289
WC0819	442111.44	4645108	2.30	20.50	318.1563	311.9079
WC0904	442154.41	4646450	2.25	5.50	316.0685	314.3921
WC0907	442154.44	4646449	2.25	8.50	316.1873	313.5965
WC0913	442154.50	4646448	3.50	14.50	316.5287	312.1091
WC1005	441765.75	4646442	2.10	6.00	316.4647	314.6359
WC1010	441765.84	4646443	1.95	11.00	316.4007	313.0479
WC1017	441765.91	4646444	2.25	18.50	316.5104	310.8716
WC1106	441765.84	4646183	2.90	7.50	317.5284	315.2424
WC1110	441765.78	4646185	1.40	11.50	317.0652	313.5600
WC1115	441765.84	4646186	1.60	16.50	317.1353	312.1061
WC1210	442084.81	4645516	1.65	11.50	315.8764	312.3712
WC1216	442084.88	4645516	3.00	17.50	316.2544	310.9204
WC1306	443802.91	4645087	2.70	7.50	316.0410	313.7550
WC1318	443805.44	4645087	3.30	19.50	316.1294	310.1858
WC1324	443804.25	4645087	2.25	25.50	315.8460	308.0736
WC1406	443272.72	4645094	2.60	7.50	315.9953	313.7093
WC1411	443271.25	4645094	2.90	12.50	316.0471	312.2371
WC1416	443269.63	4645094	2.70	17.50	316.0563	310.7223
WC1505	445439.69	4645976	4.15	6.50	306.9214	304.9402
WC1509	445440.56	4645976	2.45	10.50	306.3392	303.1388
WC1516	445441.50	4645976	3.50	17.00	306.7477	301.5661
WC1606	440569.63	4645791	3.10	7.50	322.1919	319.9059
WC1616	440569.59	4645790	2.95	17.50	322.0669	316.7329
WC1704	444786.25	4646226	3.00	5.50	307.7627	306.0863
WC1706	444786.63	4646226	3.00	7.50	307.7078	305.4218
WC1709	444787.47	4646225	3.20	10.00	307.7809	304.7329

10.1 - Listing of Well Station UTM X,Y Location and Elevations in meters

Name [String]	X [Real]	Y [Real]	Height [Real]	Depth [Real]	Top_Elev [Real]	Bottom_Elev [Real]
WC1714	444788.00	4646224	3.10	15.00	307.7108	303.1388
WC1809	442633.03	4645109	2.93	10.00	316.0471	312.9991
WC1904	445090.59	4646039	2.85	5.00	306.8848	305.3608
WC1906	445088.59	4646039	3.10	7.50	306.9184	304.6324
WC1909	445087.25	4646039	2.90	10.00	306.8909	303.8429
WC1914	445086.44	4646039	3.33	12.50	307.0098	303.1998
WC2009	440955.75	4645122	2.90	10.00	321.6250	318.5770
WC2016	440956.66	4645122	3.15	17.00	321.6707	316.4891
WC2106	440711.19	4645124	3.24	7.50	319.8906	317.6046
WC2114	440713.09	4645124	2.90	15.50	319.7962	315.0718
WC2206	440960.88	4645859	2.91	7.50	318.8360	316.5500
WC2214	440960.69	4645860	2.72	15.00	318.8056	314.2336
WC2309	441361.16	4645121	2.68	10.00	319.2140	316.1660
WC2316	441361.06	4645120	2.90	17.50	319.3085	313.9745
WC2409	441363.66	4645520	2.57	10.50	319.5218	316.3214
WC2416	441362.53	4645520	3.60	17.50	320.0004	314.6664
WC2507	442636.22	4645314	3.11	8.00	315.4040	312.9656
WC2603	446283.94	4646050	1.40	3.50	307.7748	306.7080
WC2606	446283.25	4646049	1.31	6.50	307.7596	305.7784
WC2616	446282.59	4646050	1.17	16.50	307.7139	302.6847
WC2715	449767.50	4643757	3.38	16.50	296.5582	291.5290
WC2720	449768.38	4643758	3.27	20.50	296.3509	290.1025
WC2730	449769.25	4643758	3.40	31.50	296.5033	286.9021
WC2807	449722.38	4643616	1.71	8.50	299.2344	296.6436
WC2811	449723.25	4643616	2.96	12.50	299.6123	295.8023
WC2818	449724.09	4643616	1.02	19.00	299.0210	293.2298
WC2903	444891.03	4646280	1.10	4.50	309.0123	307.6407
WC2909	444890.19	4646279	0.75	10.50	308.8843	305.6839
WC2919	444889.59	4646279	0.89	19.00	308.8386	303.0474
WC3004	445107.94	4646277	0.94	4.50	310.6613	309.2897
WC3008	445107.09	4646278	0.63	9.00	310.6613	307.9181
WC3015	445106.44	4646278	0.96	16.50	310.7558	305.7266
WC3106	445425.63	4646274	0.70	6.50	310.2620	308.2808
WC3114	445424.00	4646274	0.50	14.00	310.2011	305.9339
WC3204	444791.75	4646197	2.85	5.50	307.8358	306.1594
WC3210	444792.34	4646196	2.60	10.50	307.7596	304.5592
WC3216	444792.94	4646195	1.56	16.50	307.4396	302.4104
WC3306	444797.31	4645889	1.19	6.50	307.5188	305.5376
WC3316	444798.25	4645889	1.63	16.50	307.5950	302.5658
WC3405	445288.72	4645976	2.29	5.50	306.7812	305.1048
WC3416	445286.28	4645976	2.84	16.50	306.9306	301.9014
WC3430	445287.88	4645977	3.29	30.50	307.0769	297.7805
WC3503	445277.50	4645884	1.71	3.50	306.5465	305.4797
WC3506	445276.66	4645884	1.83	6.00	306.5800	304.7512
WC3518	445275.81	4645884	1.44	17.50	306.4093	301.0753
WC3608	441673.19	4647533	2.13	8.50	318.8056	316.2148
WC3613	441672.13	4647533	2.19	13.50	318.7507	314.6359
WC3624	441671.06	4647533	2.45	24.50	318.8909	311.4233
WC3703	441764.00	4648330	2.00	3.50	314.7395	313.6727

10.1 - Listing of Well Station UTM X,Y Location and Elevations in meters

Name [String]	X [Real]	Y [Real]	Height [Real]	Depth [Real]	Top_Elev [Real]	Bottom_Elev [Real]
WC3706	441763.97	4648331	1.00	6.50	314.3860	312.4048
WC3717	441764.06	4648332	3.10	17.50	315.0413	309.7073
WC3803	441759.91	4649015	2.00	3.50	313.9775	312.9107
WC3806	441760.34	4649015	1.04	6.50	313.8099	311.8287
WC3817	441760.22	4649016	2.10	17.50	314.3646	309.0306

10.2 – Listing of the Observed Heads on Walnut Creek in meters (msl)

Number	UTM X	UTM Y	1st Obs	2nd Obs.	3rd Obs.	4th Obs.	5th Obs.
1	446295.69	4645878	308.09184	306.75682	304.99507	306.4002	305.26939
2	446296.69	4645878.5	307.99126	306.76596	304.92192	306.4002	305.27549
3	446298.91	4645878.5	308.07355	306.66842	304.94326	306.37886	305.29073
4	445911.47	4645914	305.19014	303.77587	305.19014	303.18456	302.92548
5	445911.97	4645913	305.15052	303.76368	302.65421	303.19066	302.73955
6	445912.75	4645912	305.16271	303.77892	302.67859	303.25771	302.75784
7	445913.25	4645911	305.04384	302.85233	302.42256	303.06569	302.86757
8	445914.03	4645909.5	305.11394	300.85894	301.7459	302.94072	302.96815
9	445976.41	4645879	305.74183	305.04079	303.8917	304.24222	303.85207
10	445977.88	4645879	305.88204	304.71161	303.54422	304.27574	304.48606
11	445978.72	4645879	305.69002	304.7177	303.48631	304.2666	303.8856
12	442580.16	4645836.5	316.97371	315.5503	314.14517	314.50178	313.35574
13	442580.44	4645836	316.98895	315.56554	314.16346	314.49569	313.38926
14	441796.34	4645929.5	314.36158	315.37961	314.88278	315.45276	314.38291
15	441795.06	4645929.5	316.99505	315.34608	314.91326	315.4619	316.99505
16	443458.03	4645712	313.82818	312.43219	311.5757	311.42635	311.18556
17	443458.28	4645711	313.87999	312.51144	311.77687	311.7342	311.57266
18	442153.38	4646219	316.59881	314.83402	313.66663	313.98972	313.67578
19	442153.53	4646219.5	316.61405	314.80658	313.6453	313.97143	313.65139
20	442153.63	4646220.5	316.62624	314.80658	313.65139	313.97448	316.62624
21	442153.59	4646221.5	316.6933	314.80658	313.66358	313.98362	316.6933
22	442111.44	4645107.5	318.15329	316.94628	315.28817	315.52286	313.96838
23	442111.44	4645108	318.15634	316.85179	315.17539	316.23	313.92571
24	442154.41	4646450	316.06846	314.99556	316.06846	314.20613	314.10554
25	442154.44	4646449	316.18733	315.07786	313.69406	314.21527	313.56605
26	442154.5	4646448	316.5287	315.02604	313.62396	314.23356	313.5569
27	441765.75	4646441.5	316.4647	315.04433	316.4647	314.2427	314.0964
28	441765.84	4646443	316.40069	315.02604	313.98667	314.24575	313.57824
29	441765.91	4646444	316.51042	314.95289	314.05678	314.29757	313.68797
30	441765.84	4646182.5	317.52845	316.3824	315.09919	315.48629	314.77306
31	441765.78	4646184.5	317.06515	316.36106	315.10224	315.48019	314.77001
32	441765.84	4646186	317.13526	316.37935	315.11443	315.47714	314.7761
33	442084.81	4645516	315.87643	315.02299	313.65444	314.28842	313.45937
34	442084.88	4645515.5	316.25438	314.99556	313.68492	314.55055	313.62701
35	443802.91	4645087	316.04102	316.04102	316.04102	313.13933	316.04102
36	443805.44	4645087	316.12942	314.28233	312.61812	313.19724	311.84698
37	443804.25	4645087	315.84595	313.60262	312.12739	313.15762	312.28894
38	443272.72	4645094	315.9953	315.9953	312.99302	313.54776	315.9953
39	443271.25	4645094	316.04712	314.1726	312.99302	313.52947	316.04712
40	443269.63	4645094	316.05626	314.11469	312.99302	313.52033	312.67298
41	445439.69	4645976	306.92141	304.75733	306.92141	304.18735	306.92141
42	445440.56	4645975.5	306.33924	304.45862	303.54727	304.17516	303.94656
43	445441.5	4645975.5	306.74767	304.4891	303.5808	304.17821	303.94656
44	440569.63	4645791	322.19189	322.19189	322.19189	322.19189	322.19189
45	440569.59	4645790	322.06692	322.06692	318.25387	319.06464	318.25692
46	444786.25	4646226	307.76266	307.76266	307.76266	307.76266	307.76266
47	444786.63	4646225.5	307.70779	307.70779	304.7177	305.16576	304.62322
48	444787.47	4646224.5	307.78094	307.78094	304.68113	305.15966	304.56835
49	444788	4646224	307.71084	307.71084	304.69332	305.14138	304.5653
50	442633.03	4645108.5	316.04712	316.04712	312.92597	313.48375	312.50839
51	445090.59	4646038.5	306.88483	306.88483	306.88483	306.88483	306.88483
52	445088.59	4646039	306.91836	306.91836	304.08372	304.22393	303.86426
53	445087.25	4646039	306.89093	306.89093	304.08372	304.23307	303.85207

10.2 – Listing of the Observed Heads on Walnut Creek in meters (msl)

Number	UTM X	UTM Y	1st Obs	2nd Obs.	3rd Obs.	4th Obs.	5th Obs.
54	445086.44	4646039	307.0098	307.0098	304.01971	304.23612	303.84293
55	440955.75	4645121.5	321.62496	321.62496	317.88202	318.42151	321.62496
56	440956.66	4645122	321.67068	321.67068	317.84544	318.41542	321.67068
57	440711.19	4645124	319.89065	319.89065	317.43091	317.6077	316.80912
58	440713.09	4645124	319.79616	319.79616	317.4553	317.59855	316.83046
59	440960.88	4645859	318.83604	318.83604	316.48603	316.85179	316.29096
60	440960.69	4645860	318.80556	318.80556	316.51956	316.87618	316.33363
61	441361.16	4645120.5	319.21399	319.21399	317.04991	317.28766	316.67196
62	441361.06	4645120	319.30848	319.30848	316.94628	317.45834	319.30848
63	441363.66	4645519.5	319.52184	319.52184	316.33058	317.53759	316.17209
64	441362.53	4645519.5	320.00038	320.00038	316.44336	317.50102	316.26658
65	442636.22	4645314	315.40399	315.40399	312.5023	312.59983	315.40399
66	446283.94	4646049.5	307.77485	307.77485	307.77485	307.77485	307.77485
67	446283.25	4646049	307.75961	307.75961	307.75961	307.75961	307.75961
68	446282.59	4646049.5	307.71389	307.71389	307.71389	307.71389	307.71389
69	449767.5	4643756.5	296.55821	296.55821	296.55821	293.87597	293.42791
70	449768.38	4643757.5	296.35094	296.35094	296.35094	296.35094	296.35094
71	449769.25	4643757.5	296.50334	296.50334	296.50334	292.30625	291.58997
72	449722.38	4643615.5	299.23435	299.23435	299.23435	297.2623	296.46677
73	449723.25	4643615.5	299.6123	299.6123	299.6123	297.26534	296.42105
74	449724.09	4643615.5	299.02099	299.02099	299.02099	297.2623	296.44848
75	444891.03	4646279.5	309.01234	309.01234	309.01234	309.01234	309.01234
76	444890.19	4646279	308.88432	308.88432	308.88432	306.72024	306.12588
77	444889.59	4646279	308.8386	308.8386	308.8386	306.70195	306.17465
78	445107.94	4646277	310.6613	310.6613	310.6613	310.6613	309.14035
79	445107.09	4646277.5	310.6613	310.6613	310.6613	308.3875	308.17718
80	445106.44	4646277.5	310.75579	310.75579	310.75579	308.40883	308.22595
81	445425.63	4646274	310.26202	310.26202	310.26202	308.45455	310.26202
82	445424	4646274	310.20106	310.20106	310.20106	308.44846	307.2384
83	444791.75	4646197	307.83581	307.83581	307.83581	305.55895	305.57724
84	444792.34	4646195.5	307.75961	307.75961	307.75961	304.78781	304.46167
85	444792.94	4646195	307.43957	307.43957	307.43957	304.78171	304.46472
86	444797.31	4645889	307.51882	307.51882	307.51882	305.59858	305.34254
87	444798.25	4645889	307.59502	307.59502	307.59502	305.71745	305.40046
88	445288.72	4645976	306.7812	306.7812	306.7812	306.7812	306.7812
89	445286.28	4645976	306.93055	306.93055	306.93055	303.98314	303.69662
90	445287.88	4645976.5	307.07686	307.07686	307.07686	304.03495	303.81245
91	445277.5	4645884	306.5465	306.5465	306.5465	306.5465	306.5465
92	445276.66	4645884	306.58003	306.58003	306.58003	304.82438	304.40376
93	445275.81	4645884	306.40934	306.40934	306.40934	304.8701	306.40934
94	441673.19	4647532.5	318.80556	318.80556	318.80556	318.80556	315.76061
95	441672.13	4647532.5	318.7507	318.7507	318.7507	318.7507	314.96508
96	441671.06	4647532.5	318.8909	318.8909	318.8909	318.8909	314.99861
97	441764	4648329.5	314.73953	314.73953	314.73953	314.73953	313.21858
98	441763.97	4648330.5	314.38596	314.38596	314.38596	314.38596	312.8711
99	441764.06	4648331.5	315.04128	315.04128	315.04128	315.04128	312.15178
100	441759.91	4649014.5	313.97753	313.97753	313.97753	313.97753	312.38647
101	441760.34	4649015	313.80989	313.80989	313.80989	313.80989	313.80989
102	441760.22	4649016	314.36462	314.36462	314.36462	314.36462	311.19775

10.2 – Listing of the Observed Heads on Walnut Creek in meters (msl)

Number	6th Obs.	7th Obs.	8th Obs.	9th Obs.	10th Obs.	11th Obs.	12th Obs.	13th Obs.
1	306.63185	305.10785	306.56784	305.65039	305.97043	305.16881	305.99177	305.35474
2	306.6349	304.82438	306.43068	305.65344	305.98262	305.15662	306.00396	305.34254
3	306.6288	304.83048	306.41239	305.66258	305.98567	305.11394	306.00396	305.34864
4	303.56556	302.92548	303.25162	303.02606	303.2821	302.90719	303.12055	302.91938
5	305.15052	302.23358	303.25466	303.03521	303.28514	302.40122	303.1297	302.87976
6	303.53813	302.26406	303.25771	303.03216	303.27905	302.27016	303.1236	302.88281
7	305.04384	302.55362	303.31562	302.89195	303.2821	302.33112	303.14798	303.0413
8	303.16018	302.80966	303.16018	302.77003	303.09617	302.62373	303.08398	303.21504
9	304.59883	303.5107	304.12944	303.60518	304.07458	304.01971	303.83378	303.49546
10	304.6415	303.1998	304.17516	303.65395	304.08982	304.04714	304.00447	303.39487
11	304.62626	303.23028	304.21478	303.71796	304.12944	304.06543	303.91608	303.44669
12	315.68136	313.35269	316.97371	314.58103	314.84621	313.21248	314.90107	314.0202
13	315.65088	313.39841	316.98895	314.64809	314.87059	313.0997	314.94374	314.01715
14	315.63259	317.04991	317.04991	314.92241	317.04991	314.1787	315.08395	314.64504
15	315.63564	316.99505	316.99505	314.96508	316.99505	314.19394	315.0809	314.63894
16	312.55411	311.35015	313.82818	311.71591	312.05424	311.79211	312.14568	311.62447
17	312.83758	311.44769	313.87999	311.85002	312.12434	311.68543	312.30113	311.77382
18	314.87974	316.59881	316.59881	314.00496	316.59881	313.55081	314.49569	313.64225
19	314.86754	316.61405	316.61405	314.0202	316.61405	313.06008	314.51702	313.65749
20	314.86145	316.62624	316.62624	314.0141	316.62624	313.05094	314.50178	313.65444
21	314.86754	316.6933	316.6933	314.05982	316.6933	313.08446	314.48959	313.59653
22	316.41593	318.15329	318.15329	315.83376	316.12332	312.75223	316.23914	314.9407
23	316.3763	318.15634	318.15634	315.83071	316.12027	312.58764	316.2361	314.93765
24	315.15406	316.06846	316.06846	313.99277	316.06846	313.81294	314.45302	313.80989
25	315.1571	316.18733	316.18733	313.98972	316.18733	313.2643	314.45302	313.55995
26	315.15406	316.5287	316.5287	313.99277	316.5287	313.23077	314.45606	313.52947
27	315.11443	316.4647	316.4647	314.20613	316.4647	316.4647	314.69076	316.4647
28	315.10529	316.40069	316.40069	314.20613	316.40069	313.49594	314.68466	316.40069
29	315.19368	316.51042	316.51042	314.34938	316.51042	313.4807	314.706	316.51042
30	316.3001	317.52845	317.52845	315.4619	317.52845	314.70905	315.93434	317.52845
31	316.28182	317.06515	317.06515	315.45886	317.06515	314.70905	315.92215	317.06515
32	316.29706	317.13526	317.13526	315.48019	317.13526	314.70905	315.90996	317.13526
33	315.09614	315.87643	315.87643	314.01106	314.34024	313.31611	313.95924	315.87643
34	315.27293	316.25438	316.25438	314.00801	314.33719	316.25438	314.80658	316.25438
35	314.46521	316.04102	313.96229	313.25515	313.35269	316.04102	313.77941	316.04102
36	314.02325	311.39587	313.52033	312.82234	312.91682	311.54522	313.34354	316.12942
37	314.49264	311.83174	313.97753	313.27649	313.36183	311.98109	313.79465	315.84595
38	314.38291	315.9953	314.16346	313.42889	313.76417	315.9953	313.88914	315.9953
39	314.3311	312.55716	314.13298	313.41974	313.73978	312.84672	313.8739	316.04712
40	314.27928	312.43219	314.08726	313.45022	313.72454	312.82843	313.84646	316.05626
41	304.22088	303.90084	304.33975	303.94351	304.26355	303.89474	304.95545	306.92141
42	304.4251	303.48631	304.35194	303.94046	304.2666	303.84902	304.18126	306.33924
43	304.45558	306.74767	306.74767	306.74767	306.74767	306.74767	306.74767	306.74767
44	319.69862	322.19189	322.19189	319.18656	319.66205	319.16827	319.79616	319.16522
45	320.05524	322.06692	322.06692	319.18046	319.50965	317.38824	319.68948	318.19291
46	305.61991	307.76266	307.76266	307.76266	307.76266	307.76266	307.76266	307.76266
47	305.59553	304.96459	307.70779	307.70779	305.29378	307.70779	305.44313	307.70779
48	305.19319	304.96764	307.78094	307.78094	305.28768	304.49215	305.44922	307.78094
49	305.5681	304.97069	307.71084	307.71084	305.2511	304.46167	305.47056	307.71084
50	314.37377	312.55411	316.04712	313.7154	313.99277	313.03265	314.30062	316.04712
51	304.58969	306.88483	306.88483	306.88483	304.49825	306.88483	306.88483	306.88483

10.2 – Listing of the Observed Heads on Walnut Creek in meters (msl)

Number	6th Obs.	7th Obs.	8th Obs.	9th Obs.	10th Obs.	11th Obs.	12th Obs.	13th Obs.
52	304.61712	303.92522	306.91836	304.33366	304.40071	303.93132	304.56835	306.91836
53	304.67198	303.85817	306.89093	304.22393	304.41595	303.90998	304.55311	306.89093
54	304.69942	303.85817	307.0098	307.0098	304.42205	303.84598	304.35804	307.0098
55	319.67729	321.62496	319.5005	318.48247	318.91834	321.62496	319.278	317.85763
56	319.67119	321.67068	319.47002	318.47028	318.91224	317.49797	319.25666	317.85154
57	318.23558	319.89065	318.18986	317.58026	317.79972	316.77864	317.9887	317.13221
58	318.25082	319.79616	318.20206	317.61074	317.79362	316.79388	318.00394	317.14745
59	317.62903	318.83604	318.83604	316.76645	317.05296	316.26658	316.9158	316.55614
60	318.80556	318.80556	318.80556	316.86094	317.08344	316.23	316.92799	316.59271
61	317.38824	319.21399	317.41262	317.31814	317.38824	316.70244	317.30594	317.17183
62	317.65342	319.30848	317.7601	317.34862	317.84239	316.77559	317.38519	317.2267
63	319.52184	319.52184	318.18682	317.3791	317.5955	315.70574	317.94602	316.39459
64	320.00038	320.00038	318.17158	317.42786	317.54064	315.74842	317.93078	316.36411
65	313.35878	315.40399	312.94121	312.50839	312.71261	315.40399	312.66689	315.40399
66	307.77485	306.25999	307.77485	306.25999	306.25694	307.77485	307.77485	307.77485
67	307.75961	305.4858	307.75961	305.68392	305.87899	307.75961	307.75961	307.75961
68	307.71389	304.85182	307.71389	305.72659	305.81803	307.71389	307.71389	307.71389
69	293.98874	293.08654	296.55821	293.92778	293.98265	292.49522	296.55821	293.50716
70	296.35094	292.83355	296.35094	293.6047	293.64737	292.16604	296.35094	293.20846
71	296.50334	292.87622	296.50334	288.34385	290.05378	289.95014	296.50334	292.24529
72	299.23435	299.23435	299.23435	297.41165	297.50614	296.1833	299.23435	299.23435
73	299.6123	296.6405	299.6123	298.13402	298.22242	299.6123	299.6123	299.6123
74	299.02099	295.89679	299.02099	297.38117	297.46042	299.02099	299.02099	299.02099
75	309.01234	309.01234	309.01234	309.01234	307.40299	309.01234	308.03393	309.01234
76	308.88432	308.88432	308.88432	307.21097	307.12562	308.88432	308.09489	308.88432
77	308.8386	306.12893	308.8386	307.25364	307.07381	308.8386	308.13146	308.8386
78	310.6613	310.6613	310.6613	310.6613	310.6613	310.6613	310.6613	310.6613
79	310.6613	310.6613	310.6613	309.28666	308.95442	310.6613	309.99684	310.6613
80	310.75579	308.03698	310.75579	309.38114	309.02148	310.75579	310.10352	310.75579
81	310.26202	310.26202	310.26202	308.18633	308.78983	310.26202	310.26202	310.26202
82	310.20106	306.80863	310.20106	308.30215	308.73497	310.20106	310.20106	310.20106
83	305.58029	305.48275	307.83581	305.45227	305.44922	305.45532	305.54066	305.5559
84	305.21148	304.95545	307.75961	304.84572	304.9585	304.47996	304.98593	304.5653
85	305.20538	304.96459	307.43957	304.84572	304.95545	304.43729	304.97069	304.55921
86	306.43068	305.23891	305.90338	305.52238	305.7845	305.22367	305.59553	305.20538
87	306.22951	304.7299	305.97653	305.55286	305.89423	304.88839	305.66563	305.23891
88	304.58664	306.7812	306.7812	304.44338	304.44034	304.37633	304.44034	304.43729
89	304.70246	303.37963	306.93055	303.92218	304.08982	303.16322	304.30622	303.64176
90	304.56226	303.50155	307.07686	304.0441	304.6537	303.27295	304.4251	303.87036
91	305.34559	306.5465	305.10175	304.93411	304.93716	306.5465	306.5465	306.5465
92	305.34559	306.58003	305.0987	304.62626	304.82743	304.55006	304.83658	304.3489
93	305.36998	303.9557	305.16881	304.69637	304.86706	304.47996	304.87925	304.37633
94	317.0743	318.80556	318.80556	315.92825	318.80556	315.71489	318.80556	318.80556
95	317.08344	318.7507	318.7507	315.9191	318.7507	314.74258	318.7507	318.7507
96	317.09563	318.8909	318.8909	316.05626	318.8909	314.76391	318.8909	318.8909
97	313.42279	314.73953	314.73953	313.22162	314.73953	313.20943	314.73953	314.73953
98	313.38622	314.38596	314.38596	312.69737	314.38596	312.07558	314.38596	314.38596
99	313.38926	315.04128	315.04128	312.74918	315.04128	312.13349	315.04128	315.04128
100	313.0296	313.97753	313.97753	313.97753	313.97753	313.97753	313.97753	313.97753
101	313.06618	313.80989	313.80989	312.05119	313.80989	313.80989	313.80989	313.80989
102	313.12104	314.36462	314.36462	312.1213	314.36462	310.83199	314.36462	314.36462

```
1: C   Last change: JMH 21 Sep 2003  10:45 pm
2: C   Last change: ERB  3 Sep 2002   3:03 pm
3: C=================================================
4:      SUBROUTINE PES1GAU1AP(X,ND,NPE,HOBS,WT,WP,C,SCLE,G,H,DD,DMAX,CSA,
5:     &                  TOL,IND,IFO,AMP,AP,DMX,IOUT,B1,ITERP,IPRINT,
6:     &                  LN,MPR,PRM,JMAX,NFIT,R,GD,U,NOPT,XD,S,
7:     &                  SOSR,NIPR,IPR,BUFF,WTP,NHT,WTQ,IOWTQ,NDMH,
8:     &                  IOSTAR,NPLIST,MPRAR,IPRAR,NDMHAR,BPRI,RMARM,
9:     &                  IAP,DMXA,NPAR,AMPA,AMCA,AAP,ITMXP,RMAR,IPNG,
10:    &                  NPNG,NPNGAR)
11: C   VERSION 20030829 JMH Modified to accept alternative regressions
12: C   ******************************************************************
13: C   REGRESSION SOLUTION FOR THE MODFLOW INVERSE PROCEDURE
14: C   CALLED FROM MF2K LINE 2045
15: C   ******************************************************************
16: C     SPECIFICATIONS:
17: C X(NPE,ND)     REAL(4)  out 2-dimensional array of calculated sensitivities
18: C ND         INTEGER(4)   in Total number of dependent-variable observations
19: C NPE        INTEGER(4)   in Number of sensitivity parameters
20: C HOBS(ND)      REAL(4)   in Observed dependent-variable values
21: C WT(ND)        REAL(4)   in Weights of the observed heads, changes in head
22: C WP(MPRAR)     REAL(4)  ref Weights for prior-information equations
23: C C(NPE,NPE)    REAL(8) inout Coefficient matrix of the Gauss-Newton method
24: C SCLE(NPE)     REAL(8)   in Diagonal components of the unscaled coefficient
25: C G(NPE)        REAL(8) inout Gradient of the objective function with respect
26: C H(ND)         REAL(4)   in Simulated equivalent dependent-variable values
27: C DD(NPE)       REAL(8)   in Calculated changes in parameters, Dr of eqn. 4A
28: C DMAX          REAL(4)   in Maximum allowed fractional change for parameters
29: C CSA           REAL(4)  ref Search direction adjustment parameter GAU
30: C TOL           REAL(4)   in Parameter-estimation convergence criterion.
31: C IND        INTEGER(4)  out Indicates least-squares matrix singularity
32: C IFO        INTEGER(4)  out Indicates parameter estimation convergence
33: C AMP           REAL(4) inout Marquardt parameter Mr of equation 4a of WRIR 98
34: C AP            REAL(8) inout Factor which scales the parameter change vector
35: C DMX           REAL(4) inout Maximum calculated fractional change for parame
36: C IOUT       INTEGER(4)  ref Unit number for Global or List file
37: C B1(NPLIST)    REAL(4) inout Original parameter estimates
38: C ITERP      INTEGER(4)  ref Counter for parameter-estImation iteration loop
39: C IPRINT     INTEGER(4)  ref Controls printing of various statistics
40: C LN(NPLIST) INTEGER(4)   in Flag indicating whether parameter is log-transf
41: C MPR        INTEGER(4)  ref # of prior-information equations to be used
42: C PRM(NPLIST+1,MPRAR)
43: C            REAL(4)   in 2-dimensional array containing coefficients
44: C JMAX       INTEGER(4) inout # of parameter that controls damping in modif
45: C NFIT       INTEGER(4)  out # of Fletcher-Reeves iterations used in combine
46: C R(NPE*NPE/2+NPE)
47: C            REAL(8)   out Estimate of the difference between XwX a
48: C GD(NPE)       REAL(8)  ref Correlation matrix for the parameters
49: C U(NPE)        REAL(8)  out U is the weight matrix for prior information
50: C NOPT       INTEGER(4)  ref adjusts Gauss-Newton matrix with updates
51: C XD(NPE,ND)    REAL(4)  ref Sensivitiy matrix evaluated at npe parameter
52: C S(NPE)        REAL(8)   in S=DMXr/[(rho*DMX)r-1]  see p.81 WR984005
53: C SOSR          REAL(4)  ref Criterion for R, described in OFR 00-184, p.80
54: C NIPR(IPRAR) INTEGER(4)  ref Parameter number of a parameter for which cor
55: C IPR        INTEGER(4)  ref # of parameters included in the var-covar mat.
56: C BUFF(MPRAR)   REAL(4)  ref General-purpose storage space used as require
```

```
C WTP(IPRAR,IPRAR)
C             REAL(4)   ref Variance-covariance matrix for correlate
C NHT         INTEGER(4)   ref Total number of head observations
C WTQ(NDMHAR,NDMHAR)
C             REAL(4)   ref Weight matrix for head-dependent flow
C IOWTQ       INTEGER(4)   ref 0 by default and is set to 1 for covar matrices
C NDMH        INTEGER(4)   ref # of observations minus # of head observations
C IOSTAR      INTEGER(4)   ref When equal to 1, suppresses printing t
C NPLIST      INTEGER(4) inout Number of parameters listed in the Sensitivity
C MPRAR       INTEGER(4)   ref Number of prior-information equations
C IPRAR       INTEGER(4)   ref Number of full variance parameters
C NDMHAR      INTEGER(4)   ref Equals NDMH, but not less than 1
C BPRI(IPRAR)    REAL(4)   ref Prior parameter estimate
C RMARM          REAL(4)   ref Used in calculation of Marquardt parameter
C IAP         INTEGER(4)   out Indicates whether MAX-CHANGE applies nat
C DMXA(ITMXP)    REAL(8)   ref Maximum calculated fractional parameter chang
C NPAR(ITMXP)  INTEGER(4)   ref Parameter number of parameter that controls p
C AMPA(ITMXP)    REAL(4)   ref Marquardt parameter used each parameter-estim
C AMCA(ITMXP)    REAL(4)   ref Maximum parameter change allowed each paramet
C AAP(ITMXP)     REAL(4)   ref Damping parameter used each parameter-estimati
C ITMXP       INTEGER(4)   ref Maximum number of parameter-estimation iters.
C RMAR           REAL(4)   ref Used in calculation of Marquardt parameter.
C IPNG(NPNGAR) INTEGER(4)   ref Indicates whether a variable is allowed t
C NPNG        INTEGER(4)   ref Dimension for IPNG
C NPNGAR      INTEGER(4)   ref If NPNG is 0 then set NPNGAR=1
C     ------------------------------------------------------------------
      REAL ABDMXU, ADMXN, ADMXP, ADMXU, AMP, AP1, APU, B1, BDMXU,
     &    BPRI, BUFF, CSA, DDC, DET, DMAX, DMX, DMX1, DMXN, DMXO, DMXP,
     &    DMXU, H, HOBS, PRM, SOSR, TMPA, TMPB, TOL, W, WP, WT,
     &    WTQ, X, XD, BO
      INTEGER I, IAP, IFO, IND, IOSTAR, IOUT, IOWTQ, IP, IP1, IPM, IPNG,
     &       IPR, IPRINT, ITERP, J, JJN, JJP, JJU, JMAX, LN, LNN, MPR,
     &       N, ND, NDMH, NFIT, NHT, NIPR, NOPT, NPE, NPNG, NPNGAR
      DOUBLE PRECISION ABDMX, ABDMXN, ABDMXP, AP, APN, APO, APP, BDMX,
     &             BDMXN, BDMXP, SPR
      DOUBLE PRECISION C(NPE,NPE), SCLE(NPE), DD(NPE), DTMPA,
     &             R(NPE*NPE/2+NPE),  S(NPE),
     &             U(NPE), G, GD, DMXA(ITMXP)
      CHARACTER*1 CTYPE, DTYPE
      DIMENSION X(NPE,ND), HOBS(ND), WT(ND), WP(MPRAR), H(ND),
     &        BPRI(IPRAR), B1(NPLIST), G(NPE), LN(NPLIST),
     &        PRM(NPLIST+1,MPRAR), GD(NPE), XD(NPE,ND), NIPR(IPRAR),
     &        BUFF(MPRAR), NPAR(ITMXP), AMPA(ITMXP), AMCA(ITMXP),
     &        AAP(ITMXP), IPNG(NPNGAR)
      DIMENSION WTQ(NDMHAR,NDMHAR), WTP(IPRAR,IPRAR)
C Define some additional LAD variables
C the conditional flag LADFILE
C the file IO numbers LUNIT,LOUT
C and the Array to store the data
      CHARACTER*75 LADTITLE
      LOGICAL EXISTS
      DOUBLE PRECISION A
      ALLOCATABLE A(:,:),BO(:)
      INTEGER L,LADFILE,LUNIT,LOUT,M
      INCLUDE 'param.inc'
      INCLUDE 'parallel.inc'
```

```
C
C Begin new code to implement LAD, IO number and file exists FLAG
C
      LUNIT=88
      LOUT=89
      LADFILE=0
C
C First count the number of data values
C
      LADCOUNT=0
      DO 10 I=1,ND
        IF(H(I).GT.1 .AND. HOBS(I).GT.10) THEN
          LADCOUNT=LADCOUNT+1
C         WRITE(IOUT,'(I5,3G16.8)')LADCOUNT,WT(I),H(I),HOBS(I)
        ENDIF
   10 CONTINUE
C
C Check if the Lad file exists with previous data, M is
C the number of data values and the format for NPE=2 is
C TITLE
C M N
C WEIGHT(1) 1ST_RUN(1) 2ND_RUN(1) OBSERVED_DATA(1)
C WEIGHT(2) 1ST_RUN(2) 2ND_RUN(2) OBSERVED_DATA(2)
C ...
C WEIGHT(M) 1ST_RUN(M) 2ND_RUN(M) OBSERVED_DATA(M)
C
      INQUIRE (FILE='modflow.lad',EXIST=EXISTS)
      IF (EXISTS) THEN
        ITERP=ITERP+1
C       WRITE(IOUT,'(T4,A,I5)') 'The current iteration is ',ITERP
        IF(ITERP .EQ. ITMXP) RETURN
        LADFILE=1
        OPEN (UNIT=LUNIT,FILE='modflow.lad',STATUS='OLD')
        READ(LUNIT,'(A)') LADTITLE
        READ(LUNIT,*) M,N
C       IF (LADCOUNT.NE.M) THEN
C         WRITE (IOUT,'(A)') ' LADCOUNT .NE. M '
C       ENDIF
C For some M and NPE=2 here is a sample modflow.lad file
CThis is the LAD output of modflow
C  89   3
C  1.0000000     305.91418     306.11099     305.19016
C  1.0000000     305.90237     306.09866     305.15051
C  1.0000000     305.88812     306.08389     305.16272
C  1.0000000     305.87631     306.07153     305.04385
C  1.0000000     305.85837     306.05273     305.11395
C ...
C Allocate memory for the M=LADCOUNT+2 values and N=NPE+3
        ALLOCATE(A(M+2,N+2))
        READ(LUNIT,*) ((A(I,J),J = 1,N+1),I = 1,M)
C       DO 5 I = 1, M
C         DO 4 J = 1, N+1
C           WRITE (IOUT,'(A,I3,A,I3,A,G15.8)') 'A(',I,',',J,')=',A(I,J)
C  4      CONTINUE
C  5    CONTINUE
C Allocate memory for the BO (old B) NPE
```

```
169:       ALLOCATE(BO(NPE))
170:       READ(LUNIT,*) (BO(I),I = 1,NPE)
171:       DO 6 I = 1, NPE
172: C      WRITE (IOUT,'(A,I3,A,G15.8)') 'BO(',I,')=',BO(I)
173:     6  CONTINUE
174:      ENDIF
175: C
176: C if LADFILE.NE.0 then modflow.lad exists and A has been allocated
177: C shift the data over one column and store the current H in 2nd to
178: C last column.
179: C If NPE=1 then there will be WT,H2,H,HOBS columns
180: C If NPE=2 then there will be WT,H3,H2,H,HOBS columns
181: C If NPE=3 then there will be WT,H4,H3,H2,H,HOBS columns
182: C etc.
183: C
184:      IF (LADFILE.NE.0) THEN
185:       L=0
186:       DO 12 I=1,ND
187:        IF(H(I).GT.1 .AND. HOBS(I).GT.10) THEN
188:         L=L+1
189: C the first column of A is the weight A(L,1)=WT(I)
190:         A(L,1) = WT(I)
191: C shift rows over and insert previous A(L,3) into A(L,2) etc.
192:        DO 11 J=1,NPE
193:         A(L,J+1) = A(L,J+2)
194:    11   CONTINUE
195: C set the current H(I) into the third from last column
196:         A(L,N) = H(I)
197: C the last column of data is the observations
198:         A(L,N+1) = HOBS(I)
199:         A(L,N+2) = 0.
200:        ENDIF
201:    12  CONTINUE
202:       REWIND(LUNIT)
203:       WRITE(LUNIT,'(A)') ' This is the LAD output of modflow'
204:       WRITE(LUNIT,'(1X,I4,I4)') L,N
205:    13  FORMAT(1X,15G15.8)
206:       DO 14 I=1,L
207:        WRITE(LUNIT,13) (A(I,J),J = 1,N+1)
208: C       WRITE(IOUT,13) (A(I,J),J = 1,N+1)
209:    14  CONTINUE
210:       DO 15 I=1,NPE
211:        WRITE(LUNIT,'(T4,G16.8)')B(I)
212:    15  CONTINUE
213:       CLOSE(LUNIT)
214:      ELSE
215: C This is the first time that the lad condition has been encountered
216: C so initialize the data file with the weights,hobs,h,hobs data
217: C The A matrix has M+2,NPE+3 dimensions for the LAD subroutines
218: C WT in 1st, NPE columns, OBS in 2nd to last, and last column results
219:       ALLOCATE(A(LADCOUNT+2,NPE+3))
220:       ALLOCATE(BO(NPE))
221:       OPEN (UNIT=LOUT,FILE='modflow.lad',STATUS='NEW')
222:       WRITE(LOUT,'(A)') ' This is the LAD1 output of modflow'
223:       WRITE(LOUT,'(1X,I4,I4)') LADCOUNT,NPE+1
224: C There's alot of data that is not relevant so loop through the
```

```
C dataset and search only for H!=0 and HOBS>0
      L=0
      N=NPE+1
      DO 17 I=1,ND
       IF(H(I).GT.1 .AND. HOBS(I).GT.10) THEN
        L=L+1
        A(L,1) = WT(I)
       DO 16 J=1,NPE
        A(L,J+1) = HOBS(I)+.003
   16   CONTINUE
        A(L,N) = H(I)
        A(L,N+1) = HOBS(I)
        A(L,N+2) = 0.
       ENDIF
   17  CONTINUE
C
C now write out the WT,Heads,HOBS
C
      N=NPE+2
      DO 20 I=1,LADCOUNT
       WRITE(LOUT,'(1X,14G15.8)') (A(I,J),J = 1,N)
C       WRITE(IOUT,'(1X,14G15.8)') (A(I,J),J = 1,N)
   20  CONTINUE
C save the old B paramaters for later comparison in BO
C write the current B paramaters
      DO 21 I=1,NPE
       BO(I)=B1(I)
       WRITE(LOUT,'(T4,G16.8)')B1(I)
C       WRITE(IOUT,'(T4,G16.8)')B1(I)
   21  CONTINUE
      CLOSE(LOUT)
      M=L
      N=NPE+1
     ENDIF
C
C BO is from modflow.lad and B will be the new adjusted parameters
C
     IF (LADFILE.NE.0) THEN
       CALL PES1LAD1(A,M,N,DMAX,TOL,IND,IFO,IOUT,IPRINT,B,BO,MXPAR)
C if IFO == 1 then LAD converged
       IF(IFO .EQ. 1) GO TO 24
     ENDIF
C
C This is a temporary fix until I figure out how to call lad
C directly from mf2k.  Now, we depend upon the gaussian
C procedure to set all the matrices that pes1bas6 requires.
C
C
C B1 is the original and B will be the new adjusted parameters
C
      CALL SPES1GAU1(X,ND,NPE,HOBS,WT,WP,C,SCLE,G,H,DD,DMAX,CSA,
     &               TOL,IND,IFO,AMP,AP,DMX,IOUT,B1,ITERP,IPRINT,
     &               LN,MPR,PRM,JMAX,NFIT,R,GD,U,NOPT,XD,S,
     &               SOSR,NIPR,IPR,BUFF,WTP,NHT,WTQ,IOWTQ,NDMH,
     &               IOSTAR,NPLIST,MPRAR,IPRAR,NDMHAR,BPRI,RMARM,
     &               IAP,DMXA,NPAR,AMPA,AMCA,AAP,RMAR,IPNG,
```

```
281:     &                NPNG,NPNGAR,B,MXPAR,ITMXP,CTYPE,DTYPE,
282:     &                IPPTR,PARNAM)
283: C
284:   24 CONTINUE
285:      DEALLOCATE(A,BO)
286:      RETURN
287:      END
288: C
289: C    Last change:  JMH  30 Aug 2003   10:45 pm
290: C=================================================================
291: C    VERSION 20030830 JMH
292: C    ******************************************************************
293: C    REGRESSION BY THE LEAST ABSOLUTE DEVIATION METHOD
294: C    ******************************************************************
295:      SUBROUTINE PES1LAD1(A,M,N,DMAX,TOL,IND,IFO,IOUT,IPRINT,B,BO,NB)
296: C A            REAL(8)    in WT,H(previous),H(current),HOBS
297: C M          INTEGER(4)    in number of values that are significant
298: C N          INTEGER(4)    in NPE+1
299: C DMAX          REAL(4)    in Maximum allowed fractional change for parameters
300: C TOL           REAL(4)    in Parameter-estimation convergence criterion.
301: C IND         INTEGER(4)   out singularity indicator, IND=1 singular
302: C IFO         INTEGER(4)   out Indicates parameter estimation convergence
303: C IOUT         INTEGER(4)    in output file descriptor
304: C IPRINT       INTEGER(4)   out controls printing of statistics
305: C B            REAL(4)   out NPE parameters to be adjusted
306:      REAL    DMAX, TOL, B, BO, BTMP, HTMP, DTOLER
307:      INTEGER M, N, IND, IFO, IOUT, IPRINT
308:      REAL*8  A, AS, Y, YH, TOLER, LB, LX, LY, LE,RM,G
309:      REAL*8  D1,E1,V1,G1,T1,RH1,RH2,P1,PY,PE,R,RR,R2,RR2
310:      DIMENSION A(M+2,N+2),B(NB),BO(N-1)
311: C
312: C Original CSU LAD code used B,E,X these renamed to LB, LE, LX for LAD
313: C
314:      ALLOCATABLE AS(:,:),LY(:),YH(:),LB(:),LE(:),LX(:),RM(:,:),G(:)
315: C
316:      INTERFACE
317:        SUBROUTINE SPES1LAD1L1(M,N,A,B,TOLER,X,E)
318:          INTEGER M,N
319:          REAL*8 A,B,TOLER,X,E
320:          DIMENSION  A(:,:),X(:),E(:),B(:)
321:        END SUBROUTINE SPES1LAD1L1
322:      END INTERFACE
323: C     ----------------------------------------------------------------
324:      TOLER=0.1000E-08
325: C TOL tolerance criteria must be less than 1E-09 in order to provide
326: C TOL the best results for convergence.  DTOLER varies from the TOLER
327: C TOL in that TOLER is used in the SPES1TOL1L1 subroutine to
328: C TOL calculate an L1 solution to an over-determined system of linear
329: C TOL equations whereas DTOLER is used to determine if the sum of
330: C TOL of the differences between previously calculated inverse
331: C TOL parameters and current are less than DTOLER
332:      DTOLER=TOL
333:      ALLOCATE(AS(M+2,N+2),LY(M),YH(M),LB(M),LE(M),LX(N))
334: C
335: C LAD variables:
336: C    M == Number of non-zero values from the dependent variables
```

```
C     N == NPE
C    LX == LAD Regression coefficients
C    LY == HOBS Observations
C    RM == Results from the Model
C     G == generalized Expectation Maximization values
C
C RM=WT,H,HOBS
      ALLOCATE(RM(M,N),G(N))
       DO 1890 I = 1,M
         DO 1880 J = 1,N
           RM(I,J) = A(I,J)
 1880    CONTINUE
 1890 CONTINUE
      DO 1910 I = 1,M
         LB(I) = A(I,N+1)
 1910 CONTINUE
      DO 1930 I = 1,M
         DO 1920 J = 1,N+1
           AS(I,J) = A(I,J)
 1920    CONTINUE
         LY(I) = LB(I)
 1930 CONTINUE
      CALL SPES1LAD1L1(M,N,A,LB,TOLER,LX,LE)
      WRITE(IOUT,'(A)') ' '
      WRITE(IOUT,1935)
      WRITE(IOUT,'(T4,A,G15.8)') 'LAD  L1 TOLERANCE SET TO ',TOLER
      WRITE(IOUT,'(T4,A,G15.8)') 'LAD SUM TOLERANCE SET TO ',DTOLER
      WRITE(IOUT,'(A)') ' '
 1935 FORMAT('  *******************************************************')
      WRITE(IOUT,'(A)') ' '
      WRITE(IOUT,1935)
      DO 1940 I = 1,N
         WRITE(IOUT,1950) 'LAD REGRESSION COEFFICIENT',I,' = ',LX(I)
 1940 CONTINUE
 1950 FORMAT(T4,A,I2,A,G15.8)
      IRANK = A(M+1,N+2)
      ITER = A(M+2,N+2)
      IEXCODE = A(M+2,N+1)
      CALL SPES1LAD1LT(M,N,A,AS,LY,YH,LB,LE,LX,TOLER,
     t            D1,E1,V1,G1,T1,RH1,RH2,P1,PY,PE,R,RR,R2,RR2)
      CALL SPES1LAD1PR(IOUT,
     &            D1,E1,V1,G1,T1,RH1,RH2,P1,PY,PE,R,RR,R2,RR2)
C
      CALL SPES1LAD1EM(M,N,LX,LY,RM,G)
      WRITE(IOUT,'(A)') ' '
      DO 2000 I = 1,N
        WRITE(IOUT,'(T4,A,I2,A,G15.8)') 'EM COEFFICIENT',I-1,' = ',G(I)
 2000 CONTINUE
      WRITE(IOUT,'(A)') ' '
      WRITE(IOUT,1935)
C G is the change returned from the EM procedure
C G(1) is the general weight
C G(2) is the weight for the 1st sensitivity parameter
C G(3) is the weight for the 2nd sensitivity parameter
C ...
C first set the general weight to a positive value
```

```
393:      IF(G(1) .LT. 0) G(1)=-1.0*G(1)
394:      DO 2005 I = 2,N
395:        BTMP=0.001*(G(I)/G(1))
396:        WRITE(IOUT,'(T4,A,I2,A,G15.8)') 'EM DELTA',I-1,' = ',BTMP
397:  2005 CONTINUE
398: C
399:      DO 2010 I=2,N
400: C find the change from the EM data and set into BTMP
401: C use the G(I) weight to adjust the B(I-1) parameter
402: C the EM numbers are relative and not absolute changes
403: C however the sign and the change are significant
404:        BTMP=0.001*(G(I)/G(1))
405: C the user has set the maximum change allowed in any
406: C iteration so check if BTMP exceeds this DMAX value
407:        IF(ABS(BTMP) .GT. DMAX) THEN
408: C must keep the same sign as the EM procedure returns
409:          IF(BTMP .LT. 0.0) THEN
410:            BTMP=-1.0*DMAX
411:          ELSE IF(BTMP .GT. 0.0) THEN
412:            BTMP=DMAX
413:          ENDIF
414:        ENDIF
415: C only allow positive sensitivity parameters
416:        IF((B(I-1)+BTMP) .GT. 0.0) B(I-1)=B(I-1)+BTMP
417:        WRITE(IOUT,'(T4,A,I2,A,G15.8)') 'B DELTA',I-1,' = ',BTMP
418:  2010 CONTINUE
419:      DO 2020 I = 1,N-1
420:        WRITE(IOUT,'(T4,A,I2,A,G15.8)') 'LAD B  ',I,' = ',B(I)
421:  2020 CONTINUE
422: C find the total linear difference between original BO
423: C and new B sensitivity parameters
424:       BTMP=0
425:       DO 2030 I=1,N-1
426:          BTMP=BTMP+ABS(BO(I)-B(I))
427:  2030  CONTINUE
428:       IF(BTMP .LT. DTOLER) THEN
429:        HTMP=0
430:        DO 2040 I=1,M
431:          IF(A(I,3).GT.1 .AND. A(I,4).GT.10) THEN
432:            HTMP=HTMP+ABS(A(I,3)-A(I,4))
433:          ENDIF
434:  2040   CONTINUE
435:        WRITE(IOUT,'(T4,A,G15.8)') 'HTMP = ',HTMP
436:        IF(HTMP .LT. DTOLER) THEN
437:         WRITE(IOUT,'(A)') ' LAD convergence criteria satisfied '
438:         IFO = 1
439:        ENDIF
440:       ENDIF
441:      DEALLOCATE(AS,LY,YH,LB,LE,LX,RM,G)
442:      RETURN
443:      END
444: C
445: C    Last change:  JMH  29 Aug 2003   12:45 pm
446: C=======================================================
447: C    VERSION 20020708 ERB
448: C    CALLED FROM PES1GAU1 LINE 250
```

```
C   ****************************************************************
C   REGRESSION BY THE MODIFIED GAUSS-NEWTON METHOD
C   ****************************************************************
    SUBROUTINE SPES1GAU1(X,ND,NPE,HOBS,WT,WP,C,SCLE,G,H,DD,DMAX,CSA,
   &                TOL,IND,IFO,AMP,AP,DMX,IOUT,B1,ITERP,IPRINT,
   &                LN,MPR,PRM,JMAX,NFIT,R,GD,U,NOPT,XD,S,
   &                SOSR,NIPR,IPR,BUFF,WTP,NHT,WTQ,IOWTQ,NDMH,
   &                IOSTAR,NPLIST,MPRAR,IPRAR,NDMHAR,BPRI,RMARM,
   &                IAP,DMXA,NPAR,AMPA,AMCA,AAP,RMAR,IPNG,
   &                NPNG,NPNGAR,B,MXPAR,ITMXP,CTYPE,DTYPE,
   &                IPPTR,PARNAM)
C=======================================================
C   Separated into SPES1GAU1 to make it easier in the future to
C   recode PES1GAU1 into a generic calling module for regression
C   No include files either, analysis determined that the variables
C   B, MXPAR, ITMXP, CTYPE, DTYPE, IPPTR, and PARNAM are required
C   to be passed in the argument list. This eliminated a common block.
C=======================================================
    REAL ABDMXU, ADMXN, ADMXP, ADMXU, AMP, AP1, APU, B, B1, BDMXU,
   &  BPRI, BUFF, CSA, DDC, DET, DMAX, DMX, DMX1, DMXN, DMXO, DMXP,
   &  DMXU, H, HOBS, PRM, SOSR, TMPA, TMPB, TOL, W, WP, WT,
   &  WTQ, X, XD
    INTEGER I, IAP, IFO, IND, IOSTAR, IOUT, IOWTQ, IP, IP1, IPM, IPNG,
   &    IPR, IPRINT, ITERP, J, JJN, JJP, JJU, JMAX, LN, LNN, MPR,
   &    N, ND, NDMH, NFIT, NHT, NIPR, NOPT, NPE, NP1, NPNG, NPNGAR
    INTEGER IPPTR
    DOUBLE PRECISION ABDMX, ABDMXN, ABDMXP, AP, APN, APO, APP, BDMX,
   &          BDMXN, BDMXP, SPR
    DOUBLE PRECISION C(NPE,NPE), SCLE(NPE), DD(NPE), DTMPA,
   &          R(NPE*NPE/2+NPE),  S(NPE),
   &          U(NPE), G, GD, DMXA(ITMXP)
    CHARACTER*1 CTYPE, DTYPE
C PARNAM should be CHARACTER*10 based on usage in other modules - JMH
    CHARACTER(*) PARNAM
    DIMENSION X(NPE,ND), HOBS(ND), WT(ND), WP(MPRAR), H(ND),
   &    B(MXPAR), BPRI(IPRAR), B1(NPLIST), G(NPE), LN(NPLIST),
   &    PRM(NPLIST+1,MPRAR), GD(NPE), XD(NPE,ND), NIPR(IPRAR),
   &    BUFF(MPRAR), NPAR(ITMXP), AMPA(ITMXP), AMCA(ITMXP),
   &    AAP(ITMXP), IPNG(NPNGAR), IPPTR(MXPAR), PARNAM(MXPAR)
    DIMENSION WTQ(NDMHAR,NDMHAR), WTP(IPRAR,IPRAR)
C
 490 FORMAT (/,1X,71('-'),/,
   &    ' PARAMETER VALUES AND STATISTICS FOR ALL PARAMETER-',
   &    'ESTIMATION ITERATIONS',/,
   &    1X,71('-'),
   &    //,5X,'MODIFIED GAUSS-NEWTON CONVERGES IF THE ABSOLUTE',
   &    ' VALUE OF THE MAXIMUM',/,
   &    ' FRACTIONAL PARAMETER CHANGE (MAX CALC. CHANGE) IS',
   &    ' LESS THAN TOL OR IF THE',/,
   &    ' SUM OF SQUARED, WEIGHTED RESIDUALS CHANGES LESS',
   &    ' THAN SOSC OVER TWO',/,
   &    ' PARAMETER-ESTIMATION ITERATIONS.')

 500 FORMAT (/,' SCALED LEAST-SQUARES MATRIX :')
 505 FORMAT (/,' SCALED GRADIENT VECTOR :')
 510 FORMAT (/,' MODIFIED GAUSS-NEWTON PROCEDURE FOR',
```

```
505:     &       ' PARAMETER-ESTIMATION ITERATION NO. = ',I5,/)
506:   515 FORMAT (' VALUES FROM SOLVING THE NORMAL EQUATION :',/,
507:     &    2X,' MARQUARDT PARAMETER ------------------- = ',G11.5,/,
508:     &    2X,' MAX. FRAC. PAR. CHANGE (TOL=',G10.3,') = ',G11.5,/,
509:     &    2X,'     OCCURRED FOR PARAMETER  "',A,'",' TYPE ',A,/)
510:   520 FORMAT (' CALCULATION OF DAMPING PARAMETER',/,
511:     &    2X,' MAX-CHANGE SPECIFIED: ',G9.2,'   USED: ',G9.2,/,
512:     &    2X,' OSCILL. CONTROL FACTOR (1, NO EFFECT)-- = ',G11.5,/,
513:     &    2X,' DAMPING PARAMETER (RANGE 0 TO 1) ------ = ',G11.5,/,
514:     &    2X,'     CONTROLLED BY PARAMETER "',A,'",' TYPE ',A,/)
515:   525 FORMAT (6(3X,A10))
516:   530 FORMAT (6(2X,1PG11.4))
517:   535 FORMAT (E10.3,I5,/,3E10.3)
518:   540 FORMAT (/,
519:     &' *** PARAMETER ESTIMATION CONVERGED BY SATISFYING THE',
520:     &' TOL CRITERION ***',/)
521:   545 FORMAT (' UPDATED ESTIMATES OF REGRESSION PARAMETERS :',/)
522:   550 FORMAT (/,
523:     &' ERROR: CALCULATED PARAMETER CHANGE OF ',G12.5,/,
524:     &' FOR LOG-TRANSFORMED PARAMETER "',A,'" IS TOO LARGE FOR ',/,
525:     &' CALCULATION OF EXPONENTIAL -- STOP EXECUTION (PES1GAU1AP)')
526: C
527: C-------INITIALIZE SOME VARIABLES.
528:       NP1 = NPE - 1
529:       AMP = 0.0
530:       IND = 0
531: C-------ASSEMBLE LEAST-SQUARES MATRIX (C) AND GRADIENT VECTOR (G)
532: C----------INITIALIZE C AND CONVERT LN PARAMETERS
533:       DO 20 IP = 1, NPE
534:         IIPP = IPPTR(IP)
535:         DO 10 I = 1, NPE
536:           C(I,IP) = 0.0
537:    10   CONTINUE
538:         IF (LN(IIPP).GT.0) B(IIPP) = LOG(B(IIPP))
539:         G(IP) = 0.0
540:    20 CONTINUE
541: C     IF THIS IS THE FIRST ITERATION, PRINT HEADING
542:       IF (ITERP.EQ.1) WRITE (IOUT,490)
543: C
544: C----------CALCULATE SENSITIVITY CONTRIBUTIONS TO C AND G.
545: C----------CALCULATE SUM OF SQUARED RESIDUALS.
546: C
547: C            T
548: C     C = X * w * X + C
549: C
550:       DO 50 N = 1, NHT
551:         TMPA = HOBS(N) - H(N)
552:         W = WT(N)
553:         IF (W.LT.0.) THEN
554:           W = 1.E-20
555: cc ERB 7-8-02         IF (IFO.EQ.0) WT(N) = -WT(N)
556:         ENDIF
557:         DO 40 IP = 1, NPE
558:           DTMPA = DBLE(W)*DBLE(X(IP,N))
559:           DO 30 I = IP, NPE
560:             C(I,IP) = DBLE(X(I,N))*DTMPA + C(I,IP)
```

```
561:   30    CONTINUE
562:         G(IP) = DTMPA*TMPA + G(IP)
563:   40  CONTINUE
564:   50 CONTINUE
565: C     ADD SENSITIVITY CONTRIBUTIONS FROM NON-HEAD DATA WITH FULL WEIGHT
566: C     MATRIX TO C AND G (DEVELOPED BY STEEN CHRISTENSEN, AARHUS UNIVERSITY)
567:      CALL SOBS1BAS6WF(NPE,NHT,NDMH,WTQ,X,C,G,HOBS,H,IOWTQ,NDMHAR,
568:     &              ND)
569: C----------ACCOUNT FOR PRIOR INFORMATION
570: C-------------ESTIMATES OF PARAMETER SUMS
571:      IF (MPR.GT.0) THEN
572:        DO 100 IPM = 1, MPR
573:          TMPA = 0.
574:          DO 70 IP = 1, NPLIST
575: C v1_11     IF (PRM(IP,IPM).GT.0.) TMPA = TMPA + PRM(IP,IPM)*B(IP)
576:            TMPA = TMPA + PRM(IP,IPM)*B(IP)
577:   70     CONTINUE
578:          TMPA = PRM(NPLIST+1,IPM) - TMPA
579:          DO 90 IP = 1, NPE
580:            IIPP = IPPTR(IP)
581:            IF (PRM(IIPP,IPM).EQ.0.) GOTO 90
582:            DTMPA = DBLE(WP(IPM))*DBLE(PRM(IIPP,IPM))
583:            DO 80 I = IP, NPE
584:              C(I,IP) = DBLE(PRM(IPPTR(I),IPM))*DTMPA + C(I,IP)
585:   80       CONTINUE
586:            G(IP) = DTMPA*TMPA + G(IP)
587:   90     CONTINUE
588:  100   CONTINUE
589:      ENDIF
590: C-------------CORRELATED PRIOR
591:      IF (IPR.GT.0) CALL SPES1GAU1PF(IPR,NIPR,WTP,C,G,NPE,NPLIST,B,
592:     &                          IPRAR,BPRI)
593: C-------QUASI-NEWTON ADDITION TO COEFFICIENT MATRIX
594:      IF ((NFIT.GT.0.OR.SOSR.GT.0.) .AND. NOPT.EQ.1 .AND. IFO.EQ.0)
595:     &   CALL SPES1GAU1QN(C,DD,G,NPE,R,GD,U,ITERP,X,ND,HOBS,H,WT,S,
596:     &                NFIT,XD,SOSR,NHT,NDMH,WTQ,IOWTQ)
597: C-------FOR ONE PARAMETER CASE
598:      IF (NPE.LT.2) THEN
599: C-------CALCULATE STEP LENGTH FOR SINGLE-PARAMETER CASE
600:        DET = C(1,1)
601:        SCLE(1) = 1.0
602:        IF (IFO.GT.0) THEN
603:          IF (LN(IPPTR(1)).GT.0) B(IPPTR(1)) = EXP(B(IPPTR(1)))
604:          RETURN
605:        ENDIF
606:        DD(1) = G(1)/DET
607:      ELSE
608: C-------SCALE COEFFICIENT MATRIX AND GRADIENT VECTOR
609:        DO 110 IP = 1, NPE
610:          SCLE(IP) = 1.0
611:          IF (C(IP,IP).GT.1.E-30) SCLE(IP) = DSQRT(C(IP,IP))
612:  110   CONTINUE
613:        DO 130 IP = 1, NP1
614:          DTMPA = SCLE(IP)
615:          IP1 = IP + 1
616:          DO 120 I = IP1, NPE
```

```
617:           C(I,IP) = C(I,IP)/(SCLE(I)*DTMPA)
618:           C(IP,I) = C(I,IP)
619:   120    CONTINUE
620:         G(IP) = G(IP)/DTMPA
621:         C(IP,IP) = 1.0 + AMP
622:   130   CONTINUE
623:       G(NPE) = G(NPE)/SCLE(NPE)
624:       C(NPE,NPE) = 1.0 + AMP
625: C-------PRINT AS INDICATED BY IPRINT
626:       IF (IPRINT.NE.0) THEN
627:         WRITE (IOUT,500)
628:         DO 140 J = 1, NPE
629:           WRITE (IOUT,530) (C(I,J),I=1,NPE)
630:   140    CONTINUE
631:         WRITE (IOUT,505)
632:         WRITE (IOUT,530) (G(I),I=1,NPE)
633:       ENDIF
634: C-------COMPUTE PARAMETER STEP LENGTHS
635:       CALL SPES1GAU1SL(C,DD,G,NPE,CSA,IND,IFO,AMP,DET,RMARM,RMAR)
636: C-------IF MATRIX EQUATION IS SINGULAR, OR CONVERGENCE HAS BEEN REACHED:
637:       IF (IND.GT.0 .OR. IFO.GT.0) THEN
638: C---------CONVERT NATURAL LOGS OF PARAMETER VALUES AND RETURN
639:         DO 150 IP = 1, NPE
640:           IIPP = IPPTR(IP)
641:           IF (LN(IIPP).GT.0) B(IIPP) = EXP(B(IIPP))
642:   150    CONTINUE
643:         RETURN
644:       ENDIF
645: C-------UNSCALE PARAMETER CHANGE VECTOR
646:       DO 160 IP = 1, NPE
647:         DD(IP) = DD(IP)/SCLE(IP)
648:   160   CONTINUE
649:     ENDIF
650: C-----COMPUTE DAMPING PARAMETER AND NEW ESTIMATES OF REGRESSION
651: C-----PARAMETERS
652: C----------DEFINITION OF VARIABLES
653: C----------x IS U for untransformed parameters
654: C----------N for transformed parameters with negative DD
655: C----------P for transformed parameters with positive DD
656: C----------ABDMXx Absolute value of BDMXx
657: C----------ADMXx Values used to calculate the damping parameter (Col B,
658: C----------Table B1)
659: C----------AP   Value of the damping parameter used. Results in all
660: C----------parameters respecting MAX-CHANGE or MAX-CHANGE*
661: C----------APx  Value of the damping parameter if only the U, N, or P
662: C----------parameters were considered
663: C----------BDMXx  Maximum fractional parameter change calculated
664: C----------by the normal equations (Col A, Table B1)
665: C----------DDMAXx Maximum dd value; used only for x= N and P
666: C----------DMAXIx Indicator for adjusting dmax for extreme
667: C----------parameter values (B/B0 of eq. B4)
668: C----------DMXx  Values used for oscillation control (Col C, Table B1)
669: C----------JDx  Parameter that governs damping; JDx may not equal JJx
670: C----------JJx  Parameter with maximum change for convergence testing
671: C----------DEFINITION OF VARIABLE NOT CHANGED IN THE CALCULATIONS
672: C----------IAP = 0, Apply dmax in native space (AP is set to equal 0)
```

```
673: C----------IAP = 1, Apply dmax in parameter optimization
674: C--------------------------(possibly transformed) space
675:       DMXO = DMX
676:       ADMXU = 0.0
677:       ADMXN = 0.0
678:       ADMXP = 0.0
679:       BDMXU = 0.0
680:       BDMXN = 0.0
681:       BDMXP = 0.0
682:       ABDMXU = 0.0
683:       ABDMXN = 0.0
684:       ABDMXP = 0.0
685:       DMAXIU = 0.0
686:       DMAXIP = 0.0
687:       DDMAXN = 0.0
688:       DDMAXP = 0.0
689:       APO = AP
690:       APU = 1.0
691:       APN = 1.0
692:       APP = 1.0
693:       JJU = 1
694:       JJP = 1
695:       JJN = 1
696:       JDP = 1
697:       JDN = 1
698:       DO 170 IP = 1, NPE
699:         IIPP = IPPTR(IP)
700:         DTMPA = 0.0
701:         TMPB = 0.0
702:         LNN = LN(IIPP)
703: C-------PARAMETERS ADJUSTED AS IN REGRESSION SPACE
704:         IF (LNN.LE.0 .OR. IAP.EQ.1) THEN
705:           DTMPA = B(IIPP)
706:           IF (DABS(DTMPA).EQ.0.0) DTMPA = 1.0
707:           DTMPA = DD(IP)/ABS(DTMPA)
708:           TMPB = DABS(DTMPA)
709: C------------IF THIS IS THE LARGEST CHANGE, SAVE THE INFORMATION
710: C------------FOR CONVERGENCE AND OSCILLATION CONTROL
711:           IF(TMPB.GT.ADMXU) THEN
712:             ADMXU = TMPB
713:             JJU = IIPP
714:             DMXU = DTMPA
715:             BDMXU = DMXU
716:             ABDMXU = ABS(DMXU)
717:           ENDIF
718: C------------ADJUST DMAX IF THE PARAMETER VALUE IS VERY SMALL
719:           DMAXTMP = DMAX
720:           IF (B1(IIPP) .NE. 0.0) THEN
721:             DMAXIU = B(IIPP)/B1(IIPP)
722:             CALL SPES1GAU1DM (DMAXTMP,DMAXIU)
723:           ENDIF
724:           IF (DMAXTMP .LT. DMAX) DMAXTMP = DMAX
725: C------------CALCULATE THE DAMPING THAT WOULD BE NEEDED FOR THIS
726: C------------PARAMETER (COL B, TABLE B1)
727:           TEMP = 1.0
728:           IF (TMPB.NE.0.0) TEMP = DMAXTMP/TMPB
```

```
729: C------------IF THIS IS THE SMALLEST DAMPING PARAMETER, SAVE INFORMATION
730:         IF (NPE.EQ.1 .OR. APU.GT.TEMP) THEN
731:           IF (TEMP.LT.1.0) APU = TEMP
732:           JDU = IIPP
733:           DMAXU = DMAXTMP
734:         ENDIF
735:       ENDIF
736: C-------LOG-TRANSFORMED PARAMETERS ADJUSTED AS NATIVE
737:       IF (LNN.GT.0 .AND. IAP.EQ.0) THEN
738: C------------DECREASING PARAMETER VALUES
739:         IF (DD(IP) .LT. 0.0) THEN
740: C----------------DAMPING PARAMETER
741:           IF (DMAX .LT. 1.0) THEN
742:             ADMXN = LOG(-DMAX+1.)/DD(IP)
743:             IF (ADMXN .LT. APN) THEN
744:               APN = ADMXN
745:               JDN = IIPP
746:             ENDIF
747:           ENDIF
748: C-----------------LARGEST CHANGE
749:           IF(NPE.EQ.1 .OR. DD(IP).LT.DDMAXN) THEN
750:             JJN = IIPP
751:             DDMAXN = DD(IP)
752:             DMXN = DD(IP)
753:             IF (B(IIPP).NE.0.0) DMXN = DD(IP)/ABS(B(IIPP))
754:             BDMXN = EXP(DD(IP))-1.0
755:             ABDMXN = ABS(BDMXN)
756:           ENDIF
757:         ELSE
758: C------------INCREASING PARAMETER VALUE
759: C----------------DAMPING PARAMETER
760:           DMAXTMP = DMAX
761:           DMAXIP = EXP(B(IIPP))/EXP(B1(IIPP))
762:           CALL SPES1GAU1DM (DMAXTMP,DMAXIP)
763:           ADMXP = LOG(DMAXTMP+1.)/DD(IP)
764:           IF (ADMXP .LT. APP) THEN
765:             APP = ADMXP
766:             JDP = IIPP
767:             DMAXP = DMAXTMP
768:           ENDIF
769: C-----------------LARGEST CHANGE
770:           IF(NPE.EQ.1 .OR. DD(IP).GT.DDMAXP) THEN
771:             JJP = IIPP
772:             DDMAXP = DD(IP)
773:             DMXP = DD(IP)
774:             IF (B(IIPP).NE.0.0) DMXP = DD(IP)/ABS(B(IIPP))
775:             IF (DD(IP).LE.709.) THEN
776:               BDMXP = EXP(DD(IP))-1.0
777:             ELSE
778:               WRITE(IOUT,550) DD(IP), PARNAM(IIPP)
779:               CALL USTOP(' ')
780:             ENDIF
781:             ABDMXP = ABS(BDMXP)
782:           ENDIF
783:         ENDIF
784:       ENDIF
```

```
785:  170 CONTINUE
786: C
787: C-------FOR CONVERGENCE CHECK, FIND LARGEST BDMXx (NATIVE PARAMETER CHANGE
788: C-------FROM THE NORMAL EQUATIONS) AND ASSOCIATED PARAMETER NUMBER.
789:       IF (ABDMXU.GE.ABDMXP .AND. ABDMXU.GE.ABDMXN) THEN
790:         JCHG = JJU
791:         BDMX = BDMXU
792:         ABDMX = ABDMXU
793:         CTYPE = 'U'
794:         DMX = DMXU
795:       ELSEIF (ABDMXP.GE.ABDMXU .AND. ABDMXP.GE.ABDMXN) THEN
796:         JCHG = JJP
797:         BDMX = BDMXP
798:         ABDMX = ABDMXP
799:         CTYPE = 'P'
800:         DMX = DMXP
801:       ELSEIF (ABDMXN.GE.ABDMXU .AND. ABDMXN.GE.ABDMXP) THEN
802:         JCHG = JJN
803:         BDMX = BDMXN
804:         ABDMX = ABDMXN
805:         CTYPE = 'N'
806:         DMX = DMXN
807:       ENDIF
808: C
809: C-----WITH A MIXTURE OF NATIVE AND TRANSFORMED PARAMETERS, THE
810: C-----DAMPING PARAMETER MAY NOT BE ASSOCIATED WITH THE PARAMETER
811: C-----WITH THE LARGEST NATIVE PARAMETER CHANGE AS CALCULATED BY
812: C-----THE NORMAL EQUATIONS. USE DAMPING PARAMETERS FOR ALL
813: C-----TYPES CALCULATED ABOVE AND PICK THE SMALLEST ONE.
814: C
815: C
816: C-----DETERMINE THE DAMPING PARAMETER
817: C
818: C---------IF ALL APx = 1.0, SET DAMPING PARAMETER TO 1.0
819:       IF (APU.EQ.1.0 .AND. APN.EQ.1.0 .AND. APP.EQ.1.0) THEN
820:         AP=1.0
821:         JJ = JCHG
822:         DMAX1 = DMAX
823:         DTYPE = CTYPE
824: C       DMX FROM ABOVE
825: C--------...OTHERWISE, FIND THE SMALLEST DAMPING PARAMETER
826:       ELSE
827:         IF (APU.LE.APP .AND. APU.LE.APN) THEN
828:           AP = APU
829:           JJ = JDU
830:           DMX = DMXU
831:           DMAX1 = DMAXU
832:           DTYPE = 'U'
833:         ELSEIF (APP.LE.APU .AND. APP.LE.APN) THEN
834:           AP = APP
835:           JJ = JDP
836:           DMX = DMXP
837:           DMAX1 = DMAXP
838:           DTYPE = 'P'
839:         ELSEIF (APN.LE.APU .AND. APN.LE.APP) THEN
840:           AP = APN
```

```
841:         JJ = JDN
842:         DMX = DMXN
843:         DMAX1 = DMAX
844:         DTYPE = 'N'
845:       ENDIF
846:     ENDIF
847: C
848: C---------IF THE SAME PARAMETER CONTROLLED THE DAMPING PARAMETER
849: C---------LAST ITERATION, CONSIDER OSCILLATION CONTROL.
850: C-----------THESE CALCULATIONS ARE DONE IN TRANSFORMED PARAMETER
851: C-----------SPACE USING DMXx
852:     OCF = 1.0
853:     IF (ITERP.GT.1) THEN
854:       IF (JJ.EQ.JMAX) THEN
855:         DMX1 = DMX
856:         SPR = DMX1/(APO*DMXO)
857:         IF (SPR.LT.-1.) THEN
858:           AP1 = .5/ABS(SPR)
859:         ELSE
860:           AP1 = (3.+SPR)/(3.+ABS(SPR))
861:         ENDIF
862:       ELSE
863:         AP1 = 1.0
864:       ENDIF
865:       IF(AP1.LT.AP) then
866:         AP = AP1
867:         OCF = AP
868:       ENDIF
869:     ENDIF
870:     JMAX = JJ
871: C
872: C---------UPDATE PARAMETERS AND CONVERT TO PHYSICAL VALUES
873:     DO 180 IP = 1, NPE
874:       IIPP = IPPTR(IP)
875:         DDC = AP*DD(IP)
876:       B(IIPP) = B(IIPP) + DDC
877:       DD(IP) = DDC
878:       IF (LN(IIPP).GT.0) B(IIPP) = EXP(B(IIPP))
879:  180 CONTINUE
880: C
881: C-------PRINT TO THE LISTING FILE
882:     WRITE (IOUT,510) ITERP
883:     WRITE (IOUT,515) AMP, TOL, BDMX, PARNAM(JCHG),CTYPE
884:     WRITE (IOUT,520) DMAX, DMAX1, OCF, AP, PARNAM(JMAX), DTYPE
885: C-------CHECK FOR PARAMETER VALUES <= 0 THAT SHOULD BE > 0
886:     CALL SPES1GAU1CN(B1,IOUT,IPNG,ITERP,LN,NPE,NPLIST,NPNG,NPNGAR)
887:     WRITE (IOUT,545)
888:     WRITE (IOUT,525) (PARNAM(IPPTR(IP)),IP=1,NPE)
889:     WRITE (IOUT,'(1X)')
890:     WRITE (IOUT,530) (B(IPPTR(IP)),IP=1,NPE)
891: C-------PRINT TO THE SCREEN
892:     IF (IOSTAR.NE.1) THEN
893:       WRITE (*,510) ITERP
894:       WRITE (*,515) AMP, TOL, BDMX, PARNAM(JCHG), CTYPE
895:       WRITE (*,520) DMAX, DMAX1, OCF, AP, PARNAM(JMAX), DTYPE
896:       WRITE (*,545)
```

```
897:       WRITE (*,525) (PARNAM(IPPTR(IP)),IP=1,NPE)
898:       WRITE (*,'(1X)')
899:       WRITE (*,530) (B(IPPTR(IP)),IP=1,NPE)
900:     ENDIF
901: C-------CHECK FOR CONVERGENCE
902:     IF (ABDMX.LE.TOL) THEN
903:       IFO = 1
904:       WRITE (IOUT,540)
905:     ENDIF
906: C-------STORE STATISTICS FOR THIS ITERATION
907:     DMXA(ITERP) = BDMX
908:     NPAR(ITERP) = JMAX
909:     AMPA(ITERP) = AMP
910:     AMCA(ITERP) = DMAX1
911:     AAP(ITERP) = AP
912: C
913:     RETURN
914:     END
915: C=============================================
916:     SUBROUTINE SPES1GAU1DM (DMAX1,DMAXI)
917: C    VERSION 20000424 ERB
918: C    *****************************************************************
919: C    ADJUST DMAX FOR PARAMETER VALUES FAR FROM THE STARTING VALUE SO
920: C    SO THAT EXTREME VALUES CAN RECOVER.
921: C    *****************************************************************
922: C      SPECIFICATIONS:
923: C    -----------------------------------------------------------------
924:     REAL DMAX1, DMAXI, TEMP
925: C    -----------------------------------------------------------------
926:     IF (DMAXI.EQ.0.0 .OR. DMAX1.LT.0.4) RETURN
927:     TEMP = DMAX1
928:     DMAX1 = 1./(DMAXI*(DMAX1+1.)**4)
929:     IF (DMAX1.LT.TEMP) DMAX1=TEMP
930: C
931:     RETURN
932:     END
933: C=======================================
934:     SUBROUTINE SPES1GAU1IN(NPE,C)
935: C-----VERSION 1000 01FEB1992
936: C    CALLED FROM PES1BAS6 LINE 1090
937: C    *****************************************************************
938: C       COMPLETE CALCULATION OF THE INVERSE OF SCALED
939: C       COEFFICIENT MATRIX C STARTING FROM DECOMPOSED MATRIX
940: C       FROM SPES1GAU1SL, FOR NPE>1
941: C    *****************************************************************
942: C       SPECIFICATIONS:
943: C    -----------------------------------------------------------------
944:     INTEGER I, IM1, J, K, KP1, NPE, NP1
945:     DOUBLE PRECISION C(NPE,NPE), DSUM
946: C    -----------------------------------------------------------------
947:     NP1 = NPE - 1
948:     C(NPE,NPE) = 1.0/C(NPE,NPE)
949:     DO 50 K = 1, NP1
950:       KP1 = K + 1
951:       DO 20 I = KP1, NPE
952:         DSUM = 0.0
```

```
      IM1 = I - 1
      DO 10 J = K, IM1
        DSUM = DSUM + C(I,J)*C(J,K)
10    CONTINUE
      C(K,I) = -DSUM
      C(I,K) = -DSUM*C(I,I)
20    CONTINUE
    DO 40 I = 1, K
      DSUM = C(K,I)
      DO 30 J = KP1, NPE
        DSUM = DSUM + C(J,K)*C(I,J)
30    CONTINUE
      C(K,I) = DSUM
      C(I,K) = C(K,I)
40    CONTINUE
50  CONTINUE
    RETURN
    END
C=======================================================================
    SUBROUTINE SPES1GAU1PR(NPE,IOUT,NPLIST,IPR,MPR,IFO,OUTNAM,PAREST,
   &                ITMXP,NPAR,DMXA,IPRINT,AAP,AMCA,RSQA,RSPA,
   &                AMPA,ITERPK,SSPI,SSTO)
C     VERSION 20010613 ERB
C-----VERSION 1001 01JUN1992
C     ***************************************************************
C     PRINT PARAMETER VALUES AND STATISTICS FOR ALL ITERATIONS
C     CALLED FROM PES1BAS6 LINE 1065
C     ***************************************************************
C       SPECIFICATIONS:
C     ---------------------------------------------------------------
    REAL PAREST
    INTEGER IOUT, IP, IPR, IUPA, MPR, ITER, NPAR, NPE, NPLIST
    CHARACTER*200 OUTNAM
    CHARACTER*84 FN
    LOGICAL LOP
    DIMENSION AAP(ITMXP), AMCA(ITMXP), AMPA(ITMXP), NPAR(ITMXP+1),
   &      PAREST(ITMXP+1,NPLIST), RSPA(ITMXP+1), RSQA(ITMXP+1),
   &      SSPI(ITMXP+1), SSTO(ITMXP+1)
    DOUBLE PRECISION DMXA(ITMXP+1)
    INCLUDE 'param.inc'
    INCLUDE 'parallel.inc'
C     ---------------------------------------------------------------
 500  FORMAT (8F14.0)
 515  FORMAT (//,' SUMMARY OF PARAMETER VALUES AND STATISTICS FOR',/,
   &      '     ALL PARAMETER-ESTIMATION ITERATIONS')
 520  FORMAT (/,1X,71('-'))
 525  FORMAT (F10.0,I5,/,3F10.0)
 530  FORMAT (/,1X,'PARAMETER-ESTIMATION ITERATION: ',I4)
 535  FORMAT (/,1X,'FINAL PARAMETER VALUES AND STATISTICS:')
 540  FORMAT (/,1X,'PARAMETER NAME(S) AND VALUE(S):',/)
 545  FORMAT (6(3X,A10))
 550  FORMAT (6(2X,1PG11.4))
 560  FORMAT (1X,3(2X,G10.3,2X))
 565  FORMAT (/,1X,'SUMS OF SQUARED WEIGHTED',
   &    ' RESIDUALS:',/,3X,'OBSERVATIONS',3X,'PRIOR INFO.',4X,
   &    'TOTAL')
```

```
1009:  570 FORMAT (/,1X,'PARAMETER UPDATE:',/,
1010:     &       1X,'MAX CHANGE   MAX CALC.',4X,'AMP OR',/,
1011:     &       1X,'PARAMETER',5X,'CHANGE',7X,
1012:     &       'AGMX',/,1X,A10,2X,G10.3,2X,G10.3)
1013:  580 FORMAT (/,1X,'PARAMETER ESTIMATION DID NOT CONVERGE IN THE ',
1014:     &       'ALLOTTED NUMBER OF ITERATIONS')
1015:  585 FORMAT (/,1X,'*** PARAMETER ESTIMATION CONVERGED BY SATISFYING',
1016:     &       ' THE TOL CRITERION ***')
1017:  590 FORMAT (/,1X,'*** PARAMETER ESTIMATION CONVERGED BY SATISFYING',
1018:     &       ' THE SOSC CRITERION ***')
1019:  600 FORMAT(' ERROR READING VALUE FOR PARAMETER ',A,' FROM UNIT ',I5,/
1020:     &       ' -- STOP EXECUTION (SPES1GAU1PR)')
1021:  620 FORMAT(' ERROR READING DMX, NPAR, AMP FROM UNIT ',I5,/
1022:     &       ' -- STOP EXECUTION (SPES1GAU1PR)')
1023:  630 FORMAT (/,' WARNING: ERROR IN OPENING FILE: ',A,' (SPES1GAU1PR)')
1024:  640 FORMAT (1X,'PARAMETER: ',A,/,1X,'ITERATION   ESTIMATE')
1025:  650 FORMAT (1X,I5,6X,G14.7)
1026:  680 FORMAT(/,1X,'SUMS OF SQUARED WEIGHTED RESIDUALS FOR EACH',
1027:     &' ITERATION',//,
1028:     &8X,' SUMS OF SQUARED WEIGHTED RESIDUALS',/
1029:     &1X,'ITER.',2X,'OBSERVATIONS',2X,'PRIOR INFO. ',5X,'TOTAL')
1030:  690 FORMAT(1X,I5,3(2X,G12.5),2X,A,2X,G12.5)
1031:  695 FORMAT(1X,'FINAL',3(2X,G12.5))
1032:  700 FORMAT(/,
1033:     &' SELECTED STATISTICS FROM MODIFIED GAUSS-NEWTON ITERATIONS',//,
1034:     &8X,'MAX. PARAMETER CALC. CHANGE   MAX. CHANGE    DAMPING',/,
1035:     &' ITER.    PARNAM    MAX. CHANGE',7X,'ALLOWED     PARAMETER',/,
1036:     &' -----   ---------- ',13('-'),3X,13('-'),2X,12('-'))
1037:  710 FORMAT(1X,I4,4X,A,2X,G13.6,3X,G13.6,2X,G12.5)
1038: C
1039: C    WRITE SUMMARY OF PARAMETER VALUES AND STATISTICS FOR ALL
1040: C    PARAMETER-ESTIMATION ITERATIONS
1041:      IF (IPRINT.GT.0) WRITE (IOUT,515)
1042:      LAST = 0
1043: C
1044:      DO 100 ITER = 1, ITERPK
1045:        IF (IPRINT.GT.0) WRITE (IOUT,520)
1046:        IF (NPAR(ITER) .EQ. 0) LAST = 1
1047:        IF (LAST.EQ.0) THEN
1048:          IF (IPRINT.GT.0) WRITE (IOUT,530) ITER
1049:        ELSE
1050:          WRITE (IOUT,535)
1051:        ENDIF
1052:        IF (IPRINT.GT.0 .OR. LAST.NE.0) THEN
1053:          WRITE (IOUT,540)
1054:          WRITE (IOUT,545) (PARNAM(IPPTR(IP)),IP=1,NPE)
1055:          WRITE (IOUT,'(1X)')
1056:          WRITE (IOUT,550) (PAREST(ITER,IPPTR(IP)),IP=1,NPE)
1057:        ENDIF
1058:        RSQPI = 0.0
1059:        IF (IPR.GT.0 .OR. MPR.GT.0) RSQPI = RSPA(ITER) - RSQA(ITER)
1060:        IF (IPRINT.GT.0 .OR. LAST.NE.0) THEN
1061:          WRITE (IOUT,565)
1062:          WRITE (IOUT,560) RSQA(ITER), RSQPI, RSPA(ITER)
1063:        ENDIF
1064:        IF (LAST.EQ.0) THEN
```

```
1065:         IF (IPRINT.GT.0)
1066:     &     WRITE (IOUT,570) PARNAM(NPAR(ITER)), DMXA(ITER), AMPA(ITER)
1067:       ENDIF
1068:       IF (LAST.EQ.1) GOTO 120
1069:   100 CONTINUE
1070:   120 CONTINUE
1071: C
1072:       WRITE (IOUT,520)
1073:       IF (IFO.EQ.0) ITER = ITER - 1
1074: C
1075: C     WRITE SELECTED STATISTICS FOR EACH ITERATION
1076:       WRITE (IOUT,700)
1077:       DO 125 I=1,ITER
1078:         IF (NPAR(I).GT.0) THEN
1079:           WRITE (IOUT,710) I,PARNAM(NPAR(I)),DMXA(I),AMCA(I),AAP(I)
1080:         ENDIF
1081:   125 CONTINUE
1082: C
1083: C     WRITE SSWR FOR EACH ITERATION
1084:       WRITE (IOUT,680)
1085:       DO 130 I=1,ITER
1086:         SSO = SSTO(I) - SSPI(I)
1087:         IF (NPAR(I).GT.0) THEN
1088:           WRITE (IOUT,690) I,SSO,SSPI(I),SSTO(I)
1089:         ELSE
1090:           WRITE (IOUT,695) SSO,SSPI(ITER),SSTO(ITER)
1091:         ENDIF
1092:   130 CONTINUE
1093: C
1094: C     WRITE MESSAGE CONCERNING CONVERGENCE
1095:       IF (IFO.EQ.0) THEN
1096:         WRITE (IOUT,580)
1097:       ELSEIF (IFO.EQ.1) THEN
1098:         WRITE (IOUT,585)
1099:       ELSEIF (IFO.EQ.2) THEN
1100:         WRITE (IOUT,590)
1101:       ENDIF
1102:       WRITE (IOUT,520)
1103: C
1104: C-----WRITE PARAMETER ESTIMATES TO _pa FILE
1105: C     OPEN OUTPUT FILE
1106:       IF (OUTNAM.NE.'NONE' .AND. MYID.EQ.MPROC) THEN
1107:         LENGNAM = NONB_LEN(OUTNAM,200)
1108: C       FIND AN UNUSED FILE UNIT AND OPEN THE FILE
1109:         FN = OUTNAM(1:LENGNAM)//'._pa'
1110:         IUPA = IGETUNIT(1,1000)
1111:         IF (IUPA.GT.0) THEN
1112:           OPEN(IUPA,FILE=FN,ERR=150)
1113:         ELSE
1114:           GOTO 150
1115:         ENDIF
1116:         GOTO 160
1117:   150   CONTINUE
1118:         WRITE(IOUT,630) FN
1119:   160   CONTINUE
1120: C
```

```
1121: C      WRITE PARAMETER ESTIMATES FOR EACH ITERATION FOR EACH
1122: C      PARAMETER TO _pa FILE
1123:       DO 180 IP = 1,NPE
1124:        IIPP = IPPTR(IP)
1125:        WRITE (IUPA,640) PARNAM(IIPP)
1126:        DO 170 IT = 1,ITER
1127:         WRITE (IUPA,650) IT, PAREST(IT,IPPTR(IP))
1128:  170    CONTINUE
1129:  180   CONTINUE
1130: C
1131: C      CLOSE _pa FILE
1132:       INQUIRE(UNIT=IUPA,OPENED=LOP)
1133:       IF (LOP) CLOSE (UNIT=IUPA)
1134:      ENDIF
1135: C
1136:      RETURN
1137:      END
1138: C=============================================================
1139:      SUBROUTINE SPES1GAU1QN  (C,DD,G,NPE,R,GD,U,ITERP,X,
1140:     & ND,HOBS,H,WT,S,NFIT,XD,SOSR,NHT,NDMH,WTQ,IOWTQ)
1141: C-----VERSION 1001 01JAN1998
1142: C    CALLED FROM SPES1GAU1 LINE 494
1143: C    ******************************************************************
1144: C    COMPUTE QUASI-NEWTON COMPONENT OF C MATRIX AND ADD TO C
1145: C     (NOTE: PRIOR NOT INCLUDED BECAUSE LINEAR TERMS DO NOT
1146: C    CONTRIBUTE)
1147: C      SPECIFICATIONS:
1148: C    ------------------------------------------------------------------
1149:      REAL H, HOBS, SOSR, WT, WTQ, X, XD
1150:      INTEGER I, IIP, IOWTQ, IPP, ITERP, J, JM1, JP1, K, K1, L, L1, M,
1151:     &       N, N1, ND, NDMH, NFIT, NHT, NPE
1152:      DOUBLE PRECISION C(NPE,NPE), DD(NPE), SUM, TMPA, DS, DGD, DU, CF,
1153:     &              QD, TMP, T, R(NPE*NPE/2+NPE), DPGD, DPU, U(NPE),
1154:     &              S(NPE), G, GD, DPGI, W1
1155:      DIMENSION G(NPE), GD(NPE), X(NPE,ND), HOBS(ND), H(ND), WT(ND),
1156:     &        XD(NPE,ND), WTQ(NDMH,NDMH)
1157: C    ------------------------------------------------------------------
1158: C
1159: C------FOR FIRST PARAMETER-ESTIMATION ITERATIONS, INITIALIZE
1160: C------UPDATING MATRIX R AND SAVE G AND X
1161:      IF (ITERP.EQ.1) THEN
1162:       I = 0
1163:       DO 20 IIP = 1, NPE
1164:        DO 10 IPP = IIP, NPE
1165:         I = I + 1
1166:         R(I) = 0.0
1167:   10    CONTINUE
1168:        GD(IIP) = G(IIP)
1169:   20   CONTINUE
1170:       DO 40 N = 1, ND
1171:        DO 30 IIP = 1, NPE
1172:         XD(IIP,N) = X(IIP,N)
1173:   30    CONTINUE
1174:   40   CONTINUE
1175:       RETURN
1176:      ENDIF
```

```
1177: C
1178: C------FOR SUBSEQUENT ITERATIONS, CALCULATE MATRIX R
1179:       DO 50 IIP = 1, NPE
1180:         S(IIP) = 0.0
1181:    50 CONTINUE
1182: C---------CONTRIBUTION FROM OBSERVATIONS WITH DIAGONAL WEIGHTING
1183:       IF (NHT.GT.0) THEN
1184:         DO 70 N = 1, NHT
1185:           IF (WT(N).LT.0.0) GOTO 70
1186:           W1 = (HOBS(N)-H(N))*WT(N)
1187:           DO 60 IIP = 1, NPE
1188:             S(IIP) = S(IIP) + W1*(XD(IIP,N)-X(IIP,N))
1189:    60     CONTINUE
1190:    70   CONTINUE
1191:       ENDIF
1192: C---------CONTRIBUTION FROM OBSERVATIONS WITH FULL WEIGHT MATRIX
1193:       IF (NDMH.GT.0) THEN
1194:         IF (IOWTQ.GT.0) THEN
1195:           DO 100 K = 1, NDMH
1196:             IF (WTQ(K,K).GT.0.0) THEN
1197:               TMP = 0.0
1198:               DO 80 L = 1, NDMH
1199:                 IF (WTQ(L,L).GT.0.0) THEN
1200:                   L1 = NHT + L
1201:                   TMP = TMP + DBLE(WTQ(K,L))*DBLE(HOBS(L1)-H(L1))
1202:                 ENDIF
1203:    80         CONTINUE
1204:               K1 = NHT + K
1205:               DO 90 I = 1, NPE
1206:                 S(I) = S(I) + DBLE(XD(I,K1)-X(I,K1))*TMP
1207:    90         CONTINUE
1208:             ENDIF
1209:   100     CONTINUE
1210:         ELSE
1211:           DO 120 N = 1, NDMH
1212:             IF (WTQ(N,N).GT.0.0) THEN
1213:               N1 = NHT + N
1214:               W1 = (HOBS(N1)-H(N1))*WTQ(N,N)
1215:               DO 110 IIP = 1, NPE
1216:                 S(IIP) = S(IIP) + W1*(XD(IIP,N1)-X(IIP,N1))
1217:   110         CONTINUE
1218:             ENDIF
1219:   120     CONTINUE
1220:         ENDIF
1221:       ENDIF
1222: C
1223:       DO 130 J = 1, NPE
1224:         GD(J) = GD(J) - G(J)
1225:   130 CONTINUE
1226: C
1227:       K = 0
1228:       QD = 0.0
1229:       DS = 0.0
1230:       DO 160 J = 1, NPE
1231:         M = NPE - J + 1
1232:         K = K + 1
```

```
1233:       L = K
1234:       SUM = 0.0
1235:       IF (J.NE.NPE) THEN
1236:         JP1 = J + 1
1237:         DO 140 I = JP1, NPE
1238:           K = K + 1
1239:           SUM = SUM + DD(I)*R(K)
1240:   140     CONTINUE
1241:       ENDIF
1242:       TMPA = DD(J)*R(L)
1243:       QD = QD + (SUM+SUM+TMPA)*DD(J)
1244:       SUM = SUM + TMPA
1245:       IF (J.NE.1) THEN
1246:         JM1 = J - 1
1247:         DO 150 I = 1, JM1
1248:           L = L - M
1249:           SUM = SUM + R(L)*DD(J-I)
1250:           M = M + 1
1251:   150     CONTINUE
1252:       ENDIF
1253:       U(J) = SUM
1254:       DS = DS + DD(J)*S(J)
1255:   160 CONTINUE
1256: C
1257:       TMP = 1.0
1258:       IF (QD.NE.0.0) TMP = DABS(DS/QD)
1259:       T = 1.0
1260:       IF (TMP.LT.1.0) T = TMP
1261:       DGD = 0.0
1262:       DU = 0.0
1263:       DO 170 J = 1, NPE
1264:         DGD = DGD + DD(J)*GD(J)
1265:         U(J) = S(J) - T*U(J)
1266:         DU = DU + DD(J)*U(J)
1267:   170 CONTINUE
1268:       CF = DU/(DGD*DGD)
1269:       K = 0
1270:       DO 190 J = 1, NPE
1271:         DO 180 I = J, NPE
1272:           K = K + 1
1273:           DPGD = GD(J)
1274:           DPU = U(J)
1275:           DPGI = GD(I)
1276:           R(K) = T*R(K) + (U(I)*DPGD+DPGI*DPU)/DGD - CF*DPGI*DPGD
1277:   180   CONTINUE
1278:   190 CONTINUE
1279: C------SAVE GRADIENT
1280:       DO 200 IIP = 1, NPE
1281:         GD(IIP) = G(IIP)
1282:   200 CONTINUE
1283: C-----SAVE SENSITIVITIES
1284:       DO 220 N = 1, ND
1285:         DO 210 IIP = 1, NPE
1286:           XD(IIP,N) = X(IIP,N)
1287:   210   CONTINUE
1288:   220 CONTINUE
```

```
1289: C------ADD QN APPROXIMATION TO COEFFICIENT MATRIX
1290:       IF ((NFIT.GT.0.AND.ITERP.GT.NFIT) .OR. SOSR.LT.0.0) THEN
1291:         I = 0
1292:         DO 240 IIP = 1, NPE
1293:           DO 230 IPP = IIP, NPE
1294:             I = I + 1
1295:             C(IPP,IIP) = C(IPP,IIP) + R(I)
1296:   230     CONTINUE
1297:   240   CONTINUE
1298:       ENDIF
1299:       RETURN
1300:       END
1301: C=============================================
1302:       SUBROUTINE SPES1GAU1SL(C,DD,G,NPE,CSA,IND,IFO,AMP,DET,RMARM,RMAR)
1303: C------VERSION 1000 01FEB1992
1304: C     ***************************************************************
1305: C         COMPUTE PARAMETER STEP LENGTHS USING THE MARQUARDT
1306: C         PROCEDURE
1307: C        SPECIFICATIONS:
1308: C     ---------------------------------------------------------------
1309:       REAL AMP, CSA, DET, SUM, SUMA, SUMB
1310:       INTEGER I, IFO, IND, IP1, J, K, KP1, NM1, NPE
1311:       DOUBLE PRECISION C(NPE,NPE), DD(NPE), DPIV, DTMPA, DSUM, G
1312:       DIMENSION G(NPE)
1313: C     ---------------------------------------------------------------
1314: C         COMPUTE TRIAL PARAMETER STEP LENGTHS USING LDU FACTORIZATION:
1315: C           DECOMPOSE MATRIX
1316:       NM1 = NPE - 1
1317:    10 IND = 0
1318:       DET = 1.
1319:       DO 40 K = 1, NM1
1320:         DPIV = C(K,K)
1321:         DET = DET*DPIV
1322:         IF (DPIV.GT.1.E-13) THEN
1323:           DPIV = 1.0/DPIV
1324:           KP1 = K + 1
1325:           DO 30 J = KP1, NPE
1326:             DTMPA = C(J,K)*DPIV
1327:             DO 20 I = J, NPE
1328:               C(I,J) = C(I,J) - DTMPA*C(I,K)
1329:    20       CONTINUE
1330:    30     CONTINUE
1331:           C(K,K) = DPIV
1332:         ELSE
1333:           IND = 1
1334:           GOTO 110
1335:         ENDIF
1336:    40 CONTINUE
1337:       DET = DET*C(NPE,NPE)
1338:       IF (C(NPE,NPE).LE.1.E-13) THEN
1339:         IND = 1
1340:         IF (IFO.EQ.0) GOTO 110
1341:       ENDIF
1342:       IF (IFO.GT.0) RETURN
1343:       DO 50 K = 1, NPE
1344:         DD(K) = G(K)
```

```
1345:   50 CONTINUE
1346: C         FORWARD SUBSTITUTE
1347:       DO 70 K = 1, NM1
1348:         DTMPA = DD(K)*C(K,K)
1349:         KP1 = K + 1
1350:         DO 60 J = KP1, NPE
1351:           DD(J) = DD(J) - C(J,K)*DTMPA
1352:   60    CONTINUE
1353:   70  CONTINUE
1354: C         BACK SUBSTITUTE
1355:       DD(NPE) = DD(NPE)/C(NPE,NPE)
1356:       I = NPE
1357:   80  I = I - 1
1358:       IF (I.GT.0) THEN
1359:         IP1 = I + 1
1360:         DSUM = 0.0
1361:         DO 90 J = IP1, NPE
1362:           DSUM = DSUM + C(J,I)*DD(J)
1363:   90    CONTINUE
1364:         DD(I) = (DD(I)-DSUM)*C(I,I)
1365:         GOTO 80
1366:       ENDIF
1367: C       CHECK SOLUTION AND ADD MARQUARDT PARAMETER IF NEEDED
1368:       SUM = 0.0
1369:       SUMA = 0.0
1370:       SUMB = 0.0
1371:       DO 100 I = 1, NPE
1372:         SUM = SUM + DD(I)*G(I)
1373:         SUMA = SUMA + DD(I)*DD(I)
1374:         SUMB = SUMB + G(I)*G(I)
1375:   100 CONTINUE
1376:       IF (SUM.GT.CSA*SQRT(SUMA*SUMB)) RETURN
1377:   110 AMP = RMARM*AMP + RMAR
1378:       IF (AMP.GT.1.0) RETURN
1379:       DO 130 I = 1, NPE
1380:         C(I,I) = 1.0 + DBLE(AMP)
1381:         DO 120 J = I, NPE
1382:           C(J,I) = C(I,J)
1383:   120   CONTINUE
1384:   130 CONTINUE
1385:       GOTO 10
1386:       END
1387: C=============================================================
1388:       SUBROUTINE SPES1GAU1PF(IPR,NIPR,WTP,C,G,NPE,NPLIST,B,IPRAR,BPRI)
1389: C     VERSION 20031002 ERB
1390: C     CALLED FROM SPES1GAU1 LINE 588
1391: C*****************************************************************
1392: C     ADD COMPONENTS TO THE MATRIX AND VECTOR OF THE REGRESSION
1393: C     EQUATION FOR PRIOR INFORMATION WITH A FULL WEIGHT MATRIX
1394: C*****************************************************************
1395: C     SPECIFICATIONS
1396:       REAL B, BDIF, BPRI, WPP, WTP
1397:       INTEGER I, IPR, IPRAR, J, K, K1, L, L1, NIPR, NPE, NPLIST
1398:       DOUBLE PRECISION C(NPE,NPE), G(NPE)
1399:       DIMENSION NIPR(IPRAR), B(NPLIST), BPRI(IPRAR), WTP(IPRAR,IPRAR)
1400: C-----------------------------------------------------------------
```

```
1401: C------ADDITIONS TO C AND G
1402:       DO 60 I = 1, NPE
1403:         DO 50 J = I, NPE
1404:           DO 40 K = 1, IPR
1405:             K1 = NIPR(K)
1406:             IF (K1.EQ.I) THEN
1407:               DO 30 L = 1, IPR
1408:                 WPP = WTP(L,K)
1409:                 L1 = NIPR(L)
1410:                 IF (L1.EQ.J) THEN
1411:                   C(J,I) = C(J,I) + WPP
1412:                 ENDIF
1413:                 IF (J.EQ.I) THEN
1414:                   BDIF = BPRI(L) - B(L1)
1415:                   G(I) = G(I) + WPP*BDIF
1416:                 ENDIF
1417:    30         CONTINUE
1418:             ENDIF
1419:    40     CONTINUE
1420:    50   CONTINUE
1421:    60 CONTINUE
1422:       RETURN
1423:       END
1424: C=================================================================
1425:       SUBROUTINE SPES1GAU1CN(B1,IOUTG,IPNG,ITERP,LN,NPE,NPLIST,NPNG,
1426:      &                       NPNGAR)
1427: C-----VERSION 20030128 ERB
1428: C     ******************************************************************
1429: C     CHECK FOR PARAMETER VALUES <= 0 THAT SHOULD BE > 0.
1430: C     ******************************************************************
1431: C       SPECIFICATIONS:
1432: C     ------------------------------------------------------------------
1433:       REAL B1, BOLD
1434:       INTEGER I, IIP, IOUTG, IPNG, LN, NPE, NPNG, NPLIST
1435:       CHARACTER*4 PIDTMP
1436:       DIMENSION B1(NPLIST), IPNG(NPNGAR), LN(NPLIST)
1437:       INCLUDE 'param.inc'
1438: C     ------------------------------------------------------------------
1439:   500 FORMAT (' PARAMETER "',A,
1440:      &       '" < 0 : NOT PHYSICALLY REASONABLE.',/,
1441:      &       ' ESTIMATED VALUE OF ',G13.6,' CHANGED TO ',G13.6,
1442:      &       ' (PES1GAU1CN)',/)
1443:   505 FORMAT (' LN PARAMETER "',A,'" <= 0 : NOT ',
1444:      &       'PHYSICALLY OR MATHEMATICALLY REASONABLE.',/,
1445:      &       ' ESTIMATED VALUE OF ',G13.6,' CHANGED TO ',G13.6,
1446:      &       ' (PES1GAU1CN)',/)
1447: C
1448: cc      IF (ITERP.GT.1) THEN
1449:         DO 20 IIP = 1, NPE
1450:           IIPP = IPPTR(IIP)
1451:           PIDTMP = PARTYP(IIPP)
1452:           IF (B(IIPP).LE.B1(IIPP)/1.E6 .AND. LN(IIPP).LE.0 .AND.
1453:      &        (PIDTMP.EQ.'HK  ' .OR. PIDTMP.EQ.'SS  ' .OR.
1454:      &        PIDTMP.EQ.'SY  ' .OR. PIDTMP.EQ.'VK ' .OR.
1455:      &        PIDTMP.EQ.'VANI' .OR. PIDTMP.EQ.'GHB ' .OR.
1456:      &        PIDTMP.EQ.'RIV ' .OR. PIDTMP.EQ.'STR ' .OR.
```

```
1457:     &      PIDTMP.EQ.'DRN ' .OR. PIDTMP.EQ.'ANI ' .OR.
1458:     &      PIDTMP.EQ.'EVT ' .OR. PIDTMP.EQ.'VKCB' .OR.
1459:     &      PIDTMP.EQ.'DRT ' .OR. PIDTMP.EQ.'ETS ')) THEN
1460:           IF (NPNG.GT.0) THEN
1461:            DO 10 I = 1, NPNG
1462:             IF (IIPP.EQ.IPNG(I)) GOTO 20
1463:    10      CONTINUE
1464:           ENDIF
1465:           BOLD = B(IIPP)
1466:           B(IIPP) = B1(IIPP)/100.
1467:           WRITE (IOUTG,500) PARNAM(IIPP), BOLD, B(IIPP)
1468:         ENDIF
1469:         IF (B(IIPP).LT.1.E-14 .AND. LN(IIPP).GT.0) THEN
1470:           BOLD = B(IIPP)
1471:           B(IIPP) = 1.E-14
1472:           WRITE (IOUTG,505) PARNAM(IIPP), BOLD, B(IIPP)
1473:         ENDIF
1474:    20   CONTINUE
1475: cc      ENDIF
1476: C
1477:       RETURN
1478:       END
1479: C
1480: C LEAST ABSOLUTE DEVIATION
1481: C
1482:       SUBROUTINE SPES1LAD1LT(M,N,A,AS,Y,YH,B,E,X,TOLER,
1483:      1  DELTA1,EDEL1,VAR1,GAM1,T1,RHO1,RHO2,PROB,PY,PE,R,RR,R2,RR2)
1484: C
1485: C    VERSION 20030809 JMH Created
1486: C
1487:       IMPLICIT REAL*8(A-H,O-Z)
1488:       DIMENSION A(M+2,N+2),B(M),E(M),X(N)
1489:       DIMENSION AS(M+2,N+2),Y(M),YH(M)
1490:       DOUBLE PRECISION COOP,DELTA_YH
1491:       INTERFACE
1492:         SUBROUTINE SPES1LAD1L1(M,N,A,B,TOLER,X,E)
1493:           IMPLICIT REAL*8(A-H,O-Z)
1494:           DIMENSION  A(:,:),X(:),E(:),B(:)
1495:         END SUBROUTINE SPES1LAD1L1
1496:         SUBROUTINE SPES1LAD1AL(KG,Y,YH,DELTA,EDEL,VAR,GAM,T,RHO,PROB)
1497:           IMPLICIT REAL*8(A-H,O-Z)
1498:           DIMENSION  Y(:),YH(:)
1499:         END SUBROUTINE SPES1LAD1AL
1500:         SUBROUTINE SPES1LAD1DT(NCASES,DATA,PROB)
1501:           IMPLICIT REAL*8(A-H,O-Z)
1502:           DIMENSION  DATA(:)
1503:         END SUBROUTINE SPES1LAD1DT
1504:       END INTERFACE
1505: C
1506:       ZERO = 0.0D0
1507:       ONE = 1.0D0
1508:       IRANK = A(M+1,N+2)
1509:       ITER = A(M+2,N+2)
1510:       IEXCODE = A(M+2,N+1)
1511:       SUMRE = A(M+1,N+1)
1512:       DELTA1 = SUMRE/M
```

```
1513:      DO 2100 I = 1,M
1514:        YH(I) = Y(I)-E(I)
1515:  2100 CONTINUE
1516:      CO = ZERO
1517:      COO = ZERO
1518:      CP = ZERO
1519:      CPP = ZERO
1520:      COP = ZERO
1521:      DO 2200 I = 1,M
1522:        CO = CO+Y(I)
1523:        COO = COO+Y(I)**2
1524:        CP = CP+YH(I)
1525:        CPP = CPP+YH(I)**2
1526:        COP = COP+Y(I)*YH(I)
1527:  2200 CONTINUE
1528:      COOP=((COO*M-CO*CO)*(CPP*M-CP*CP))
1529:      R = (COP*M-CO*CP)/DSQRT(COOP)
1530:      RR =  R*R
1531: C
1532:      CALL SPES1LAD1AL(M,Y,YH,DELTA1,EDEL1,VAR1,GAM1,T1,RHO1,PROB)
1533:      CALL SPES1LAD1DT(M,Y,PY)
1534:      CALL SPES1LAD1DT(M,E,PE)
1535: C
1536:      MC = M-1
1537:      DELTA2 = ZERO
1538:      DO 2750 I = 1,M
1539:        YH(I) = ZERO
1540:        DO 2550 J = 1,I-1
1541:          DO 2500 K = 1,N+1
1542:            A(J,K) = AS(J,K)
1543:  2500      CONTINUE
1544:          B(J) = AS(J,N+1)
1545:  2550    CONTINUE
1546:        DO 2650 J = I+1,M
1547:          DO 2600 K = 1,N+1
1548:            A(J-1,K) = AS(J,K)
1549:  2600      CONTINUE
1550:          B(J-1) = AS(J,N+1)
1551:  2650    CONTINUE
1552:        CALL SPES1LAD1L1(MC,N,A,B,TOLER,X,E)
1553:        DO 2700 J = 1,N
1554:          YH(I) = YH(I)+X(J)*AS(I,J)
1555:  2700    CONTINUE
1556:        DELTA_YH = (Y(I)-YH(I))
1557:        DELTA2 = DELTA2+DABS(DELTA_YH)
1558:  2750 CONTINUE
1559:      DELTA2 = DELTA2/M
1560:      EDEL2 = ZERO
1561:      DO 2850 I = 1,M
1562:        DO 2800 J = 1,M
1563:          EDEL2 = EDEL2+DABS(Y(I)-YH(J))
1564:  2800    CONTINUE
1565:  2850 CONTINUE
1566:      EDEL2 = EDEL2/M/M
1567:      RHO2 = ONE-DELTA2/EDEL2
1568:      CO = ZERO
```

```
      COO = ZERO
      CP = ZERO
      CPP = ZERO
      COP = ZERO
      DO 2900 I = 1,M
        CO = CO+Y(I)
        COO = COO+Y(I)**2
        CP = CP+YH(I)
        CPP = CPP+YH(I)**2
        COP = COP+Y(I)*YH(I)
 2900 CONTINUE
      R2 = (COP*M-CO*CP)/DSQRT((COO*M-CO*CO)*(CPP*M-CP*CP))
      RR2 =  R2*R2
      RETURN
      END
C
      SUBROUTINE SPES1LAD1PR(IOUT,
     &          D1,E1,V1,G1,T1,RH1,RH2,P1,PY,PE,R,RR,R2,RR2)
C     VERSION 20030809 JMH Created
      IMPLICIT REAL*8(A-H,O-Z)
      WRITE(IOUT,3200) ' '
      WRITE(IOUT,3500)
      WRITE(IOUT,3200) 'OBSERVED VALUE OF DELTA = ',D1
 3200 FORMAT(T4,A,G15.8)
      WRITE(IOUT,3200) 'EXPECTED VALUE OF DELTA = ',E1
      WRITE(IOUT,3200) 'VARIANCE OF DELTA = ',V1
      WRITE(IOUT,3200) 'SKEWNESS OF DELTA = ',G1
      WRITE(IOUT,3200) 'STANDARDIZED VALUE OF DELTA = ',T1
      WRITE(IOUT,3200) 'MEASURE OF AGREEMENT BETWEEN Y AND Y-HAT = ',RH1
      WRITE(IOUT,3200) 'ESTIMATED SHRINKAGE MEASURE OF AGREEMENT = ',RH2
      WRITE(IOUT,3200) 'P-VALUE FOR THIS LAD REGRESSION FUNCTION = ',P1
      WRITE(IOUT,3200) 'FIRST-ORDER AUTOREGRESSIVE P-VALUE OF Y = ',PY
      WRITE(IOUT,3300) 'FIRST-ORDER AUTOREGRESSIVE P-VALUE OF E = ',PE
 3300 FORMAT(T4,A,G15.8,/)
      WRITE(IOUT,3200) 'PEARSON CORRELATION COEFFICIENT = ',R
      WRITE(IOUT,3200) 'SQUARED CORRELATION COEFFICIENT = ',RR
      WRITE(IOUT,3200) 'PEARSON SHRINKAGE CORRELATION COEFFICIENT = ',R2
      WRITE(IOUT,3200) 'SQUARED SHRINKAGE CORRELATION COEFFICIENT = ',RR2
      WRITE(IOUT,3500)
 3500 FORMAT('  ******************************************************')
      RETURN
      END
C
C     ALGORITHM 478 COLLECTED ALGORITHMS FROM ACM.
C     ALGORITHM APPEARED IN COMM. ACM, VOL. 17, NO. 06,
C     P. 319. L1
      SUBROUTINE SPES1LAD1L1(M,N,A,B,TOLER,X,E)
C     VERSION 20030809 JMH Created
C THIS SUBROUTINE USES A MODIFICATION OF THE SIMPLEX METHOD
C OF LINEAR PROGRAMMING TO CALCULATE AN L1 SOLUTION TO AN
C OVER-DETERMINED SYSTEM OF LINEAR EQUATIONS.
C DESCRIPTION OF PARAMETERS.
C M      NUMBER OF EQUATIONS.
C N      NUMBER OF UNKNOWNS (M.GE.N).
C A      TWO DIMENSIONAL REAL ARRAY OF SIZE (M2,N2).
C       ON ENTRY, THE COEFFICIENTS OF THE MATRIX MUST BE
```

```
1625: C       STORED IN THE FIRST M ROWS AND N COLUMNS OF A.
1626: C       THESE VALUES ARE DESTROYED BY THE SUBROUTINE.
1627: C B      ONE DIMENSIONAL REAL ARRAY OF SIZE M. ON ENTRY, B
1628: C       MUST CONTAIN THE RIGHT HAND SIDE OF THE EQUATIONS.
1629: C       THESE VALUES ARE DESTROYED BY THE SUBROUTINE.
1630: C TOLER  A SMALL POSITIVE TOLERANCE. EMPIRICAL EVIDENCE
1631: C       SUGGESTS TOLER=10**(-D*2/3) WHERE D REPRESENTS
1632: C       THE NUMBER OF DECIMAL DIGITS OF ACCURACY AVALABLE
1633: C       (SEE DESCRIPTION).
1634: C X      ONE DIMENSIONAL REAL ARRAY OF SIZE N. ON EXIT, THIS
1635: C       ARRAY CONTAINS A SOLUTION TO THE L1 PROBLEM.
1636: C E      ONE DIMENSIONAL REAL ARRAY OF SIZE M. ON EXIT, THIS
1637: C       ARRAY CONTAINS THE RESIDUALS IN THE EQUATIONS.
1638: C ON EXIT FROM THE SUBROUTINE, THE ARRAY A CONTAINS THE
1639: C FOLLOWING INFORMATION.
1640: C A(M+1,N+1)  THE MINIMUM SUM OF THE ABSOLUTE VALUES OF
1641: C           THE RESIDUALS.
1642: C A(M+1,N+2)  THE RANK OF THE MATRIX OF COEFFICIENTS.
1643: C A(M+2,N+1)  EXIT CODE WITH VALUES.
1644: C           0 - OPTIMAL SOLUTION WHICH IS PROBABLY NON-
1645: C              UNIQUE (SEE DESCRIPTION).
1646: C           1 - UNIQUE OPTIMAL SOLUTION.
1647: C           2 - CALCULATIONS TERMINATED PREMATURELY DUE TO
1648: C              ROUNDING ERRORS.
1649: C A(M+2,N+2)  NUMBER OF SIMPLEX ITERATIONS PERFORMED.
1650:       IMPLICIT REAL*8(A-H,O-Z)
1651: C
1652:       DIMENSION  A(:,:),X(:),E(:),B(:)
1653:       LOGICAL  STAGE,TEST
1654: C     LS IS INTEGER ARRAY OF SIZE M USED FOR WORKSPACE.
1655: C     IN ORIGINAL ACM 478 SOURCE THIS WAS PASSED AS S
1656:       ALLOCATABLE LS(:)
1657:       ALLOCATE (LS(M))
1658:       DATA  BIG/1.0D200/
1659: C
1660:       ZERO = 0.0D0
1661:       ONE = 1.0D0
1662:       TWO = 2.0D0
1663:       M1 = M+1
1664:       N1 = N+1
1665:       M2 = M+2
1666:       N2 = N+2
1667:       DO 10 J = 1,N
1668:         A(M2,J) = J
1669:         X(J) = ZERO
1670:  10   CONTINUE
1671:       DO 40 I = 1,M
1672:         A(I,N2) = N+I
1673:         A(I,N1) = B(I)
1674:         IF(B(I).GE.ZERO) GO TO 30
1675:         DO 20 J = 1,N2
1676:           A(I,J) = -A(I,J)
1677:  20     CONTINUE
1678:  30     E(I) = ZERO
1679:  40   CONTINUE
1680: C
```

```
1681:       DO 60 J = 1,N1
1682:         SUM = ZERO
1683:         DO 50 I = 1,M
1684:           SUM = SUM+A(I,J)
1685:  50     CONTINUE
1686:         A(M1,J) = SUM
1687:  60   CONTINUE
1688: C
1689:       STAGE = .TRUE.
1690:       KOUNT = 0
1691:       KR = 1
1692:       KL = 1
1693:  70   AMAX = -ONE
1694:       DO 80 J = KR,N
1695:         IF(DABS(A(M2,J)).GT.N) GO TO 80
1696:         D = DABS(A(M1,J))
1697:         IF(D.LE.AMAX) GO TO 80
1698:         AMAX = D
1699:         IN = J
1700:  80   CONTINUE
1701:       IF(A(M1,IN).GE.ZERO) GO TO 100
1702:       DO 90 I = 1,M2
1703:         A(I,IN) = -A(I,IN)
1704:  90   CONTINUE
1705: C
1706:  100  K = 0
1707:       DO 110 I = KL,M
1708:         D = A(I,IN)
1709:         IF(D.LE.TOLER) GO TO 110
1710:         K = K+1
1711:         B(K) = A(I,N1)/D
1712:         LS(K) = I
1713:         TEST = .TRUE.
1714:  110  CONTINUE
1715:  120  IF(K.GT.0) GO TO 130
1716:       TEST = .FALSE.
1717:       GO TO 150
1718:  130  AMIN = BIG
1719:       DO 140 I = 1,K
1720:         IF(B(I).GE.AMIN) GO TO 140
1721:         J = I
1722:         AMIN = B(I)
1723:         IOUT = LS(I)
1724:  140  CONTINUE
1725:       B(J) = B(K)
1726:       LS(J) = LS(K)
1727:       K = K-1
1728: C
1729:  150  IF(TEST.OR..NOT.STAGE) GO TO 170
1730:       DO 160 I = 1,M2
1731:         D = A(I,KR)
1732:         A(I,KR) = A(I,IN)
1733:         A(I,IN) = D
1734:  160  CONTINUE
1735:       KR = KR+1
1736:       GO TO 260
```

```
1737:  170  IF(TEST) GO TO 180
1738:       A(M2,N1) = TWO
1739:       GO TO 350
1740:  180  PIVOT = A(IOUT,IN)
1741:       IF(A(M1,IN)-PIVOT-PIVOT.LE.TOLER) GO TO 200
1742:       DO 190 J = KR,N1
1743:         D = A(IOUT,J)
1744:         A(M1,J) = A(M1,J)-D-D
1745:         A(IOUT,J) = -D
1746:  190  CONTINUE
1747:       A(IOUT,N2) = -A(IOUT,N2)
1748:       GO TO 120
1749: C
1750:  200  DO 210 J = KR,N1
1751:         IF(J.EQ.IN) GO TO 210
1752:         A(IOUT,J) = A(IOUT,J)/PIVOT
1753:  210  CONTINUE
1754:       DO 220 J = KR,N1
1755:         IF(J.EQ.IN) GO TO 220
1756:         CALL SPES1LAD1CL(A(1,J),A(1,IN),A(IOUT,J),M1,IOUT)
1757:  220  CONTINUE
1758:       DO 240 I = 1,M1
1759:         IF(I.EQ.IOUT) GO TO 240
1760:         A(I,IN) = -A(I,IN)/PIVOT
1761:  240  CONTINUE
1762:       A(IOUT,IN) = ONE/PIVOT
1763:       D = A(IOUT,N2)
1764:       A(IOUT,N2) = A(M2,IN)
1765:       A(M2,IN) = D
1766:       KOUNT = KOUNT+1
1767:       IF(.NOT.STAGE) GO TO 270
1768: C
1769:       KL = KL+1
1770:       DO 250 J = KR,N2
1771:         D = A(IOUT,J)
1772:         A(IOUT,J) = A(KOUNT,J)
1773:         A(KOUNT,J) = D
1774:  250  CONTINUE
1775:  260  IF(KOUNT+KR.NE.N1) GO TO 70
1776: C
1777:       STAGE = .FALSE.
1778: C
1779:  270  AMAX = -BIG
1780:       DO 290 J = KR,N
1781:         D = A(M1,J)
1782:         IF(D.GE.ZERO) GO TO 280
1783:         IF(D.GT.-TWO) GO TO 290
1784:         D = -D-TWO
1785:  280  IF(D.LE.AMAX) GO TO 290
1786:       AMAX = D
1787:       IN = J
1788:  290  CONTINUE
1789:       IF(AMAX.LE.TOLER) GO TO 310
1790:       IF(A(M1,IN).GT.ZERO) GO TO 100
1791:       DO 300 I = 1,M2
1792:         A(I,IN) = -A(I,IN)
```

```
1793:  300  CONTINUE
1794:       A(M1,IN) = A(M1,IN)-TWO
1795:       GO TO 100
1796: C
1797:  310  L = KL-1
1798:       DO 330 I = 1,L
1799:         IF(A(I,N1).GE.ZERO) GO TO 330
1800:         DO 320 J = KR,N2
1801:           A(I,J) = -A(I,J)
1802:  320    CONTINUE
1803:  330  CONTINUE
1804:       A(M2,N1) = ZERO
1805:       IF(KR.NE.1) GO TO 350
1806:       DO 340 J = 1,N
1807:         D = DABS(A(M1,J))
1808:         IF(D.LE.TOLER.OR.TWO-D.LE.TOLER) GO TO 350
1809:  340  CONTINUE
1810:       A(M2,N1) = ONE
1811:  350  DO 380 I = 1,M
1812:         K = A(I,N2)
1813:         D = A(I,N1)
1814:         IF(K.GT.0) GO TO 360
1815:         K = -K
1816:         D = -D
1817:  360  IF(I.GE.KL) GO TO 370
1818:       X(K) = D
1819:       GO TO 380
1820:  370  K = K-N
1821:       E(K) = D
1822:  380  CONTINUE
1823:       A(M2,N2) = KOUNT
1824:       A(M1,N2) = N1-KR
1825:       SUM = ZERO
1826:       DO 390 I = KL,M
1827:         SUM = SUM+A(I,N1)
1828:  390  CONTINUE
1829:       A(M1,N1) = SUM
1830:       DEALLOCATE (LS)
1831:       RETURN
1832:       END
1833: C
1834: C
1835:       SUBROUTINE SPES1LAD1CL(V1,V2,AMLT,M1,IOUT)
1836: C     VERSION 20030809 JMH Created
1837: C THIS SUBROUTINE WILL ASSIGN VALUES INTO V1 FROM
1838: C THE COMPUTED VALUE OF V1(I)-V2(I)*AMLT
1839: C IF THE I INTEGER DOES NOT EQUAL IOUT
1840:       IMPLICIT REAL*8(A-H,O-Z)
1841: C
1842:       DIMENSION  V1(M1),V2(M1)
1843: C
1844:       DO 10 I = 1,M1
1845:         IF(I.EQ.IOUT) GO TO 10
1846:         V1(I) = V1(I)-V2(I)*AMLT
1847:   10  CONTINUE
1848:       RETURN
```

```
1849:      END
1850: C
1851: C
1852:      SUBROUTINE SPES1LAD1AL(KG,Y,YH,DELTA,EDEL,VAR,GAM,T,RHO,PROB)
1853: C     VERSION 20030809 JMH Created
1854: C MULTIVARIATE RANDOMIZED BLOCK PERMUTATION (MRBP) PROCEDURES
1855: C THIS SUBROUTINE COLLECTS TWO VECTORS IN ONE ARRAY FOR CALLING
1856: C PES1LAD1CA
1857:      IMPLICIT REAL*8(A-H,O-Z)
1858: C
1859:      DIMENSION  Y(:),YH(:)
1860:      ALLOCATABLE DATA(:,:)
1861:      INTERFACE
1862:        SUBROUTINE SPES1LAD1CA(V,KG,DATA,DELTA,EDEL,VAR,GAM,T,RHO,PROB)
1863:          IMPLICIT REAL*8(A-H,O-Z)
1864:          DIMENSION DATA(:,:)
1865:        END SUBROUTINE SPES1LAD1CA
1866:      END INTERFACE
1867:      ALLOCATE (DATA(KG,2))
1868: C
1869:      V = 1.0D0
1870:      DO 20 I = 1,KG
1871:        DATA(I,1) = Y(I)
1872:        DATA(I,2) = YH(I)
1873:   20 CONTINUE
1874:      CALL SPES1LAD1CA(V,KG,DATA,DELTA,EDEL,VAR,GAM,T,RHO,PROB)
1875:      DEALLOCATE (DATA)
1876:      RETURN
1877:      END
1878: C
1879: C
1880:      SUBROUTINE SPES1LAD1CA(V,KG,DATA,DELTA,EDEL,VAR,GAM,T,RHO,PROB)
1881: C     VERSION 20030809 JMH Created
1882: C CALCULATION OF THE MRBP
1883:      IMPLICIT REAL*8(A-H,O-Z)
1884: C
1885:      DIMENSION DATA(:,:),
1886:     1  SIJ(2,2),SIJ2(2,2),SIJ3(2,2),
1887:     2  UIJ(2,2),TIJ2(2,2),TIJ3(2,2),VI(2,2)
1888: C
1889:      ALLOCATABLE D(:,:),SJ(:,:,:),SJ2(:,:,:),SJ3(:,:,:)
1890:      ALLOCATABLE UJ(:,:,:)
1891:      ALLOCATE (D(KG*2,KG*2),SJ(KG,2,2),SJ2(KG,2,2),SJ3(KG,2,2))
1892:      ALLOCATE (UJ(KG,2,2))
1893: C
1894:      ONE=1.0D0
1895:      ZERO=0.0D0
1896:      K2G=KG*2
1897:      DO 4 I=1,K2G
1898:        DO 3 J=1,K2G
1899:          D(I,J)=ZERO
1900:    3   CONTINUE
1901:    4 CONTINUE
1902:      DO 10 I=1,KG
1903:        DO 9 J=1,2
1904:          DO 8 K=I,KG
```

```
1905:            LO=1
1906:            IF(I.EQ.K) LO=J
1907:            DO 7 L=LO,2
1908:              IJ=2*(I-1)+J
1909:              KL=2*(K-1)+L
1910:              D(IJ,KL)=D(IJ,KL)+(DATA(I,J)-DATA(K,L))**2
1911:              D(IJ,KL)=D(IJ,KL)**(V/2)
1912:              D(KL,IJ)=D(IJ,KL)
1913:   7        CONTINUE
1914:   8      CONTINUE
1915:   9    CONTINUE
1916:  10 CONTINUE
1917:     DO 30 IS=1,2
1918:       DO 29 IR=1,2
1919:         IF(IR.EQ.IS) GO TO 29
1920:         SIJ(IR,IS)=ZERO
1921:         SIJ2(IR,IS)=ZERO
1922:         SIJ3(IR,IS)=ZERO
1923:         DO 28 I=1,KG
1924:           SJ(I,IR,IS)=ZERO
1925:           SJ2(I,IR,IS)=ZERO
1926:           SJ3(I,IR,IS)=ZERO
1927:           DO 27 J=1,KG
1928:             IRR=(I-1)*2+IR
1929:             JSS=(J-1)*2+IS
1930:             SJ(I,IR,IS)=SJ(I,IR,IS)+D(IRR,JSS)
1931:             SJ2(I,IR,IS)=SJ2(I,IR,IS)+D(IRR,JSS)**2
1932:             SJ3(I,IR,IS)=SJ3(I,IR,IS)+D(IRR,JSS)**3
1933:  27         CONTINUE
1934:           SIJ(IR,IS)=SIJ(IR,IS)+SJ(I,IR,IS)
1935:           SIJ2(IR,IS)=SIJ2(IR,IS)+SJ2(I,IR,IS)
1936:           SIJ3(IR,IS)=SIJ3(IR,IS)+SJ3(I,IR,IS)
1937:  28       CONTINUE
1938:  29     CONTINUE
1939:  30 CONTINUE
1940:     TIJ2(1,2)=ZERO
1941:     TIJ3(1,2)=ZERO
1942:     VI(1,2)=ZERO
1943:     UIJ(1,2)=ZERO
1944:     DO 100 I=1,KG
1945:       TIJ2(1,2)=TIJ2(1,2)+SJ(I,1,2)**2+
1946:    1    SJ(I,2,1)**2
1947:       TIJ3(1,2)=TIJ3(1,2)+SJ(I,1,2)**3+
1948:    1    SJ(I,2,1)**3
1949:       VI(1,2)=VI(1,2)+SJ(I,1,2)*SJ2(I,1,2)+
1950:    1    SJ(I,2,1)*SJ2(I,2,1)
1951:       UJ(I,1,2)=ZERO
1952:       DO 90 J=1,KG
1953:         IRR=2*(I-1)+1
1954:         JSS=2*(J-1)+2
1955:         UJ(I,1,2)=UJ(I,1,2)+
1956:    1      D(IRR,JSS)*SJ(I,1,2)*SJ(J,2,1)
1957:  90    CONTINUE
1958:       UIJ(1,2)=UIJ(1,2)+UJ(I,1,2)
1959: 100 CONTINUE
1960:     T1=ZERO
```

```
1961:      EDEL=ZERO
1962:      VAR=ZERO
1963:      IF(KG.LE.2) GO TO 150
1964:      T1=T1+(SIJ(1,2)**3)*4-SIJ(1,2)*TIJ2(1,2)*KG*6+
1965:     1  UIJ(1,2)*KG*KG*6+TIJ3(1,2)*KG*KG*2+
1966:     2  SIJ(1,2)*SIJ2(1,2)*KG*KG*3-
1967:     3  VI(1,2)*KG*KG*KG*3+SIJ3(1,2)*KG*KG*KG*KG
1968:   150 EDEL=EDEL+SIJ(1,2)
1969:      VAR=VAR+SIJ(1,2)*SIJ(1,2)-TIJ2(1,2)*KG+SIJ2(1,2)*KG*KG
1970:      IF(KG.LE.2) GO TO 200
1971:      T1=T1/(KG-2)
1972:   200 C1=ONE/KG/KG
1973:      EDEL=C1*EDEL
1974:      VAR=VAR*C1*C1/(KG-1)
1975:      GAM=C1*C1*C1*T1/(KG-1)
1976:      GAM=GAM/DSQRT(VAR**3)
1977:      T=(DELTA-EDEL)/DSQRT(VAR)
1978:      RHO=ONE-(DELTA/EDEL)
1979:      CALL SPES1LAD1PV(T,GAM,PROB)
1980:      DEALLOCATE(D,SJ,SJ2,SJ3,UJ)
1981:      RETURN
1982:      END
1983: C
1984:      SUBROUTINE SPES1LAD1DT(NCASES,DATA,PROB)
1985: C     VERSION 20030809 JMH Created
1986:      IMPLICIT REAL*8(A-H,O-Z)
1987: C
1988: C        SUBROUTINE SPES1LAD1DT CALCULATES THE DISTANCES BETWEEN THE DATA
1989: C        POINTS, PLUS THE PARAMETERS WHICH ARE DIRECTLY DEPENDENT
1990: C        ON THE DISTANCES.
1991: C
1992:      DIMENSION  B(12),DATA(:),Q(6)
1993:      ALLOCATABLE DC(:,:),DEL(:,:)
1994:      ALLOCATE (DC(3,NCASES),DEL(NCASES,NCASES))
1995: C
1996:      ZERO = 0.0D0
1997:      ONE = 1.0D0
1998:      DBAR = ZERO
1999:      DO 1400 I = 1,NCASES
2000:        DO 1300 J = 1,NCASES
2001:          DEL(I,J) = DABS(DATA(I)-DATA(J))
2002:   1300   CONTINUE
2003:   1400 CONTINUE
2004:      DO 1600 I = 1,NCASES-1
2005:        DBAR = DBAR+DEL(I,I+1)
2006:   1600 CONTINUE
2007:      DBAR = DBAR/(NCASES-1)
2008:      DO 2150 I = 1,NCASES
2009:        DO 2140 K = 1,3
2010:          DC(K,I) = ZERO
2011:          DO 2130 J = 1,NCASES
2012:            DC(K,I) = DC(K,I)+DEL(I,J)**K
2013:   2130     CONTINUE
2014:   2140   CONTINUE
2015:   2150 CONTINUE
2016:      DO 2160 I = 1,12
```

```
2017:        B(I) = ZERO
2018:  2160 CONTINUE
2019:      DO 2170 I = 1,NCASES
2020:        B(1) = B(1)+DC(1,I)
2021:        B(2) = B(2)+DC(2,I)
2022:        B(3) = B(3)+DC(1,I)**2
2023:        B(5) = B(5)+DC(3,I)
2024:        B(6) = B(6)+DC(1,I)*DC(2,I)
2025:        B(10) = B(10)+DC(1,I)**3
2026:  2170 CONTINUE
2027:      DO 2190 I = 1,NCASES-1
2028:        DO 2180 J = I+1,NCASES
2029:          B(9) = B(9)+DC(1,I)*DC(1,J)*DEL(I,J)
2030:  2180    CONTINUE
2031:  2190 CONTINUE
2032:      B(9) = B(9)*2
2033:      DO 2240 I = 1,NCASES-2
2034:        DO 2230 J = I+1,NCASES-1
2035:          DO 2220 K = J+1,NCASES
2036:            SUM = DEL(I,J)*DEL(I,K)*DEL(J,K)
2037:            B(7) = B(7)+SUM
2038:  2220      CONTINUE
2039:  2230    CONTINUE
2040:  2240 CONTINUE
2041:      B(7) = B(7)*6
2042:      B(4) = B(1)**2
2043:      B(8) = B(1)*B(2)
2044:      B(11) = B(1)*B(3)
2045:      B(12) = B(1)**3
2046:      A1 = B(1)
2047:      A2 = B(2)
2048:      A3 = B(3)-B(2)
2049:      A4 = B(4)-A2*2-A3*4
2050:      A5 = B(5)
2051:      A6 = B(6)-B(5)
2052:      A7 = B(7)
2053:      A8 = B(8)+(B(5)-B(6)*2)*2
2054:      A9 = B(9)+B(5)-B(6)*2-B(7)
2055:      A10 = B(10)+B(5)*2-B(6)*3
2056:      A11 = B(11)-(B(10)-B(7)+(B(5)+B(9))*2)*2+B(6)*10-B(8)
2057:      A12 = B(12)-(A5+(A7+A10+(A6+A9)*3)*2)*4-(A8+A11*2)*6
2058:      Q(1) = NCASES
2059:      DO 2280 I = 2,6
2060:        Q(I) = Q(I-1)*(NCASES-I+1)
2061:  2280 CONTINUE
2062:      D1 = A1/Q(2)
2063:      D2 = A2/Q(2)
2064:      D3 = A3/Q(3)
2065:      D4 = A4/Q(4)
2066:      D5 = A5/Q(2)
2067:      D6 = A6/Q(3)
2068:      D7 = A7/Q(3)
2069:      D8 = A8/Q(4)
2070:      D9 = A9/Q(4)
2071:      D10 = A10/Q(4)
2072:      D11 = A11/Q(5)
```

```
      D12 = A12/Q(6)
      M = NCASES
      VAR = (D2*(M-1)-D3*(M-2)*2+D4*(M-3))*M*(M-2)/Q(2)**2
      STDEV = DSQRT(VAR)
      T = (DBAR-D1)/STDEV
      GAM1 = (D5*(M-1)*(M-2)-D6*(M-2)*(M-2)*6+D7*(M-2)*4+
     1      (D8*3+D10*4)*(M-2)*(M-3)+(D9-D11*2)*(M-3)*(M-4)*6+
     2      D12*(M-3)*(M-5)*4)*M*(M-4)/(STDEV*Q(2))**3
      CALL SPES1LAD1PV(T,GAM1,PROB)
      DEALLOCATE (DC,DEL)
      RETURN
      END
C
C
      SUBROUTINE SPES1LAD1PV(T,GAM,PROB)
C     VERSION 20030809 JMH Created
C
C       SUBROUTINE SPES1LAD1PV CALCULATES THE PROBABILITY THAT A VALUE OF
C       T IS LESS THAN OR EQUAL TO THE OBSERVED VALUE OF T.
C
      DOUBLE PRECISION  T,GAM,PROB,ZERO,ONE,PI,A,B,C,D,EE,F,G,H,R,
     +  U,W,X,Y,E1,E2,E3,E4,E5,G1,G2,G3,H0,H1,H1N,H2,H3
C
      ZERO = 0.0D0
      ONE = 1.0D0
      PI = DATAN(ONE)*4
      IF(DABS(GAM).LT.0.01D0) GO TO 50
      R = ONE*2/DABS(GAM)
      D = R*R
      F = R*R+10
      DO 10 I = 1,9
        D = D*(R*R+I)
   10 CONTINUE
      U = (F*2-1)*DLOG(F)/2-F+DLOG(PI*2)/2-DLOG(D)+ONE/(F*12)
     +      -ONE/(F*F*F*360)
      A = R*R-1
      B = R*R*(DLOG(R)-1)-U
      G1 = 0.0375D0
      G2 = 0.075D0
      G3 = 0.0125D0
      H1 = ZERO
      H2 = ZERO
      W = -1.99D0/GAM
      IF(GAM.LT.ZERO) GO TO 30
      IF(T.LT.W) THEN
        PROB = ZERO
        RETURN
      END IF
      X = T
      Y = T+15
      DO 20 I = 1,199
        G = A*DLOG(R+X+G1*(2*I-1))-R*(X+G1*(2*I-1))+B
        H1 = H1+EE(G)
        G = A*DLOG(R+X+G2*I)-R*(X+G2*I)+B
        H2 = H2+EE(G)
   20 CONTINUE
```

```
2129:       G = A*DLOG(R+X+G1*399)-R*(X+G1*399)+B
2130:       H1N = EE(G)
2131:       G = A*DLOG(R+X)-R*X+B
2132:       H0 = EE(G)
2133:       G = A*DLOG(R+Y)-R*Y+B
2134:       H3 = EE(G)
2135:       PROB = ONE-G3*(H0+(H1+H1N)*4+H2*2+H3)
2136:       RETURN
2137:    30 IF(T.GT.W) THEN
2138:         PROB = ONE
2139:         RETURN
2140:       END IF
2141:       X = T-15
2142:       Y = T
2143:       DO 40 I = 1,199
2144:         G = A*DLOG(R-X-G1*(2*I-1))+R*(X+G1*(2*I-1))+B
2145:         H1 = H1+EE(G)
2146:         G = A*DLOG(R-X-G2*I)+R*(X+G2*I)+B
2147:         H2 = H2+EE(G)
2148:    40 CONTINUE
2149:       G = A*DLOG(R-X-G1*399)+R*(X+G1*399)+B
2150:       H1N = EE(G)
2151:       G = A*DLOG(R-X)+R*X+B
2152:       H0 = EE(G)
2153:       G = A*DLOG(R-Y)+R*Y+B
2154:       H3 = EE(G)
2155:       PROB = G3*(H0+(H1+H1N)*4+H2*2+H3)
2156:       RETURN
2157:    50 PROB = ZERO
2158:       G = -T*T/2
2159:       IF(G.LT.-738.0D0) GO TO 70
2160:       A = 4.2D0
2161:       U = DABS(T)
2162:       IF(U.GT.A) GO TO 60
2163:       E1 = 0.31938153D0
2164:       E2 = -0.356563782D0
2165:       E3 = 1.781477937D0
2166:       E4 = -1.821255978D0
2167:       E5 = 1.330274429D0
2168:       H = 0.2316419D0
2169:       W = ONE/(H*U+1)
2170:       PROB = ((((E5*W+E4)*W+E3)*W+E2)*W+E1)*W*EE(G)/DSQRT(PI*2)
2171:       GO TO 70
2172:    60 B = 2.2D0
2173:       C = 4.9D0
2174:       PROB = EE(G)*(ONE/U-ONE/U**3+B/U**C)/DSQRT(PI*2)
2175:    70 IF(T.GT.ZERO) PROB = ONE-PROB
2176:       RETURN
2177:       END
2178: C
2179: C
2180:       DOUBLE PRECISION FUNCTION EE(G)
2181: C
2182:       DOUBLE PRECISION G
2183: C
2184:       EE = 0.0D0
```

```
2185:      IF(G.GE.-738.0D0) EE = DEXP(G)
2186:      RETURN
2187:      END
2188: C
2189:      SUBROUTINE SPES1LAD1EM(M,N,X,Y,R,EM)
2190: C     VERSION 20030913 JMH Created
2191: C
2192: C   PES1LAD1EM CALCULATES THE GENERALIZED EXPECTATION MAXIMIZATION (EM)
2193: C
2194: C   NUMBER OF OBSERVATIONS     M       (ND IN MF2K)
2195: C   NUMBER OF PARAMETERS       N       (NPE IN MF2K)
2196: C   LAD REGRESSION COEFFICIENT X(N)
2197: C   OBSERVATIONS              Y(M)
2198: C   RESULTS MODEL             R(M,N)
2199: C   MODEL WEIGHTS             W(M,M)
2200: C   LAD ESTIMATE OF RESIDUAL   E(M)
2201: C    e.g.                     X(1) =  -43.348484
2202: C                            X(2) =   1.1417310
2203: C                            E(M) =   Y(M)-43.34+1.141*R(M)
2204: C            T     -1   T
2205: C    EM = (R W R)  R  W E
2206: C       = (NxM)(MxM)(MxN)(NxM)(MxM)(Mx1)
2207: C       = (Nx1)
2208: C
2209: C   REFERENCE: Least absolute deviations estimation via the EM algorithm
2210: C            Robert F. Phillips, Statistics and Computing 12: 281-285
2211: C            2002
2212: C SUBROUTINE SPES1LAD1EM(M,N,X,Y,R,EM)
2213:      DOUBLE PRECISION  X,Y,R,EM,C,CI,CR,RT,E,W
2214:      DIMENSION  X(N),Y(M),R(M,N),EM(N)
2215:      ALLOCATABLE E(:),C(:,:),CI(:,:),RT(:,:),CR(:,:)
2216:      ALLOCATE (E(M),C(N,N),CI(N,N),RT(N,M),CR(M,N))
2217: C
2218: C   THE WEIGHT MATRIX WILL COME LATER
2219:      W = 1.0
2220: C
2221: C   CALCULATE THE LAD ESTIMATE OF RESIDUAL
2222: C
2223: C     DO 5 J = 1, N
2224: C       print *, 'X(',J,')=',X(J)
2225: C   5 CONTINUE
2226:      DO 20 J = 1, M
2227:        E(J)=X(1)*R(J,1)
2228:        DO 10 JN = 2, N
2229:          E(J)=E(J)+X(JN)*R(J,JN)
2230:    10  CONTINUE
2231:        E(J)=E(J)-Y(J)
2232: C       print *, 'E(',J,')=',E(J),'Y(',J,')=',Y(J)
2233:    20 CONTINUE
2234: C
2235: C   WEIGHT MATRIX
2236: C      W = WT(J)
2237: C      IF (W.LT.0.) THEN
2238: C        W = 1.E-20
2239: C      ENDIF
2240: C
```

```
2241: C  CALCULATE THE LAD C MATRIX WHERE
2242: C
2243: C           T
2244: C    RT = R
2245: C
2246:      CALL SPES1LAD1TRA(M,N,R,RT)
2247: C    DO 40 I = 1, M
2248: C      DO 30 J = 1, N
2249: C        print *, 'R(',I,',',J,')=',R(I,J)
2250: C  30   CONTINUE
2251: C  40 CONTINUE
2252: C    DO 60 J = 1, M
2253: C      DO 50 I = 1, N
2254: C        print *, 'RT(',I,',',J,')=',RT(I,J)
2255: C  50   CONTINUE
2256: C  60 CONTINUE
2257: C
2258: C           T
2259: C    C = R W R
2260: C
2261: C
2262: C  IF THE WEIGHT MATRIX IS GIVEN
2263: C
2264: C  MATRIX MULTIPLICATION RTW=RT*W       NxM=(NxM)*(M*M)
2265: C    CALL SPES1LAD1MUL(RTW,N,M,CI,M,M,W)
2266: C  MATRIX MULTIPLICATION C=RTW*R        NxN=(NxM)*(M*N)
2267: C    CALL SPES1LAD1MUL(C,N,M,RTW,M,N,R)
2268: C
2269: C  FOR NOW ASSUME THE WEIGHT MATRIX IS 1.0 ON THE DIAGONAL
2270: C
2271: C  MATRIX MULTIPLICATION C=RT*R         NxN=(NxM)*(M*N)
2272:      CALL SPES1LAD1MUL(C,N,M,RT,M,N,R)
2273: C  MATRIX MULTIPLICATION C=RT*R
2274: C    DO 80 I = 1, N
2275: C      DO 70 J = 1, N
2276: C        print *, 'C(',I,',',J,')=',C(I,J)
2277: C  70   CONTINUE
2278: C  80 CONTINUE
2279: C
2280:      CALL SPES1LAD1INV(C,CI,N,IRET)
2281: C                   -1
2282: C  MATRIX INVERSION CI=C
2283: C    DO 100 I = 1, N
2284: C      DO 90 J = 1, N
2285: C        print *, 'CI(',I,',',J,')=',CI(I,J)
2286: C  90   CONTINUE
2287: C 100 CONTINUE
2288: C
2289: C             T     -1 T
2290: C    EM = (R W R) R W E
2291: C        -------
2292: C          CI       NxN MATRIX
2293: C        -----------
2294: C           CR      NxM MATRIX
2295: C        ---------------
2296: C             EM     Nx1 MATRIX
```

```
2297: C
2298: C  MATRIX MULTIPLICATION CR=CI*RT
2299:       CALL SPES1LAD1MUL(CR,N,N,CI,N,M,RT)
2300: C  MATRIX MULTIPLICATION EM=CR*E
2301:       CALL SPES1LAD1MUL(EM,N,M,CR,M,1,E)
2302:       DEALLOCATE (E,C,CI,RT,CR)
2303:       RETURN
2304:       END
2305: C
2306:       SUBROUTINE SPES1LAD1MUL(C,MA,NA,A,MB,NB,B)
2307: C     VERSION 20030913 JMH Created
2308: C  MATRIX MULTIPLICATION C=A*B
2309:       DOUBLE PRECISION  A,B,C,ZERO,DTMP
2310:       DIMENSION  C(MA,NB),A(MA,NA),B(MB,NB)
2311: C  MATRICES DO NOT HAVE BE SQUARE BUT INNER DIMENSIONS MUST BE EQUAL
2312:       IF(NA.NE.MB) GOTO 7
2313:       DATA ZERO /0.0/
2314:       DO 2 I = 1, MA
2315:         DO 1 J = 1, NB
2316:           C(I, J) = ZERO
2317:     1     CONTINUE
2318:     2   CONTINUE
2319: C
2320:       DO 6 I = 1, MA
2321:         DO 5 J = 1, NB
2322:           DTMP=ZERO
2323:           DO 4 K = 1, NA
2324:             DTMP = DTMP + A(I, K) * B(K, J)
2325:     4     CONTINUE
2326:           C(I,J) = DTMP
2327:     5   CONTINUE
2328:     6 CONTINUE
2329:     7 CONTINUE
2330:       RETURN
2331:       END
2332:       SUBROUTINE SPES1LAD1TRA(M,N,A,AT)
2333: C     VERSION 20030913 JMH Created
2334: C                                          T
2335: C  MATRIX TRANSPOSITION AT=A
2336: C
2337:       DIMENSION A(M,N),AT(N,M)
2338:       DOUBLE PRECISION A,AT
2339:       DO 2 I = 1, M
2340:         DO 1 J = 1, N
2341:           AT(J,I)=A(I,J)
2342:     1   CONTINUE
2343:     2 CONTINUE
2344:       RETURN
2345:       END
2346: C
2347:       SUBROUTINE SPES1LAD1INV(A,AI,N,IRET)
2348: C     VERSION 20030913 JMH Created
2349: C
2350: C  ROUTINE TO INVERT A MATRIX BY GAUSSIAN ELIMINATION
2351: C  INPUT MATRIX IS A, OUTPUT MATRIX IS AI, BOTH ARE NxN
2352: C  CHECK IRET FOR RETURN VALUE: IRET=0 THEN NON-SINGULAR
```

```
2353: C                    IRET=1 THEN SINGULAR
2354: C                          -1
2355: C  MATRIX INVERSION AI=A
2356: C
2357:       DIMENSION A(N,N),AI(N,N)
2358:       DOUBLE PRECISION A,AI,D
2359:       ALLOCATABLE D(:,:)
2360:       ALLOCATE(D(N,N*2))
2361:       IRET=1
2362:       N2 = 2*N
2363:       DO 1 I = 1,N
2364:         DO 2 J = 1,N
2365:           D(I,J) = A(I,J)
2366:           D(I,N+J) = 0.
2367:  2      CONTINUE
2368:         D(I,N+I) = 1.
2369:  1    CONTINUE
2370: C  DO THE REDUCTION
2371:       DO 3 I = 1,N
2372:         ALPHA = D(I,I)
2373:         IF(ALPHA .EQ. 0.) GO TO 300
2374:         DO 4 J = 1,N2
2375:           D(I,J) = D(I,J)/ALPHA
2376:  4      CONTINUE
2377:         DO 5 K = 1,N
2378:           IF((K-I).EQ.0) GO TO 5
2379:           BETA = D(K,I)
2380:           DO 6 J = 1,N2
2381:             D(K,J) = D(K,J) - BETA*D(I,J)
2382:  6        CONTINUE
2383:  5      CONTINUE
2384:  3    CONTINUE
2385: C  COPY RESULT INTO OUTPUT MATRIX
2386:       DO 7 I = 1,N
2387:         DO 8 J = 1,N
2388:           AI(I,J) = D(I,J+N)
2389:  8      CONTINUE
2390:  7    CONTINUE
2391: C IF IT MADE IT HERE THEN IT IS NON-SINGULAR SO SET IRET=0
2392:       IRET=0
2393:  300  CONTINUE
2394:       DEALLOCATE (D)
2395:       RETURN
2396:       END
```

10.4 - Listing of Model and Observed Heads in Meters (msl)

Name	UTM X	UTM Y	5 meters	6 meters	7 meters	observed
WC1616	440569.6	4645790	334.37	338.86	322.83	322.06
WC1606	440569.6	4645791	334.37	338.86	322.83	322.19
WC2214	440960.7	4645860	333.43	337.72	322.29	318.80
WC2206	440960.9	4645859	333.43	337.72	322.29	318.83
WC2416	441362.5	4645520	331.53	335.49	321.18	320.00
WC2409	441363.7	4645520	331.52	335.48	321.18	319.52
WC3624	441671.1	4647533	333.92	337.99	322.56	318.89
WC3613	441672.1	4647533	333.91	337.98	322.55	318.75
WC3608	441673.2	4647533	333.91	337.98	322.55	318.80
WC3803	441759.9	4649015	338.76	343.65	325.22	313.97
WC3817	441760.2	4649016	338.76	343.66	325.22	314.36
WC3806	441760.3	4649015	338.76	343.65	325.22	313.80
WC3706	441764.0	4648331	336.40	340.88	323.92	314.38
WC3703	441764.0	4648330	336.40	340.87	323.92	314.73
WC3717	441764.1	4648332	336.40	340.88	323.92	315.04
WC1005	441765.8	4646442	331.03	334.72	320.92	316.46
WC1110	441765.8	4646185	330.67	334.35	320.71	317.06
WC1106	441765.8	4646183	330.67	334.35	320.71	317.52
WC1115	441765.8	4646186	330.67	334.35	320.71	317.13
WC1010	441765.8	4646443	331.03	334.72	320.92	316.40
WC1017	441765.9	4646444	331.03	334.72	320.92	316.51
WC0514	441795.1	4645930	330.16	333.80	320.41	316.99
WC0506	441796.3	4645930	330.16	333.79	320.41	314.36
WC1210	442084.8	4645516	328.22	331.58	319.28	315.87
WC1216	442084.9	4645516	328.22	331.58	319.28	316.25
WC0814	442111.4	4645108	327.61	330.95	318.91	318.15
WC0819	442111.4	4645108	327.62	330.95	318.91	318.15
WC0706	442153.4	4646219	328.51	331.74	319.49	316.59
WC0709	442153.5	4646220	328.51	331.74	319.49	316.61
WC0721	442153.6	4646222	328.51	331.74	319.49	316.69
WC0714	442153.6	4646221	328.51	331.74	319.49	316.62
WC0904	442154.4	4646450	328.75	331.96	319.64	316.06
WC0907	442154.4	4646449	328.75	331.96	319.64	316.18
WC0913	442154.5	4646448	328.74	331.96	319.64	316.52
WC0409	442580.2	4645837	325.73	328.53	317.87	316.97
WC0416	442580.4	4645836	325.73	328.53	317.87	316.98
WC1809	442633.0	4645109	325.53	328.47	317.68	316.04
WC2507	442636.2	4645314	325.46	328.35	317.66	315.40
WC1416	443269.6	4645094	322.34	324.72	315.78	316.05
WC1411	443271.3	4645094	322.33	324.71	315.77	316.04
WC1406	443272.7	4645094	322.32	324.70	315.77	315.99
WC0609	443458.0	4645712	319.94	321.76	314.44	313.82
WC0619	443458.3	4645711	319.94	321.76	314.44	313.88
WC1306	443802.9	4645087	319.35	321.26	313.95	316.04
WC1324	443804.3	4645087	319.34	321.25	313.95	315.84
WC1318	443805.4	4645087	319.33	321.25	313.95	316.12
WC1704	444786.3	4646226	309.91	310.43	308.24	307.76

10.4 - Listing of Model and Observed Heads in Meters (msl)

Name	UTM X	UTM Y	5 meters	6 meters	7 meters	observed
WC1706	444786.6	4646226	309.90	310.43	308.24	307.70
WC1709	444787.5	4646225	309.90	310.42	308.23	307.78
WC1714	444788.0	4646224	309.89	310.42	308.23	307.71
WC3204	444791.8	4646197	309.83	310.35	308.17	307.83
WC3210	444792.3	4646196	309.82	310.34	308.17	307.75
WC3216	444792.9	4646195	309.82	310.34	308.17	307.43
WC3306	444797.3	4645889	309.20	309.69	307.65	307.51
WC3316	444798.3	4645889	309.19	309.68	307.64	307.59
WC2919	444889.6	4646279	309.26	309.71	307.83	308.83
WC2909	444890.2	4646279	309.25	309.71	307.83	308.88
WC2903	444891.0	4646280	309.25	309.70	307.83	309.01
WC1914	445086.4	4646039	307.00	307.24	306.27	307.00
WC1909	445087.3	4646039	307.00	307.24	306.27	306.89
WC1906	445088.6	4646039	306.98	307.23	306.26	306.91
WC1904	445090.6	4646039	306.97	307.21	306.25	306.88
WC3015	445106.4	4646278	307.99	308.34	306.98	310.75
WC3008	445107.1	4646278	307.99	308.33	306.98	310.66
WC3004	445107.9	4646277	307.98	308.33	306.98	310.66
WC3517	445239.1	4645888	305.30	305.36	305.16	306.40
WC3506	445276.7	4645884	305.31	305.36	305.16	306.58
WC3503	445277.5	4645884	305.31	305.36	305.16	306.54
WC3416	445286.3	4645976	305.72	305.83	305.43	306.93
WC3430	445287.9	4645977	305.72	305.83	305.44	307.07
WC3405	445288.7	4645976	305.72	305.83	305.44	306.78
WC3114	445424.0	4646274	307.14	307.42	306.40	310.20
WC3106	445425.6	4646274	307.14	307.42	306.40	310.26
WC1505	445439.7	4645976	305.88	306.01	305.56	306.92
WC1509	445440.6	4645976	305.88	306.01	305.56	306.33
WC1516	445441.5	4645976	305.88	306.01	305.56	306.74
WC0204	445911.5	4645914	305.47	305.60	305.14	305.19
WC0206	445912.0	4645913	305.46	305.59	305.13	305.15
WC0209	445912.8	4645912	305.45	305.58	305.12	305.16
WC0214	445913.3	4645911	305.44	305.57	305.11	305.04
WC0221	445914.0	4645910	305.05	305.05	305.10	305.11
WC0304	445976.4	4645879	304.82	304.94	304.51	305.74
WC0306	445977.9	4645879	304.81	304.93	304.50	305.88
WC0311	445978.7	4645879	304.81	304.93	304.49	305.69
WC2807	449722.4	4643616	294.41	294.57	294.25	299.23
WC2811	449723.3	4643616	294.40	294.57	294.25	299.61
WC2818	449724.1	4643616	294.78	294.78	294.78	299.02
WC2715	449767.5	4643757	294.39	294.56	294.23	296.55
WC2720	449768.4	4643758	294.39	294.56	294.23	296.35
WC2730	449769.3	4643758	294.23	294.23	294.23	296.50

10.5 - Listing of Model and Observed Head Differences in Meters

Name	UTM X	UTM Y	delta5	delta6	delta7
WC1616	440569.6	4645790	12.31	16.80	0.77
WC1606	440569.6	4645791	12.18	16.67	0.64
WC2214	440960.7	4645860	14.63	18.92	3.49
WC2206	440960.9	4645859	14.60	18.89	3.46
WC2416	441362.5	4645520	11.53	15.49	1.18
WC2409	441363.7	4645520	12.00	15.96	1.66
WC3624	441671.1	4647533	15.03	19.10	3.67
WC3613	441672.1	4647533	15.16	19.23	3.80
WC3608	441673.2	4647533	15.11	19.18	3.75
WC3803	441759.9	4649015	24.79	29.68	11.25
WC3817	441760.2	4649016	24.40	29.30	10.86
WC3806	441760.3	4649015	24.96	29.85	11.42
WC3706	441764.0	4648331	22.02	26.50	9.54
WC3703	441764.0	4648330	21.67	26.14	9.19
WC3717	441764.1	4648332	21.36	25.84	8.88
WC1005	441765.8	4646442	14.57	18.26	4.46
WC1110	441765.8	4646185	13.61	17.29	3.65
WC1106	441765.8	4646183	13.15	16.83	3.19
WC1115	441765.8	4646186	13.54	17.22	3.58
WC1010	441765.8	4646443	14.63	18.32	4.52
WC1017	441765.9	4646444	14.52	18.21	4.41
WC0514	441795.1	4645930	13.17	16.81	3.42
WC0506	441796.3	4645930	15.80	19.43	6.05
WC1210	442084.8	4645516	12.35	15.71	3.41
WC1216	442084.9	4645516	11.97	15.33	3.03
WC0814	442111.4	4645108	9.46	12.80	0.76
WC0819	442111.4	4645108	9.47	12.80	0.76
WC0706	442153.4	4646219	11.92	15.15	2.90
WC0709	442153.5	4646220	11.90	15.13	2.88
WC0721	442153.6	4646222	11.82	15.05	2.80
WC0714	442153.6	4646221	11.89	15.12	2.87
WC0904	442154.4	4646450	12.69	15.90	3.58
WC0907	442154.4	4646449	12.57	15.78	3.46
WC0913	442154.5	4646448	12.22	15.44	3.12
WC0409	442580.2	4645837	8.76	11.56	0.90
WC0416	442580.4	4645836	8.75	11.55	0.89
WC1809	442633.0	4645109	9.49	12.43	1.64
WC2507	442636.2	4645314	10.06	12.95	2.26
WC1416	443269.6	4645094	6.29	8.67	-0.27
WC1411	443271.3	4645094	6.29	8.67	-0.27
WC1406	443272.7	4645094	6.33	8.71	-0.22
WC0609	443458.0	4645712	6.12	7.94	0.62
WC0619	443458.3	4645711	6.06	7.88	0.56
WC1306	443802.9	4645087	3.31	5.22	-2.09
WC1324	443804.3	4645087	3.50	5.41	-1.89
WC1318	443805.4	4645087	3.21	5.13	-2.17
WC1704	444786.3	4646226	2.15	2.67	0.48

10.5 - Listing of Model and Observed Head Differences in Meters

Name	UTM X	UTM Y	delta5	delta6	delta7
WC1706	444786.6	4646226	2.20	2.73	0.54
WC1709	444787.5	4646225	2.12	2.64	0.45
WC1714	444788.0	4646224	2.18	2.71	0.52
WC3204	444791.8	4646197	2.00	2.52	0.34
WC3210	444792.3	4646196	2.07	2.59	0.42
WC3216	444792.9	4646195	2.39	2.91	0.74
WC3306	444797.3	4645889	1.69	2.18	0.14
WC3316	444798.3	4645889	1.60	2.09	0.05
WC2919	444889.6	4646279	0.43	0.88	-1.00
WC2909	444890.2	4646279	0.37	0.83	-1.05
WC2903	444891.0	4646280	0.24	0.69	-1.18
WC1914	445086.4	4646039	0.00	0.24	-0.73
WC1909	445087.3	4646039	0.11	0.35	-0.62
WC1906	445088.6	4646039	0.07	0.32	-0.65
WC1904	445090.6	4646039	0.09	0.33	-0.63
WC3015	445106.4	4646278	-2.76	-2.41	-3.77
WC3008	445107.1	4646278	-2.67	-2.33	-3.68
WC3004	445107.9	4646277	-2.68	-2.33	-3.68
WC3517	445239.1	4645888	-1.10	-1.04	-1.24
WC3506	445276.7	4645884	-1.27	-1.22	-1.42
WC3503	445277.5	4645884	-1.23	-1.18	-1.38
WC3416	445286.3	4645976	-1.21	-1.10	-1.50
WC3430	445287.9	4645977	-1.35	-1.24	-1.63
WC3405	445288.7	4645976	-1.06	-0.95	-1.34
WC3114	445424.0	4646274	-3.06	-2.78	-3.80
WC3106	445425.6	4646274	-3.12	-2.84	-3.86
WC1505	445439.7	4645976	-1.04	-0.91	-1.36
WC1509	445440.6	4645976	-0.45	-0.32	-0.77
WC1516	445441.5	4645976	-0.86	-0.73	-1.18
WC0204	445911.5	4645914	0.28	0.41	-0.05
WC0206	445912.0	4645913	0.31	0.44	-0.02
WC0209	445912.8	4645912	0.29	0.42	-0.04
WC0214	445913.3	4645911	0.40	0.53	0.07
WC0221	445914.0	4645910	-0.06	-0.06	-0.01
WC0304	445976.4	4645879	-0.92	-0.80	-1.23
WC0306	445977.9	4645879	-1.07	-0.95	-1.38
WC0311	445978.7	4645879	-0.88	-0.76	-1.20
WC2807	449722.4	4643616	-4.82	-4.66	-4.98
WC2811	449723.3	4643616	-5.21	-5.04	-5.36
WC2818	449724.1	4643616	-4.24	-4.24	-4.24
WC2715	449767.5	4643757	-2.16	-1.99	-2.32
WC2720	449768.4	4643758	-1.96	-1.79	-2.12
WC2730	449769.3	4643758	-2.27	-2.27	-2.27

10.5 - Listing of Model and Observed Head Differences in Meters

Figure 10.5.1
Differences Between the Model and Observed Heads for a Top Depth = 5m Using LAD-EM

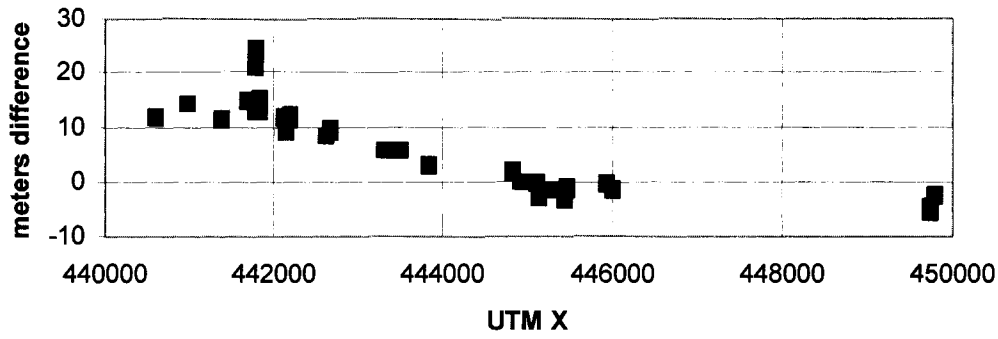


Figure 10.5.2
Differences Between the Model and Observed Heads for a Top Depth = 6m Using LAD-EM

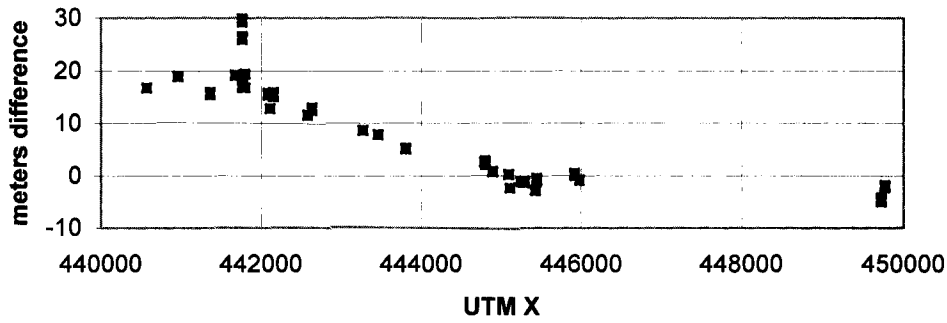
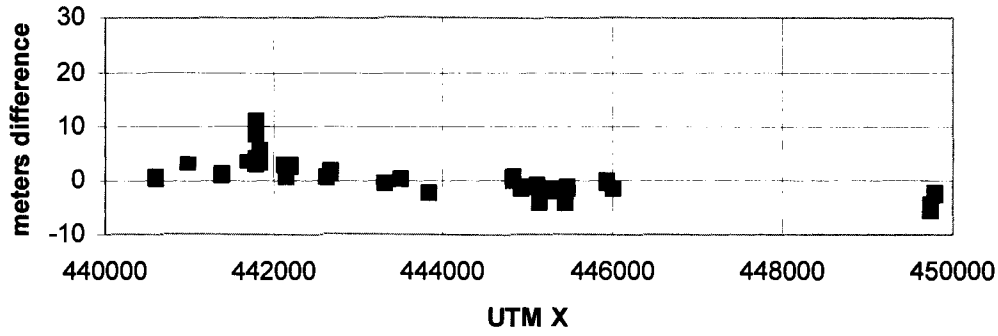


Figure 10.5.3
Differences Between the Model and Observed Heads for a Top Depth = 7m Using LAD-EM



10.6 - Listing of the Exp2Shp Utility

```
 1: /* Contributions by this author are shown in BOLD
 2:  * $Id: Exp2Shp.c,v 1.8 2000/06/12 13:23:35 jhudd Exp $
 3:  *
 4:  *   Copyright (C) 1999 by Jan-Oliver Wagner <jan@intevation.de>
 5:  *   Revised by JHuddleston to read ArgusONE Export files , e.g.
 6:  *   ## Name:
 7:  *   ## Icon:0
 8:  *   # Points Count  Value
 9:  *   6  251.
10:  *   # X pos  Y pos
11:  *   451948.814592741  4641468.11396978
12:  *   451634.812160449  4641683.50008198
13:  *   451361.607936195  4641820.47646611
14:  *   451357.23750419  4641741.82097803
15:  *   451751.296256557  4641455.23307377
16:  *   451948.814592741  4641468.11396978
17:  *
18:  *   This program is free software; you can redistribute it and/or
19:  *   modify it under the terms of the GNU General Public License
20:  *   as published by the Free Software Foundation; either version 2
21:  *   of the License, or (at your option) any later version.
22:  *
23:  *   This program is distributed in the hope that it will be useful,
24:  *   but WITHOUT ANY WARRANTY; without even the implied warranty of
25:  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
26:  *   GNU General Public License for more details.
27:  *
28:  *   You should have received a copy of the GNU GPL
29:  *   along with this program; if not, write to the Free Software
30:  *   Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
31:  *
32:  */
33: #include <stdio.h>
34: #include <stdlib.h>
35: #include <string.h>
36: #include <shapefil.h>
37:
38: #define VERSION "0.3.2"
39:
40: #ifdef DEBUG
41: #define DEBUG_OUT(str) fprintf(stderr,"Exp2Shp debug: " str)
42: #define DEBUG_OUT1(str,v) fprintf(stderr,"Exp2Shp debug: " str,v)
43: #define DEBUG_OUT2(str,v,w) fprintf(stderr,"Exp2Shp debug: " str,v,w)
44: #define DEBUG_OUT3(str,v,w,x) fprintf(stderr,"Exp2Shp debug:
    "str,v,w,x)
45: #else
46: #define DEBUG_OUT(str)
47: #define DEBUG_OUT1(str,v)
48: #define DEBUG_OUT2(str,v,w)
49: #define DEBUG_OUT3(str,v,w,x)
50: #endif
51:
52: /* Error codes for exit() routine: */
53: #define  ERR_USAGE  1
54: #define ERR_TYPE  2
```

```
 55: #define ERR_FORMAT  3
 56: #define ERR_OBJECTTYPE  4
 57: #define ERR_ALLOC  5
 58:
 59: #define ERR_DBFCREATE  10
 60: #define ERR_DBFADDFIELD  11
 61: #define ERR_DBFOPEN  12
 62: #define  ERR_DBFWRITEINTEGERATTRIBUTE  13
 63:
 64: #define ERR_SHPOPEN  20
 65:
 66: /* Object Type codes used in main(): */
 67: #define OBJECTTYPE_NONE    0
 68: #define OBJECTTYPE_POINT  1
 69: #define OBJECTTYPE_LINE    2
 70: #define OBJECTTYPE_POLYGON  3
 71: #define OBJECTTYPE_ARCS    4
 72:
 73: /* minimum number of coordinates allocated blockwise */
 74: #define COORDS_BLOCKSIZE  100
 75:
 76: /* maximum length for read strings,
 77:  * if input lines with more characters appear,
 78:  * errors are likely to occur */
 79: #define STR_BUFFER_SIZE    300
 80:
 81: #ifdef USE_STRICMP
 82: #define CASE_INSENSITIVE_STR_CMP  stricmp
 83: #else
 84: #define CASE_INSENSITIVE_STR_CMP  strcasecmp
 85: #endif
 86: double atof();
 87: int getline(FILE *fp, char s[] )
 88: {  int c, i;
 89:
 90:   i=0;
 91:   while ( (c=fgetc(fp))!=EOF && c!='\n' )
 92:     s[i++]=c;
 93:   if(i>0 && s[i-1] == '\r') s[i-1]='\0';
 94:   s[i]='\0';
 95:   return c;
 96: }
 97:
 98: void print_version(FILE *file)
 99: {
100:   fprintf(file,"Exp2Shp version " VERSION "\n");
101: #ifdef DEBUG
102:   fprintf(file,"compiled with option: DEBUG\n");
103: #endif
104: }
105:
106: static DBFHandle LaunchDbf (  const char *fname ) {
107:   DBFHandle  hDBF;
108:   char    dbffname[STR_BUFFER_SIZE];
109:   char    fieldname[STR_BUFFER_SIZE];
110:
```

```
111:    sprintf(dbffname, "%s.dbf", fname);
112:    sprintf(fieldname, "%s-id", fname);
113:
114:    hDBF = DBFCreate( dbffname );
115:    if( hDBF == NULL ) {
116:      fprintf(stderr, "DBFCreate(%s) failed.\n", fname );
117:      exit(ERR_DBFCREATE);
118:    }
119:
120:    if (DBFAddField( hDBF, fieldname, FTInteger, 11, 0 ) == -1) {
121:   fprintf(stderr,"DBFAddField(hDBF,%s,FTInteger,11,0)failed.\n",
122:      fieldname);    exit(ERR_DBFADDFIELD);
123:    }
124:
125:    DBFClose( hDBF );
126:
127:    hDBF = DBFOpen( dbffname, "r+b" );
128:    if( hDBF == NULL ) {
129:      fprintf(stderr, "DBFOpen(%s,\"r+b\") failed.\n", dbffname );
130:      exit(ERR_DBFOPEN);
131:    }
132:
133:    return hDBF;
134: }
135:
136: static SHPHandle LaunchShp(  const char *fname,
137:          int ObjectType ) {
138:    SHPHandle  hSHP;
139:    SHPObject  *psShape;
140:    char    shpfname[STR_BUFFER_SIZE];
141:
142:    sprintf(shpfname, "%s.shp", fname);
143:
144:    switch (ObjectType) {
145:      case OBJECTTYPE_POINT:
146:        hSHP = SHPCreate( shpfname, SHPT_POINT );
147:        break;
148:      case OBJECTTYPE_ARCS:
149:        hSHP = SHPCreate( shpfname, SHPT_ARC );
150:        break;
151:      case OBJECTTYPE_LINE:
152:        hSHP = SHPCreate( shpfname, SHPT_ARC );
153:        break;
154:      case OBJECTTYPE_POLYGON:
155:        hSHP = SHPCreate( shpfname, SHPT_POLYGON );
156:        break;
157:      default:
158:        fprintf(stderr, "internal error: "
159:          "unknown ObjectType=%d\n", ObjectType);
160:        exit(ERR_OBJECTTYPE);
161:    }
162:
163:    if( hSHP == NULL ) {
164:      fprintf(stderr, "SHPOpen(%s, shape_type) failed.\n", shpfname );
165:      exit(ERR_SHPOPEN);
166:    }
```

```
167:
168:   return hSHP;
169: }
170:
171: static void WriteDbf (  DBFHandle hDBF,
172:       int rec,
173:       int id ) {
174:   if (! DBFWriteIntegerAttribute(hDBF, rec, 0, id)) {
175:     fprintf(stderr, "DBFWriteIntegerAttribute(hDBFs,%d,1,%d)
176:     failed.\n", rec, id );  exit(ERR_DBFWRITEINTEGERATTRIBUTE);
177:   }
178: }
179:
180: static void WritePoint(  SHPHandle hSHP,
181:       int rec,
182:       double x,
183:       double y ) {
184:   SHPObject  *psShape;
185:
186:   psShape = SHPCreateObject( SHPT_POINT, rec, 0, NULL, NULL,
187:                                 1, &x, &y, NULL, NULL );
188:   SHPWriteObject( hSHP, -1, psShape );
189:   SHPDestroyObject( psShape );
190: }
191:
192: static void WriteLine(  SHPHandle hSHP,
193:       int rec,
194:       int coords,
195:       double * x,
196:       double * y ) {
197:   SHPObject  *psShape;
198:
199:   psShape = SHPCreateObject( SHPT_ARC, rec, 0, NULL, NULL,
200:     coords, x, y, NULL, NULL );
201:   SHPWriteObject( hSHP, -1, psShape );
202:   SHPDestroyObject( psShape );
203: }
204:
205: static void WritePolygon(  SHPHandle hSHP,
206:         int rec,
207:         int coords,
208:         double * x,
209:         double * y,
210:         int nparts,
211:         int * partstarts) {
212:   SHPObject  *psShape;
213:
214: DEBUG_OUT1("WritePolygon: rec = %d\n", rec);
215: DEBUG_OUT1("WritePolygon: nparts = %d\n", nparts);
216: DEBUG_OUT1("WritePolygon: coords = %d\n", coords);
217:
218:   psShape = SHPCreateObject( SHPT_POLYGON, rec, nparts, partstarts,
219:     NULL,coords, x, y, NULL, NULL );
220:   SHPWriteObject( hSHP, -1, psShape );
221:   SHPDestroyObject( psShape );
222: }
```

```
223:
224: /* read from fp and generate point shapefile to hDBF/hSHP */
225: static void GeneratePoints (  FILE *fp,
226:         DBFHandle hDBF,
227:         SHPHandle hSHP ) {
228:   char linebuf[STR_BUFFER_SIZE];/*buffer for reading from file*/
229:   int id;      /* ID of point */
230:   double x, y;    /* coordinates of point */
231:   char * str;    /* tmp variable needed for assertions */
232:   char * dstr;    /* tmp variable needed to find out substrings */
233:   int rec = 0;    /* Counter for records */
234:   char *p,*q;
235:   int chr;
236: /*
237: ## Name:
238: ## Icon:0
239: # Points Count  Value
240: 2  310.
241: # X pos  Y pos
242: 438048.75  4673090
243: */
244:   while (chr = getline(fp, linebuf) != EOF) {
245:     if (linebuf[0] == '#' || linebuf[0] == '\0') continue;
246:     q=&linebuf[0]; while (*q != '\t') q++; *q++=0;
247:     p=q; while(*q != 0) q++; if(*(q-1) == '.') *(q-1)=0;
248:     id = atoi(p);
249:     chr = getline(fp, linebuf); if(chr == EOF) break;
250:
251:     /* get the data */
252:     while (chr = getline(fp, linebuf) != EOF) {
253:       if (linebuf[0] == '#' || linebuf[0] == '\0') break;
254:       q=p=&linebuf[0]; while (*q != '\t') q++; *q++=0;
255:       x = atof(p); y=atof(q);
256:       DEBUG_OUT3("id=%d, x=%f, y=%f\n", id, x, y);
257:       WriteDbf(hDBF, rec, id);
258:       WritePoint(hSHP, rec, x, y);
259:       rec ++;
260:     }
261:     if(chr == EOF) break;
262:   }
263: }
264:
265: /* read from fp and generate line/arc shapefile to hDBF/hSHP */
266: static void GenerateLines (  FILE *fp,
267:         DBFHandle hDBF,
268:         SHPHandle hSHP ) {
269:   char linebuf[STR_BUFFER_SIZE];/*buffer for reading from file*/
270:   int id;      /* ID of point */
271:   double  * x = NULL,
272:     * y = NULL;  /* coordinates arrays */
273:   int vector_size = 0;  /* current size of the vectors x and y */
274:   char * str;    /* tmp variable needed for assertions */
275:   char * dstr;    /* tmp variable needed to find out substrings */
276:   int rec = 0;    /* Counter for records */
277:   int coord = 0;    /* Counter for coordinates */
278:   char *p, *q;
```

```
279:   int chr;
280:
281: /*
282: ## Name:
283: ## Icon:0
284: # Points Count  Value
285: 2  310.
286: # X pos  Y pos
287: 438048.75  4673090
288: */
289:   while (chr = getline(fp, linebuf) != EOF) {
290:     if (linebuf[0] == '#' || linebuf[0] == '\0') continue;
291:     q=&linebuf[0]; while (*q != '\t') q++; *q++=0;
292:     p=q; while(*q != 0) q++; if(*(q-1) == '.') *(q-1)=0;
293:     id = atoi(p);
294:     DEBUG_OUT1("id=%d\n", id);
295:     chr = getline(fp, linebuf); if(chr == EOF) break;
296:
297:     coord = 0;
298:
299:     /* loop coordinates of line 'id' */
300:     while (chr = getline(fp, linebuf) != EOF) {
301:       if (linebuf[0] == '#' || linebuf[0] == '\0') break;
302:       /* allocate coordinate vectors if to small */
303:       if (vector_size <= coord) {
304:         vector_size += COORDS_BLOCKSIZE;
305:         x = realloc(x, vector_size * sizeof(double));
306:         y = realloc(y, vector_size * sizeof(double));
307:         if (x == NULL || y == NULL) {
308:           fprintf(stderr, "memory allocation failed\n");
309:           exit(ERR_ALLOC);
310:         }
311:       }
312:       q=p=&linebuf[0]; while (*q != '\t') q++; *q++=0;
313:       x[coord] = atof(p); y[coord]=atof(q);
314:       DEBUG_OUT2("x=%f, y=%f\n", x[coord], y[coord]);
315:       coord ++;
316:     }
317:     WriteDbf(hDBF, rec, id);
318:     WriteLine(hSHP, rec, coord, x, y);
319:     rec ++;
320:     if(chr == EOF) break;
321:   }
322:   free(x);
323:   free(y);
324: }
325:
326: /* read from fp and generate line/arc shapefile to hDBF/hSHP */
327: static void GenerateArcs (  FILE *fp,
328:         DBFHandle hDBF,
329:         SHPHandle hSHP ) {
330:   char linebuf[STR_BUFFER_SIZE];/*buffer for reading from file*/
331:   int id;       /* ID of point */
332:   double  * x = NULL,
333:     * y = NULL;  /* coordinates arrays */
334:   int vector_size = 0;  /* current size of the vectors x and y */
```

```
335:   char * str;    /* tmp variable needed for assertions */
336:   char * dstr;    /* tmp variable needed to find out substrings */
337:   int rec = 0;    /* Counter for records */
338:   int coord = 0;    /* Counter for coordinates */
339:   char *p, *q;
340:   int chr;
341:
342:   if (vector_size <= coord) {
343:     vector_size += COORDS_BLOCKSIZE;
344:     x = realloc(x, vector_size * sizeof(double));
345:     y = realloc(y, vector_size * sizeof(double));
346:     if (x == NULL || y == NULL) {
347:       fprintf(stderr, "memory allocation failed\n");
348:       exit(ERR_ALLOC);
349:     }
350:   }
351: /*
352: ## Name:
353: ## Icon:0
354: # Points Count  Value
355: 2  310.
356: # X pos  Y pos
357: 438048.75  4673090
358: */
359:   while (chr = getline(fp, linebuf) != EOF) {
360:     if (linebuf[0] == '#' || linebuf[0] == '\0') continue;
361:     q=&linebuf[0]; while (*q != '\t') q++; *q++=0;
362:     p=q; while(*q != 0) q++; if(*(q-1) == '.') *(q-1)=0;
363:     id = atoi(p);
364:     DEBUG_OUT1("id=%d\n", id);
365:     chr = getline(fp, linebuf); if(chr == EOF) break;
366:
367:     chr = getline(fp, linebuf); if(chr == EOF) break;
368:     if (linebuf[0] == '#' || linebuf[0] == '\0') break;
369:     q=p=&linebuf[0]; while (*q != '\t') q++; *q++=0;
370:     x[0] = atof(p); y[0]=atof(q);
371:     DEBUG_OUT2("x=%f, y=%f\n", x[0], y[0]);
372:     coord=2;
373:     while (chr = getline(fp, linebuf) != EOF) {
374:       if (linebuf[0] == '#' || linebuf[0] == '\0') break;
375:       q=p=&linebuf[0]; while (*q != '\t') q++; *q++=0;
376:       x[1] = atof(p); y[1]=atof(q);
377:       DEBUG_OUT2("x=%f, y=%f\n", x[1], y[1]);
378:       WriteDbf(hDBF, rec, id);
379:       WriteLine(hSHP, rec, coord, x, y);
380:       x[0]=x[1]; y[0]=y[1];
381:       rec ++;
382:     }
383:     if(chr == EOF) break;
384:   }
385:   free(x);
386:   free(y);
387: }
388:
389: /* read from fp and generate polgon shapefile to hDBF/hSHP */
390: static void GeneratePolygons (  FILE *fp,
```

```
391:          DBFHandle hDBF,
392:          SHPHandle hSHP ) {
393:    char linebuf[STR_BUFFER_SIZE];/*buffer for reading from file*/
394:    int id = -1;       /* ID of polygon */
395:    double  * x = NULL,
396:      * y = NULL;  /* coordinates arrays */
397:    int vector_size = 0;  /* current size of the vectors x and y */
398:    int nparts = 0; /* number of parts */
399:    int * partstarts = NULL; /* new parts start in x[],y[] */
400:    char * str;     /* tmp variable needed for assertions */
401:    char * dstr;     /* tmp variable needed to find out substrings */
402:    int rec = 0;     /* Counter for records */
403:    int coord = 0;     /* Counter for coordinates */
404:    char *p, *q;
405:    int chr;
406:
407: /*
408: ## Name:
409: ## Icon:0
410: # Points Count  Value
411: 2  310.
412: # X pos  Y pos
413: 438048.75  4673090
414: */
415:    while (chr = getline(fp, linebuf) != EOF) {
416:      if (linebuf[0] == '#' || linebuf[0] == '\0') continue;
417:      q=&linebuf[0]; while (*q != '\t') q++; *q++=0;
418:      p=q; while(*q != 0) q++; if(*(q-1) == '.') *(q-1)=0;
419:      id = atoi(p);
420:      DEBUG_OUT1("id=%d\n", id);
421:      coord = 0;
422:      nparts = 0;
423:      chr = getline(fp, linebuf); if(chr == EOF) break;
424:
425:      partstarts = realloc(partstarts, sizeof(int) * (nparts+1));
426:      if (partstarts == NULL) {
427:        fprintf(stderr, "memory allocation failed\n");
428:        exit(ERR_ALLOC);
429:      }
430:
431:      while (chr = getline(fp, linebuf) != EOF) {
432:        if (linebuf[0] == '#' || linebuf[0] == '\0') break;
433:        /* allocate coordinate vectors if to small */
434:        if (vector_size <= coord) {
435:          vector_size += COORDS_BLOCKSIZE;
436:          x = realloc(x, vector_size * sizeof(double));
437:          y = realloc(y, vector_size * sizeof(double));
438:          if (x == NULL || y == NULL) {
439:            fprintf(stderr, "memory allocation failed\n");
440:            exit(ERR_ALLOC);
441:          }
442:        }
443:        q=p=&linebuf[0]; while (*q != '\t') q++; *q++=0;
444:        x[coord] = atof(p); y[coord]=atof(q);
445:        DEBUG_OUT2("x=%f, y=%f\n", x[coord], y[coord]);
446:        coord ++;
```

```
447:      }
448:      partstarts[nparts] = coord;
449:      DEBUG_OUT1("newpart at %d\n", coord);
450:      WriteDbf(hDBF, rec, id);
451:      if (partstarts) partstarts[0] = 0;
452:      WritePolygon(hSHP,rec,coord,x,y,(nparts>0 ? nparts+1 : 0),
453:      partstarts); free(partstarts); partstarts = NULL;
454:      rec ++;
455:      if(chr == EOF) break;
456:    }
457:    free(partstarts);
458:    free(x);
459:    free(y);
460: }
461:
462: int main(  int argc,
463:      char ** argv ) {
464:    DBFHandle hDBF;     /* handle for dBase file */
465:    SHPHandle hSHP;     /* handle for shape files .shx and .shp */
466:    int ObjectType = OBJECTTYPE_NONE;
467:
468:    if (argc != 3) {
469:      print_version(stderr);
470:      fprintf(stderr, "usage: %s outfile type < infile\n", argv[0]);
471:      fprintf(stderr, "\treads stdin and creates outfile.shp, "
472:        "outfile.shx and outfile.dbf\n"
473:        "\ttype must be one of these: points arcs lines polygons\n"
474:        "\tinfile must be in ArgusONE 'Exp' export format\n");
475:      fprintf(stderr, "points are single x,y coordinates,\n");
476:      fprintf(stderr, "arcs are split into separate polylines,\n");
477:      fprintf(stderr, "lines are groups of polylines,\n");
478:      fprintf(stderr, "polygons cover an entire area.\n");
479:      exit(ERR_USAGE);
480:    }
481:
482:    /* determine Object Type: */
483:    if (strcmp(argv[2], "points") == 0) ObjectType = OBJECTTYPE_POINT;
484:    if (strcmp(argv[2], "arcs") == 0) ObjectType = OBJECTTYPE_ARCS;
485:    if (strcmp(argv[2], "lines") == 0) ObjectType = OBJECTTYPE_LINE;
486:    if (strcmp(argv[2], "polygons") == 0) ObjectType =
487:     OBJECTTYPE_POLYGON; if (ObjectType == OBJECTTYPE_NONE) {
488:      fprintf(stderr, "type '%s' unknown, use one of these: "
489:        "points arcs lines polygons.\n", argv[2]);
490:      fprintf(stderr, "where: points are single x,y coordinates,\n");
491:      fprintf(stderr, "arcs are split into separate polylines,\n");
492:      fprintf(stderr, "lines are groups of polylines,\n");
493:      fprintf(stderr, "polygons cover an entire area.\n");
494:      exit(ERR_TYPE);
495:    }
496:
497:    DEBUG_OUT1("outfile=%s\n", argv[1]);
498:    DEBUG_OUT1("type=%s\n", argv[2]);
499:
500:    /* Open and prepare output files */
501:    hDBF = LaunchDbf(argv[1]);
502:    hSHP = LaunchShp(argv[1], ObjectType);
```

```
503:
504:    /* Call generate function */
505:    switch (ObjectType) {
506:      case OBJECTTYPE_POINT:
507:        GeneratePoints(stdin, hDBF, hSHP);
508:        break;
509:      case OBJECTTYPE_ARCS:
510:        GenerateArcs(stdin, hDBF, hSHP);
511:        break;
512:      case OBJECTTYPE_LINE:
513:        GenerateLines(stdin, hDBF, hSHP);
514:        break;
515:      case OBJECTTYPE_POLYGON:
516:        GeneratePolygons(stdin, hDBF, hSHP);
517:        break;
518:      default:
519:        fprintf(stderr, "internal error: "
520:          "unknown ObjectType=%d\n", ObjectType);
521:        exit(ERR_OBJECTTYPE);
522:    }
523:
524:    /* Finish output files */
525:    DBFClose( hDBF );
526:    SHPClose( hSHP );
527:
528:    /* success */
529:    exit(0);
530: }
```

10.6.2 Listing of shapefil.h

```
  1: #ifndef _SHAPEFILE_H_INCLUDED
  2: #define _SHAPEFILE_H_INCLUDED
  3:
  4: /******************************************************
  5:  * $Id: shapefil.h,v 1.27 2003/04/21 18:30:37 warmerda Exp $
  6:  *
  7:  * Project:  Shapelib
  8:  * Purpose:  Primary include file for Shapelib.
  9:  * Author:   Frank Warmerdam, warmerdam@pobox.com
 10:  *
 11:  ******************************************************
 12:  * Copyright (c) 1999, Frank Warmerdam
 13:  *
 14:  * This software is available under the following "MIT Style"
 15:  * or at the option of the licensee under the LGPL.
 16:  * This option is discussed in more detail in shapelib.html.
 17:  *
 18:  * --
 19:  *
 20:  * Permission is hereby granted, free of charge, to any person
    obtaining a
 21:  * copy of this software and associated documentation files
    ("Software"),
 22:  * to deal in the Software without restriction, including w/o
    limitation
```

10.6 - Listing of the Exp2Shp Utility

```
23:  * the rights to use, copy, modify, merge, publish, distribute,
   sublicense,
24:  * and/or sell copies of the Software, and to permit persons to whom
   the
25:  * Software is furnished to do so, subject to the following
   conditions:
26:  *
27:  * The above copyright notice and this permission notice shall be
   included
28:  * in all copies or substantial portions of the Software.
29:  *
30:  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
   EXPRESS
31:  * IMPLIED,INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
   MERCHANTABILITY,
32:  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
   SHALL
33:  * THE AUTHORS COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
   OTHER
34:  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
   ARISING
35:  * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
   OTHER
36:  * DEALINGS IN THE SOFTWARE.
37:  ******************************************************
38:  *
39:  * $Log: shapefil.h,v $
40:  * Revision 1.27  2003/04/21 18:30:37  warmerda
41:  * added header write/update public methods
42:  *
43:  * Revision 1.26  2002/09/29 00:00:08  warmerda
44:  * added FTLogical and logical attribute read/write calls
45:  *
46:  * Revision 1.25  2002/05/07 13:46:30  warmerda
47:  * added DBFWriteAttributeDirectly().
48:  *
49:  * Revision 1.24  2002/04/10 16:59:54  warmerda
50:  * added SHPRewindObject
51:  *
52:  * Revision 1.23  2002/01/15 14:36:07  warmerda
53:  * updated email address
54:  *
55:  * Revision 1.22  2002/01/15 14:32:00  warmerda
56:  * try to improve SHPAPI_CALL docs
57:  *
58:  * Revision 1.21  2001/11/01 16:29:55  warmerda
59:  * move pabyRec into SHPInfo for thread safety
60:  *
61:  * Revision 1.20  2001/07/20 13:06:02  warmerda
62:  * fixed SHPAPI attribute for SHPTreeFindLikelyShapes
63:  *
64:  * Revision 1.19  2001/05/31 19:20:13  warmerda
65:  * added DBFGetFieldIndex()
66:  *
67:  * Revision 1.18  2001/05/31 18:15:40  warmerda
68:  * Added support for NULL fields in DBF files
```

10.6 - Listing of the Exp2Shp Utility

```
 69:  *
 70:  * Revision 1.17  2001/05/23 13:36:52  warmerda
 71:  * added use of SHPAPI_CALL
 72:  *
 73:  * Revision 1.16  2000/09/25 14:15:59  warmerda
 74:  * added DBFGetNativeFieldType()
 75:  *
 76:  * Revision 1.15  2000/02/16 16:03:51  warmerda
 77:  * added null shape support
 78:  *
 79:  * Revision 1.14  1999/11/05 14:12:05  warmerda
 80:  * updated license terms
 81:  *
 82:  * Revision 1.13  1999/06/02 18:24:21  warmerda
 83:  * added trimming code
 84:  *
 85:  * Revision 1.12  1999/06/02 17:56:12  warmerda
 86:  * added quad'' subnode support for trees
 87:  *
 88:  * Revision 1.11  1999/05/18 19:11:11  warmerda
 89:  * Added example searching capability
 90:  *
 91:  * Revision 1.10  1999/05/18 17:49:38  warmerda
 92:  * added initial quadtree support
 93:  *
 94:  * Revision 1.9  1999/05/11 03:19:28  warmerda
 95:  * added new Tuple api, and improved extension handling
 96:  *
 97:  * Revision 1.8  1999/03/23 17:22:27  warmerda
 98:  * Added extern "C" protection for C++ users of shapefil.h.
 99:  *
100:  * Revision 1.7  1998/12/31 15:31:07  warmerda
101:  * TRIM_DBF_WHITESPACE and DISABLE_MULTIPATCH_MEASURE
102:  *
103:  * Revision 1.6  1998/12/03 15:48:15  warmerda
104:  * Added SHPCalculateExtents().
105:  *
106:  * Revision 1.5  1998/11/09 20:57:16  warmerda
107:  * Altered SHPGetInfo() call.
108:  *
109:  * Revision 1.4  1998/11/09 20:19:33  warmerda
110:  * Added 3D support, and use of SHPObject.
111:  *
112:  * Revision 1.3  1995/08/23 02:24:05  warmerda
113:  * Added support for reading bounds.
114:  *
115:  * Revision 1.2  1995/08/04  03:17:39  warmerda
116:  * Added header.
117:  *
118:  */
119:
120: #include <stdio.h>
121:
122: #ifdef USE_DBMALLOC
123: #include <dbmalloc.h>
124: #endif
```

```
125:
126: #ifdef __cplusplus
127: extern "C" {
128: #endif
129:
130: /************************************************************************/
131: /*                 Configuration options.                               */
132: /************************************************************************/
133:
134: /*---------------------------------------------------------------- */
135: /*Should the DBFReadStringAttribute() strip leading and           */
136: /*trailing white space?                                           */
137: /*---------------------------------------------------------------- */
138: #define TRIM_DBF_WHITESPACE
139:
140: /*---------------------------------------------------------------- */
141: /*Should we write measure values to the Multipatch object?        */
142: /*Reportedly ArcView crashes if we do write it, so for now it     */
143: /*is disabled.                                                    */
144: /*---------------------------------------------------------------- */
145: #define DISABLE_MULTIPATCH_MEASURE
146:
147: /*---------------------------------------------------------------- */
148: /*SHPAPI_CALL                                                     */
149: /*                                                               */
150: /*The following two macros are present to allow forcing          */
151: /*various calling conventions on the Shapelib API.               */
152: /*                                                               */
153: /*To force __stdcall conventions (needed to call Shapelib        */
154: /*from Visual Basic and/or Dephi I believe) the makefile could   */
155: /* be modified to define:                                        */
156: /*                                                               */
157: /* /DSHPAPI_CALL=__stdcall                                       */
158: /*                                                               */
159: /*If it is desired to force export of the Shapelib API without   */
160: /*      using the shapelib.def file, use the following definition*/
161: /*                                                               */
162: /*        /DSHAPELIB_DLLEXPORT                                   */
163: /*                                                               */
164: /*      To get both at once it will be necessary to hack this    */
165: /*      include file to define:                                  */
166: /*                                                               */
167: /*        #define SHPAPI_CALL __declspec(dllexport) __stdcall    */
168: /*        #define SHPAPI_CALL1 __declspec(dllexport) * __stdcall */
169: /*                                                               */
170: /*      The complexity of the situtation is partly caused by the */
171: /*      peculiar requirement of Visual C++ that __stdcall appear */
172: /*      after any "*"'s in the return value of a function while */
173: /*      __declspec(dllexport) must appear before them.          */
174: /* ---------------------------------------------------------- */
175:
176: #ifdef SHAPELIB_DLLEXPORT
177: #  define SHPAPI_CALL __declspec(dllexport)
178: #  define SHPAPI_CALL1(x)  __declspec(dllexport) x
179: #endif
180:
```

```
181: #ifndef SHPAPI_CALL
182: #  define SHPAPI_CALL
183: #endif
184:
185: #ifndef SHPAPI_CALL1
186: #  define SHPAPI_CALL1(x)      x SHPAPI_CALL
187: #endif
188:
189: /************************************************************/
190: /*                      SHP Support.                        */
191: /************************************************************/
192: typedef  struct
193: {
194:     FILE        *fpSHP;
195:     FILE  *fpSHX;
196:
197:     int    nShapeType;        /* SHPT_* */
198:
199:     int    nFileSize;        /* SHP file */
200:
201:     int         nRecords;
202:     int    nMaxRecords;
203:     int    *panRecOffset;
204:     int    *panRecSize;
205:
206:     double  adBoundsMin[4];
207:     double  adBoundsMax[4];
208:
209:     int    bUpdated;
210:
211:     unsigned char *pabyRec;
212:     int         nBufSize;
213: } SHPInfo;
214:
215: typedef SHPInfo * SHPHandle;
216:
217: /* ------------------------------------------------------------ */
218: /*      Shape types (nSHPType)                                  */
219: /* ------------------------------------------------------------ */
220: #define SHPT_NULL  0
221: #define SHPT_POINT  1
222: #define SHPT_ARC  3
223: #define SHPT_POLYGON  5
224: #define SHPT_MULTIPOINT  8
225: #define SHPT_POINTZ  11
226: #define SHPT_ARCZ  13
227: #define SHPT_POLYGONZ  15
228: #define SHPT_MULTIPOINTZ 18
229: #define SHPT_POINTM  21
230: #define SHPT_ARCM  23
231: #define SHPT_POLYGONM  25
232: #define SHPT_MULTIPOINTM 28
233: #define SHPT_MULTIPATCH 31
234:
235:
236: /* ------------------------------------------------------------ */
```

```
237: /*      Part types - everything but SHPT_MULTIPATCH just uses   */
238: /*      SHPP_RING.                                              */
239: /* -------------------------------------------------------------- */
240:
241: #define SHPP_TRISTRIP  0
242: #define SHPP_TRIFAN  1
243: #define SHPP_OUTERRING  2
244: #define SHPP_INNERRING  3
245: #define SHPP_FIRSTRING  4
246: #define SHPP_RING  5
247:
248: /* ---------------------------------------------------------------- */
249: /*      SHPObject - represents on shape (without attributes) read*/
250: /*      from the .shp file.                                         */
251: /* ---------------------------------------------------------------- */
252: typedef struct
253: {
254:     int    nSHPType;
255:
256:     int    nShapeId; /* -1 is unknown/unassigned */
257:
258:     int    nParts;
259:     int    *panPartStart;
260:     int    *panPartType;
261:
262:     int    nVertices;
263:     double  *padfX;
264:     double  *padfY;
265:     double  *padfZ;
266:     double  *padfM;
267:
268:     double  dfXMin;
269:     double  dfYMin;
270:     double  dfZMin;
271:     double  dfMMin;
272:
273:     double  dfXMax;
274:     double  dfYMax;
275:     double  dfZMax;
276:     double  dfMMax;
277: } SHPObject;
278:
279: /* -------------------------------------------------------------- */
280: /*      SHP API Prototypes                                      */
281: /* -------------------------------------------------------------- */
282: SHPHandle SHPAPI_CALL
283:       SHPOpen( const char * pszShapeFile, const char * pszAccess );
284: SHPHandle SHPAPI_CALL
285:       SHPCreate( const char * pszShapeFile, int nShapeType );
286: void SHPAPI_CALL
287: SHPGetInfo( SHPHandle hSHP, int * pnEntities, int * pnShapeType,
288:                   double * padfMinBound, double * padfMaxBound );
289:
290: SHPObject SHPAPI_CALL1(*)
291:       SHPReadObject( SHPHandle hSHP, int iShape );
292: int SHPAPI_CALL
```

```
293: SHPWriteObject( SHPHandle hSHP,int iShape,SHPObject *psObject);
294:
295: void SHPAPI_CALL
296:       SHPDestroyObject( SHPObject * psObject );
297: void SHPAPI_CALL
298:       SHPComputeExtents( SHPObject * psObject );
299: SHPObject SHPAPI_CALL1(*)
300:       SHPCreateObject( int nSHPType, int nShapeId,
301:          int nParts, int * panPartStart, int * panPartType,
302:          int nVertices, double * padfX, double * padfY,
303:          double * padfZ, double * padfM );
304: SHPObject SHPAPI_CALL1(*)
305:       SHPCreateSimpleObject( int nSHPType, int nVertices,
306:       double * padfX, double * padfY, double * padfZ );
307:
308: int SHPAPI_CALL
309:       SHPRewindObject( SHPHandle hSHP, SHPObject * psObject );
310:
311: void SHPAPI_CALL SHPClose( SHPHandle hSHP );
312: void SHPAPI_CALL SHPWriteHeader( SHPHandle hSHP );
313:
314: const char SHPAPI_CALL1(*)
315:       SHPTypeName( int nSHPType );
316: const char SHPAPI_CALL1(*)
317:       SHPPartTypeName( int nPartType );
318:
319: /* ------------------------------------------------ */
320: /*      Shape quadtree indexing API.                */
321: /* ------------------------------------------------ */
322:
323: /* this can be two or four for binary or quad tree */
324: #define MAX_SUBNODE  4
325:
326: typedef struct shape_tree_node
327: {
328:     /* region covered by this node */
329:     double  adfBoundsMin[4];
330:     double  adfBoundsMax[4];
331:
332:     /* list of shapes The papsShapeObj pointers
333:        or the whole list can be NULL */
334:     int    nShapeCount;
335:     int    *panShapeIds;
336:     SHPObject   **papsShapeObj;
337:
338:     int    nSubNodes;
339:     struct shape_tree_node *apsSubNode[MAX_SUBNODE];
340:
341: } SHPTreeNode;
342:
343: typedef struct
344: {
345:     SHPHandle   hSHP;
346:
347:     int    nMaxDepth;
348:     int    nDimension;
```

```
349:
350:     SHPTreeNode  *psRoot;
351: } SHPTree;
352:
353: SHPTree SHPAPI_CALL1(*)
354: SHPCreateTree( SHPHandle hSHP, int nDimension, int nMaxDepth,
355:                       double *padfBoundsMin, double *padfBoundsMax );
356: void    SHPAPI_CALL
357:       SHPDestroyTree( SHPTree * hTree );
358:
359: int  SHPAPI_CALL
360:       SHPWriteTree( SHPTree *hTree, const char * pszFilename );
361: SHPTree SHPAPI_CALL
362:       SHPReadTree( const char * pszFilename );
363:
364: int  SHPAPI_CALL
365:       SHPTreeAddObject( SHPTree * hTree, SHPObject * psObject );
366: int  SHPAPI_CALL
367:       SHPTreeAddShapeId( SHPTree * hTree, SHPObject * psObject );
368: int  SHPAPI_CALL
369:       SHPTreeRemoveShapeId( SHPTree * hTree, int nShapeId );
370:
371: void   SHPAPI_CALL
372:       SHPTreeTrimExtraNodes( SHPTree * hTree );
373:
374: int    SHPAPI_CALL1(*)
375:       SHPTreeFindLikelyShapes( SHPTree * hTree,
376:                                double * padfBoundsMin,
377:                                double * padfBoundsMax,
378:                                int * );
379: int     SHPAPI_CALL
380:       SHPCheckBoundsOverlap( double *,
381:            double *, double *, double *, int );
382: /************************************************************/
383: /*                      DBF Support.                        */
384: /************************************************************/
385: typedef  struct
386: {
387:     FILE  *fp;
388:
389:     int         nRecords;
390:
391:     int    nRecordLength;
392:     int    nHeaderLength;
393:     int    nFields;
394:     int    *panFieldOffset;
395:     int    *panFieldSize;
396:     int    *panFieldDecimals;
397:     char  *pachFieldType;
398:
399:     char  *pszHeader;
400:
401:     int    nCurrentRecord;
402:     int    bCurrentRecordModified;
403:     char  *pszCurrentRecord;
404:
```

```
405:     int    bNoHeader;
406:     int    bUpdated;
407: } DBFInfo;
408:
409: typedef DBFInfo * DBFHandle;
410:
411: typedef enum {
412:   FTString,
413:   FTInteger,
414:   FTDouble,
415:   FTLogical,
416:   FTInvalid
417: } DBFFieldType;
418:
419: #define XBASE_FLDHDR_SZ       32
420:
421: DBFHandle SHPAPI_CALL
422:       DBFOpen( const char * pszDBFFile, const char * pszAccess );
423: DBFHandle SHPAPI_CALL
424:       DBFCreate( const char * pszDBFFile );
425:
426: int  SHPAPI_CALL
427:       DBFGetFieldCount( DBFHandle psDBF );
428: int  SHPAPI_CALL
429:       DBFGetRecordCount( DBFHandle psDBF );
430: int  SHPAPI_CALL
431:       DBFAddField( DBFHandle hDBF, const char * pszFieldName,
432:                    DBFFieldType eType, int nWidth, int nDecimals );
433:
434: DBFFieldType SHPAPI_CALL
435:       DBFGetFieldInfo( DBFHandle psDBF, int iField,
436:         char * pszFieldName, int * pnWidth, int * pnDecimals );
437:
438: int SHPAPI_CALL
439:       DBFGetFieldIndex(DBFHandle psDBF, const char *pszFieldName);
440:
441: int   SHPAPI_CALL
442: DBFReadIntegerAttribute( DBFHandle hDBF, int iShape, int iField );
443: double   SHPAPI_CALL
444: DBFReadDoubleAttribute( DBFHandle hDBF, int iShape, int iField );
445: const char SHPAPI_CALL1(*)
446: DBFReadStringAttribute( DBFHandle hDBF, int iShape, int iField );
447: const char SHPAPI_CALL1(*)
448: DBFReadLogicalAttribute( DBFHandle hDBF, int iShape, int iField );
449: int     SHPAPI_CALL
450:   DBFIsAttributeNULL( DBFHandle hDBF, int iShape, int iField );
451:
452: int SHPAPI_CALL
453: DBFWriteIntegerAttribute( DBFHandle hDBF, int iShape, int iField,
454:                                  int nFieldValue );
455: int SHPAPI_CALL
456:  DBFWriteDoubleAttribute( DBFHandle hDBF, int iShape, int iField,
457:                                 double dFieldValue );
458: int SHPAPI_CALL
459: DBFWriteStringAttribute( DBFHandle hDBF, int iShape, int iField,
460:                                 const char * pszFieldValue );
```

```
461: int SHPAPI_CALL
462: DBFWriteNULLAttribute( DBFHandle hDBF, int iShape, int iField );
463:
464: int SHPAPI_CALL
465: DBFWriteLogicalAttribute( DBFHandle hDBF, int iShape, int iField,
466:              const char lFieldValue);
467: int SHPAPI_CALL
468: DBFWriteAttributeDirectly(DBFHandle psDBF, int hEntity, int iField,
469:                                   void * pValue );
470: const char SHPAPI_CALL1(*)
471:       DBFReadTuple(DBFHandle psDBF, int hEntity );
472: int SHPAPI_CALL
473: DBFWriteTuple(DBFHandle psDBF, int hEntity, void * pRawTuple );
474:
475: DBFHandle SHPAPI_CALL
476:       DBFCloneEmpty(DBFHandle psDBF, const char * pszFilename );
477:
478: void  SHPAPI_CALL
479:       DBFClose( DBFHandle hDBF );
480: void    SHPAPI_CALL
481:       DBFUpdateHeader( DBFHandle hDBF );
482: char    SHPAPI_CALL
483: DBFGetNativeFieldType( DBFHandle hDBF, int iField );
484:
485: #ifdef __cplusplus
486: }
487: #endif
488:
489: #endif /* ndef _SHAPEFILE_H_INCLUDED */
```

Listing of utils.h

```
  1: /* Taken from txt2dbf 1.0.2 by Frank Koormann, see
  2:  * http://www.usf.uni-
    osnabrueck.de/~fkoorman/software/dbftools.en.html
  3:  */
  4:
  5: /*
  6:  * $Source: utils.h,v$
  7:  *
  8:  * $Author: fkoorman $
  9:  *
 10:  * $Revision: 1.2 $
 11:  *
 12:  * Description: header of utils.c
 13:  *  collection of useful functions:
 14:  *  - getline
 15:  *  - do_nothing
 16:  *  - tabtok
 17:  *  explanations see below.
 18:  */
 19:
 20: #ifndef __utils__
 21: #define __utils__
 22:
 23: #include <stdio.h>
```

10.6 - Listing of the Exp2Shp Utility

```
24: #include <stdlib.h>
25: #include <string.h>
26:
27: /* global variables ------------------------------------*/
28: /* action of a program after evaluating the command line */
29: #define ABORT 0
30: #define EVALUATE 1
31: #define LIST 2
32:
33: /* function declaration ---------------------------------*/
34:
35: /* getline: reads a line out of stream fp, returns last
36:  * char, esp eof              */
37: int getline(FILE *fp, char s[]);
38:
39: /* do_nothing: dto.              */
40: void do_nothing( void );
41:
42: /* tabtok -----------------------------------------------
43:  * breaks a string in sequences delimited by tabs, but not
44:  * sequences of directly followed tabs: like "\t\t\ttest"
45:  */
46: char *tabtok( char *s );
47:
48: /* dtok -------------------------------------------------------
49:  * breaks a string in sequences delimited by delim, but do not
50:  * sequences of directly followed delims: like "\t\t\ttest"
51:  */
52: char *dtok( char *s , char delim );
53:
54: #endif
```

10.7 - Listing of the C Code to Intersect Contours with Rivers

Care must be taken with GIS data. All data was converted to metric if not already. Some data was available on a county basis, data from county boundaries must agree. I found some discrepancies and sent the information off to Iowa.

The prescribed head data for the rivers had to be identified. The Rivers shapefile arc data were split into separate polylines. The Contours arc data were also split into polylines. The two sets of data were then intersected and the CONTOUR from the Contours shapefile was attached to the position of the Rivers shapefile where they intersected. This C program took more than ten minutes to run on a 2.4 GHz windows 2000 computer with 512 DDR 2700 SDRAM. The source code is shown below.

```
#include        <stdio.h>
#define MAX(a,b) ((a)<(b) ? (b) : (a))
#define MIN(a,b) ((a)<(b) ? (a) : (b))
/*
 * This code will read in all the river arcs into memory
 * which is 13764 different arcs.  This is from the
 * River.txt file.  Then the topographic arcs will be read
 * from the Topo.txt file.  Each topo arc will be
 * compared to each of the river arcs to see if there
 * is an intersect point.
 */
int mygets(p,fpin)
char *p;
FILE *fpin;
{
    int ch;
    int chcnt;
    chcnt=0;
    while((ch = fgetc(fpin)) != (int)'\n' && ch != EOF)
    {
      *p++ = (char)ch;
      chcnt++;
    }
    *p = 0;
    if(chcnt>1 && *(p-1) == '\r') *(p-1)=0;
    return(ch);
}

void main(argc, argv)
int argc;
char *argv[];
{
    int ch,chr; /* for character input */
    char *p,*q,cntbuf[512],labelbuf[512]; /* for character processing */
    int r,cnt,rivercnt[14000],atoi(); /* for river data */
    double X1[14000],Y1[14000],X2[14000],Y2[14000],atof(); /* for river data
    */
    double TopoX1,TopoY1,TopoX2,TopoY2; /* for Topographic data */
    double rr,rru,rrl,ss,ssu,ssl,Px,Py; /* for intersection caalculation */
    FILE *fp1, *fp2; /* two input files: River and Topographic data */

    if(argc == 2 &&
       (strcmp(argv[1],"-?") == 0 ||
            strcmp(argv[1],"-h") == 0 ||
            strcmp(argv[1],"-H") == 0)
      )
```

10.7 - Listing of the C Code to Intersect Contours with Rivers

```
    {
      printf("Usage : %s  Rivers.txt Topo.txt\n", argv[0]);
      exit(0);
    }
    if(argc < 3)
    {
      printf("Usage : %s  Rivers.txt Topo.txt\n", argv[0]);
      exit(0);
    }
    if( (fp1 = fopen( argv[1], "r" )) == NULL )
    {
      perror( argv[1] );
      exit( 1 );
    }
    if( (fp2 = fopen( argv[2], "r+" )) == NULL )
    {
      perror( argv[2] );
      fclose(fp1);
      exit( 1 );
    }
    /* Rivers.txt
0
455657.543535679,4641529.915645910
455657.500015679,4641529.999997910
end
1
455657.500015679,4641529.999997910
455633.624943657,4641627.999998000
end
     */
    cnt=0;
    ch = mygets(labelbuf,fp1); /* get first label */
    while( (ch = mygets(cntbuf,fp1)) != EOF)
    {
      rivercnt[cnt]=atoi(labelbuf);

      p=&cntbuf[0]; q=p;
      while(*q != ',') q++; *q++=0;
      X1[cnt]=atof(p); Y1[cnt]=atof(q);

      ch = mygets(cntbuf,fp1); /* get second point */
      p=&cntbuf[0]; q=p;
      while(*q != ',') q++; *q++=0;
      X2[cnt]=atof(p); Y2[cnt]=atof(q);

      ch = mygets(cntbuf,fp1); /* get end */

      ch = mygets(labelbuf,fp1); /* get the label */
      p=&labelbuf[0];
      if(strcmp(p,"end")==0 || strcmp(p,"END")==0)
                        break;
      cnt++;

    }
    fclose(fp1);

    ch = mygets(labelbuf,fp2); /* get first label */
```

10.7 - Listing of the C Code to Intersect Contours with Rivers

```
while( (ch = mygets(cntbuf,fp2)) != EOF)
{
  p=&cntbuf[0]; q=p;
  while(*q != ',') q++; *q++=0;
  TopoX1=atof(p); TopoY1=atof(q);

  ch = mygets(cntbuf,fp2); /* get second point */
  p=&cntbuf[0]; q=p;
  while(*q != ',') q++; *q++=0;
  TopoX2=atof(p); TopoY2=atof(q);

  ch = mygets(cntbuf,fp2); /* get end */

  /* this is the UTM limits of the entire basin
   *                4706300
   * 423100                              462500
   *                4641670
   */
  /* leftX = MIN(TopoX1,TopoX2); rightX=MAX(TopoX1,TopoX2); */
  /* lowerY = MIN(TopoY1,TopoY2); upperY=MAX(TopoY1,TopoY2); */
  for(r=0; r<cnt; r++)
  { /*
        if(X1[r] < leftX && X2[r] < leftX) continue;
        if(X1[r] > rightX && X2[r] > rightX) continue;
        if(Y1[r] > upperY && Y2[r] > upperY) continue;
        if(Y1[r] < lowerY && Y2[r] < lowerY) continue;
        */
rru = ((TopoY1-Y1[r])*(X2[r]-X1[r]))-((TopoX1-X1[r])*(Y2[r]-Y1[r]));
rrl = ((TopoX2-TopoX1)*(Y2[r]-Y1[r]))-((TopoY2-TopoY1)*(X2[r]-X1[r]));
        if(rrl != 0)
        {
              rr = rru/rrl; /* egn1 */
ssu = ((TopoY1-Y1[r])*(TopoX2-TopoX1))-((TopoX1-X1[r])*(TopoY2-TopoY1));
ssl = ((TopoX2-TopoX1)*(Y2[r]-Y1[r]))-((TopoY2-TopoY1)*(X2[r]-X1[r]));
              if(ssl != 0)
              {
                    ss = ssu/ssl; /* eqn2 */
                    if(rr >= 0 && rr <= 1 && ss >= 0 && ss <= 1)
                    {
                          Px=TopoX1+rr*(TopoX2-TopoX1);
                          Py=TopoY1+rr*(TopoY2-TopoY1);
                          fprintf(stdout,"%d\t%.10f\t%.10f\t%s\n",
                                      rivercnt[r],Px,Py,labelbuf);
                    }
              }
        }

/* taken from http://www.faqs.org/faqs/graphics/algorithms-faq/
    Let A,B,C,D be 2-space position vectors.  Then the directed line
    segments AB & CD are given by:
        AB=A+r(B-A), r in [0,1]
        CD=C+s(D-C), s in [0,1]
    If AB & CD intersect, then
        A+r(B-A)=C+s(D-C), or
        Ax+r(Bx-Ax)=Cx+s(Dx-Cx)
        Ay+r(By-Ay)=Cy+s(Dy-Cy)  for some r,s in [0,1]
    Solving the above for r and s yields
```

10.7 - Listing of the C Code to Intersect Contours with Rivers

```
                (Ay-Cy)(Dx-Cx)-(Ax-Cx)(Dy-Cy)
          r = -----------------------------  (eqn 1)
                (Bx-Ax)(Dy-Cy)-(By-Ay)(Dx-Cx)
                (Ay-Cy)(Bx-Ax)-(Ax-Cx)(By-Ay)
          s = -----------------------------  (eqn 2)
                (Bx-Ax)(Dy-Cy)-(By-Ay)(Dx-Cx)
    Let P be the position vector of the intersection point, then
          P=A+r(B-A) or
          Px=Ax+r(Bx-Ax)
          Py=Ay+r(By-Ay)
    By examining the values of r & s, you can also determine some
    other limiting conditions:
          If 0<=r<=1 & 0<=s<=1, intersection exists
              r<0 or r>1 or s<0 or s>1 line segments do not intersect
          If the denominator in eqn 1 is zero, AB & CD are parallel
          If the numerator in eqn 1 is also zero, AB & CD are collinear.
    If they are collinear, then the segments may be projected to the x-
    or y-axis, and overlap of the projected intervals checked.
    If the intersection point of the 2 lines are needed (lines in this
    context mean infinite lines) regardless whether the two line
    segments intersect, then
          If r>1, P is located on extension of AB
          If r<0, P is located on extension of BA
          If s>1, P is located on extension of CD
          If s<0, P is located on extension of DC
    Also note that the denominators of eqn 1 & 2 are identical.

    References:

    [O'Rourke (C)] pp. 249-51
    [Gems III] pp. 199-202 "Faster Line Segment Intersection,"

    Computational Geometry in C (2nd Ed.)
    Joseph O'Rourke, Cambridge University Press 1998,
    ISBN 0-521-64010-5 Pbk, ISBN 0-521-64976-5 Hbk
    Additional information and code at http://cs.smith.edu/~orourke/ .
             */
      }

      ch = mygets(labelbuf,fp2); /* get the label */
      p=&labelbuf[0];
      if(strcmp(p,"end")==0 || strcmp(p,"END")==0)
                        break;
    }

    fclose(fp2);
}
```

10.8 - Description of Iowa DNR GIS Data

Map Projection: Universal Transverse Mercator (UTM), Zone 15.
Map Units: Meters
Datum: NAD27
Software Version: PC ARC/INFO 3.4D
Coverage Created: 8/31/92
Coverage Modifications: The linework for the rivers was modified to match the 1:100,000 DLG linework and the stream names (PNAME) were corrected to match USGS maps or the 'Drainage Areas of Iowa Streams' names. The stream levels (LEVEL) were also corrected. This coverage contains arcs representing the river system and water bodies for BOONE County. This coverage was developed from the USEPA's REACH FILE 3 data system. Graphic elements for this coverage are from 1:100,000 scale digital line graphs (DLG) files from the USGS. Attibute data for each arc were added by USEPA contractors. Rivers were originally subdivided by river basin, but currently are merged and clipped by county. The RF3 data files were an interim version with many errors and incomplete attributes and documentation. A new version of this file from the improved RF3 data will be eventually produced. The positional accuracy of the coverage coordinates is a deductive estimate based on possible errors that may have occurred during each production step. These errors include source of data, digitization, and processing. The accuracy is estimated to be 52 meters using a root-sum-square error calculation.

The topo coverage contains arcs representing the surface elevation (topographic) contours of Boone County. These arcs were derived from the hypsography layer of the 1 x 1/2 degree USGS 100k digital line graph files (DLG). The contours are metric and have a contour interval of 10 meters.