Paper 1199 Title

Groundwater Modeling of the Upper Skunk Basin

Author

John M. Huddleston, PE, PhD

Paper Abstract

Groundwater modeling of a 156,000 hectare basin in IOWA was made possible by intersection of hypsography (contours) and hydrography (river) data layers to determine elevations of the rivers. The elevation of the river is needed as input prescribed heads for the groundwater model. In order to intersect the river and contour data the river and contour data were exported into a GEN file using ArcMAP and a plug-in called ET GeoWizards. The exported files resemble the ArcInfo Generate (GEN) format. A new procedure was devised to intersect the river and contour polylines. The non-trivial logic had been worked out by others at www.faqs.org. The data used for the study is hosted on the IOWA Department of Natural Resources (DNR) Web site (ftp.igsb.uiowa.edu).

TABLE OF CONTENTS

CHAPTER 1
GENERAL INTRODUCTION

1.1 INTRODUCTION 1
1.2 DEFINITION OF THE PROBLEM 1
1.3 OBJECTIVE 2
1.4 LAYOUT OF PAPER 1199 2

CHAPTER 2
INVERSE MODELING

2.1 UNCERTAINTY 3
2.2 OPTIMIZATION 3
2.3 OPTIMIZATION APPLIED TO GROUNDWATER MODELING 3

CHAPTER 3
AVAILABLE SOURCES OF DATA

3.1 INTRODUCTION 5
3.2 USDA NATIONAL SOIL TILTH LAB 5
3.3 IOWA DEPARTMENT OF NATURAL RESOURCES 5

CHAPTER 4
GIS MODELING OF THE UPPER SKUNK BASIN

4.1 ESTABLISHING THE FORWARD MODEL 6
4.2 UPPER SKUNK GROUNDWATER MODEL DATA 8
4.3 UPPER SKUNK GROUNDWATER MODELING RESULTS 8
4.4 SUMMARY OF THE UPPER SKUNK GROUNDWATER MODELING 9

CHAPTER 5
CONCLUSION

5.1 ANALYSIS 10
5.2 RESULTS 10

CHAPTER 6
APPENDICES

6.1 C LISTING 10
6.2 EXP2SHP LISTING 12

CHAPTER 7
REFERENCES 17

1.0 GENERAL INTRODUCTION

1.1 INTRODUCTION

Groundwater modeling of a large basin of 156,000 hectares, the Upper Skunk River basin in Iowa, would take years of analysis if it were not for Geographical Information System (GIS) data. Even then the level of accuracy for the output results is only as good as the input data. To improve on the accuracy of the input data, a groundwater model of a 5,200 hectare Walnut Creek watershed within the Upper Skunk basin was created (Huddleston, 1984). Using the USGS MODFLOW 2000 FORTRAN model and detailed GIS data from the National Soil Tilth Lab in Ames, Iowa an inverse model was created for Walnut Creek. Walnut Creek was modeled as a two layer unconfined system. The results from the MODFLOW 2000 least squares (LSG) inverse procedure did not agree with published data. The LSG procedure was replaced with a Least sum Absolute Deviation regression and Expectation Maximization (LAD-EM) procedure. The LAD-EM MODFLOW results were in agreement with published data. The results of both inverse models were incorporated into a groundwater model of the larger encompassing Upper Skunk River basin. The problems of applying small watershed parameters to a larger encompassing basin will be discussed. Both the Walnut Creek watershed and the Upper Skunk River basin are to be modeled using ArgusONE GIS interface to MODFLOW. The issues related to GIS modeling will be solved using ESRI, ET GeoWizards, and compiled C software.

1.2 DEFINITION OF THE PROBLEM

The Walnut Creek watershed in north central Iowa is 5178.8 hectares. Walnut Creek watershed is primarily an agricultural watershed within the low-relief landscape of the Des Moines Lobe.

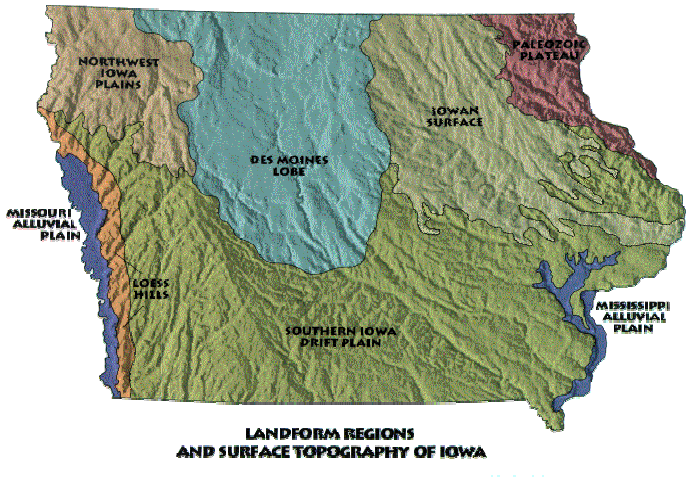

Figure 1.1 Landform Regions In Iowa

Eight years of rainfall, runoff, and groundwater observations are available for the Walnut Creek watershed. However, while literature cites estimates of the hydraulic conductivity for this area, a more detailed number is required to match groundwater model output with observed data.

The problem of determining the hydraulic conductivity is referred to as the inverse problem. 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. Before MODFLOW 2000 was released, experienced groundwater modelers used trial and error approaches to estimate the parameters. The number and combinations of parameter adjustments are not bounded making "trial and error" a time consuming process. 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.

Automated inverse models identify unknown model parameters by using mathematical optimization techniques. The most important technique is to minimize the model forward parameter. This minimization is carried

out over a geographically defined domain using a user specified grid cell size. The forward parameter in a groundwater flow model is the hydraulic conductivity. When the maximum value of any one cell of a model computed hydraulic conductivity is less than a pre-defined tolerance, then the inverse model is said to converge.

When convergence of the forward parameter does not occur, the groundwater flow inverse model will minimize an objective function based on pressure head. Initial forward parameter estimates are used, the forward model is run, the objective function is evaluated, and new estimates of the forward parameter are used for a new forward model run. The cycle continues until the objective function criteria have been met.

When the forward parameter cannot be minimized and the objective function does not converge to a minimum tolerance, the process is terminated after a user specified number of cycles. If this occurs, the groundwater model itself will have to be re-architected.

1.3 OBJECTIVE

The objective is to provide a groundwater model of the Upper Skunk basin. Results of the inverse modeling for the smaller Walnut Creek watershed (Huddleston, 2004) are used to define the hydraulic conductivity for the aquifer layer. GIS data is used to define the elevation of the bedrock topology, surface contours, watershed domain, and streams for the Upper Skunk groundwater model. MODFLOW 2000 by itself does not contain a geospatial interface; however there are several to choose from in the commercial sector.

ArgusONE is one of the geospatial interfaces that can be used to pre-process hydrography, hypsography elevation data, and rainfall data as input into MODFLOW 2000. This ArgusONE (http://www.argusint.com) interface also performs post processing of the MODFLOW 2000 output for graphic and image analysis.

Modeling of Walnut Creek watershed (Huddleston, 1984) determined that the forward parameters computed by the MODFLOW 2000 LSG procedure predicted a confined groundwater aquifer. Using a modified MODFLOW 2000 LAD-EM procedure, a second set of forward parameters was calculated predicting an unconfined aquifer. Both sets of forward parameters (hydraulic conductivity) will be used in the Upper Skunk river groundwater model.

1.4 LAYOUT OF PAPER 1199

Chapter 2 will present the theory behind the inverse modeling. Normally, inverse modeling theory would not be presented within a geospatial paper; however, the groundwater modeling will be performed with two sets of results from MODFLOW 2000. For completeness, both the LSG and LAD-EM theory is presented. Chapter 3 will discuss the available sources of GIS data. Chapter 4 will present the GIS solution for modeling the Upper Skunk basin. Analysis of model results and comparisons between the groundwater model results of the differing sets of forward parameters will be discussed. Chapter 5 presents a summary of the analysis and results. Chapter 6 contains a listing of the source code. Chapter 7 contains a compendium of references for this topic.

2.0 INVERSE MODELING

2.1 UNCERTAINTY

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 usual 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 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,

$$\bar{X} - z_{\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right) \le \mu \le \bar{X} \le \bar{X} + z_{\alpha/2}\left(\frac{\sigma}{\sqrt{n}}\right) \quad 2.1$$

in which $\bar{X}$ is the sample mean, $z_{\alpha/2}$ are values of random variables having a standard normal distribution; α is the level of significance; and σ is the standard deviation of the population n. 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

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.

2.2 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).

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.

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 of equations $Ax = b$, the least absolute deviation regression estimate that minimizes the error $\hat{x} = \text{argmin}_x \|b - Ax\|_1$ is shown as follows:

$$f(x) = \|b - Ax\| = \sum_i \left| b_i - \sum_j a_{ij} x_j \right| \qquad 2.2$$

The minimum can be found by differentiating f(x) and setting the result to 0

$$\frac{\partial f}{\partial x_k}(\hat{x}) = \sum_{i=1}^{m} \frac{b_i - \sum_j a_{ij}\hat{x}_j}{\left| b_i - \sum_j a_{ij}\hat{x}_j \right|}(-a_{ik}) = 0, \quad k=1,2,...,n \qquad 2.3$$

Equation 2.3 can be rearranged as

$$\sum_i \frac{a_{ik} b_i}{e_i(\hat{x})} = \sum_i \sum_j \frac{a_{ik} a_{ij} \hat{x}_j}{e(\hat{x})}, \quad k=1,2,...,n \qquad 2.4$$

Equation 2.4 can be expressed in matrix notation as:

$$A^T E(\hat{x}) b = A^T E(\hat{x}) A\hat{x}, \text{ where } E(\hat{x}) = \text{Diag}(e(\hat{x}))^{-1} \qquad 2.5$$

If $A^T E(\hat{x}) A$ is invertible, then the estimate can be found as:

$$\hat{x} = \left(A^T E(\hat{x}) A\right)^{-1} A^T E(\hat{x}) b \qquad 2.6$$

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:

$$f(x) = \|b - Ax\|_2^2 = \sum_i \left(b_i - \sum_j a_{ij} x_j \right)^2 \qquad 2.7$$

The minimum can be found by differentiating f(x) and setting the result to 0

$$\frac{\partial f}{\partial x}(\bar{x}) = \sum_i 2\left(b_i - \sum_j a_{ij}\bar{x}_j \right)(-a_{ik}) = 0, \quad k=1,2,...,n \qquad 2.8$$

Equation 2.8 can be rearranged as follows:

$$\sum_i a_{ik} b_i = \sum_i \sum_j a_{ik} a_{ij} \bar{x}_j, \quad k = 1,2,...,n \qquad 2.9$$

Equation 2.9 can be expressed in matrix notation as:

$$A^T b = A^T A\bar{x} \qquad 2.10$$

If $A^T A$ is invertible, then the estimate can be found as:

$$\bar{x} = \left(A^T A\right)^{-1} A^T b \qquad 2.11$$

In the generation of a solution of equation 2.11, several problems may be encountered. It is possible that the sequence does not converge. Also, matrix $A^T A$ may be near singular (elements very close to zero), and a solution

cannot be obtained. Finally, the displacement vector Δx may become so large that parameter values are no longer in the admissible region.

In order to avoid these difficulties, it is necessary to modify equation 2.11 to guarantee convergence. The modification includes an additional term added to A^TA to avoid the singularity:

$$\hat{x} = \left(A^TA+\lambda_M I\right)^{-1} A^Tb \qquad 2.12$$

where λ_M is a coefficient and I is the unit matrix. The MODFLOW 2000 inverse procedure uses an algorithm based on Dennis et al. (1981) for estimating the additional term.

$$\lambda_M I = A_k^T w\, A - \frac{\partial^2 Y}{\partial b_i \partial b_j} \qquad 2.13$$

The estimate is the difference between $A_k^T w A_k$ and the Hessian matrix equation defined in Hill, 1992, Sun, 1994, and Huddleston, 2004 where w is a matrix of weights.

3.0 AVAILABLE SOURCES OF DATA

3.1 INTRODUCTION

The study area references for the geology and hydrogeology in this paper 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).

3.2 USDA NATIONAL SOIL TILTH LAB

Detailed GIS data is made possible for Walnut Creek as a result of the work of the USDA National Soil Tilth Lab in Ames, Iowa. Coverages of the Walnut Creek watershed are available in NAD83 format. Many of these were created by Wolfgang Oesterreich. These are located in Walnut Creek Watershed, Boone & Story Counties, Iowa. The data has been projected from NAD27 to NAD83. The Trimble Pathfinder Professional GPS model ASSETSURVEYOR DATA COLLECTION program was used to spotcheck accuracy of the GIS data. The accuracy of the data is one meter.

A grid of elevations in-and-around Walnut Creek Watershed, Boone & Story Counties, Iowa was created by Jon Pickus, Lockheed, Las Vegas, NV for EPA using the Grid KRIGING method. The projection is UTM. The zone is north 15. The units are meters. This data was also used to verify the data.

3.3 IOWA DEPARTMENT OF NATURAL RESOURCES

The Iowa Department of Natural Resources Natural Resources (DNR) began development of the Natural Resources Geographic Information System (NRGIS) following passage of the 1987 Groundwater Protection Act. This legislation included provisions for developing a Geographic Information System (GIS) because it recognized that GIS technology successfully was being employed to assess complex natural resource and environmental problems. Further, it could be used to develop information for the public to help explain these issues. The groundwater legislation also required DNR to develop a map depicting the vulnerability of groundwater to contamination. Thus, DNR initially focused much of its GIS activities towards the issue of groundwater contamination and development of the map product, Groundwater Vulnerability Regions of Iowa. The GIS data developed for this project became the foundation of the NRGIS.

The main FTP internet resource site for NRGIS is ftp://ftp.igsb.uiowa.edu/pub/gisdata/. The NRGIS Library is the cornerstone to the department's GIS capability. It is an organized collection of geographically referenced databases. The databases are thoroughly

documented and available to all NRGIS users. The NRGIS Library currently contains 384 PC Arc/Info coverages that are organized by geographic extent: statewide, county or regional. Practical factors were used to determine the geographic extent of each coverage; factors included database size and anticipated usage. A number of coverages are based on geographic themes including boundaries of the state, counties, townships, sections, and towns. Resource-related themes are developed in coverages such as rivers, topography, roads, and geology. Examples of some specific themes in the NRGIS Library include public water supplies, waste disposal sites, plugged water wells and registered underground storage tanks.

The coverages of interest for groundwater modeling are the hydrography and hypsography data containing river locations, surface and bedrock elevations. In Figure 3.1 below, the three counties Boone, Hamilton, and Story are shown against a backdrop of the state basins.

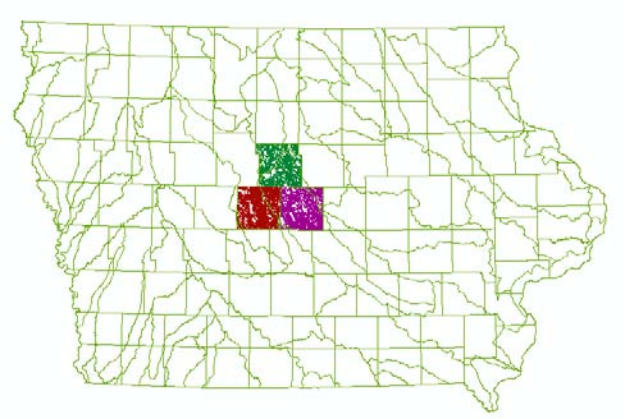

Figure 3.1 Hypsography data for three counties in Iowa

4.0 GIS MODELING OF THE UPPER SKUNK BASIN

4.1 ESTABLISHING THE FORWARD MODEL

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.

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. The Iowa GIS has a data map projection of Universal Transverse Mercator (UTM) Zone 15 and a North American Datum of 1927 (NAD27).

The boundary outline (domain) for the basin was delineated within ArgusONE using the contours as a visual guide. This ArgusONE domain outline was 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 6 Section 2 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 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 river basin. Similarly, the hypsography elevation contour layer data was intersected with the domain to produce a contour layer for the Upper Skunk.

The hydrography layer 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.

Three counties' (Boone, Story, and Hamilton) hypsography data 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). The GeoProcessing Wizard under Tools in ArcMAP "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 6 Section 1 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 4.1 shows the locations of the intersections.

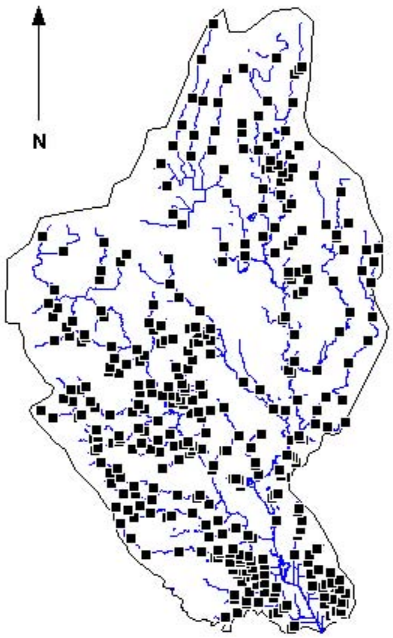

Figure 4.1 Prescribed heads in the Upper Skunk river basin In Iowa

The Iowa DNR hypsography bedrock topology was intersected with the domain shapefile using ArcMAP. The new shapefile was imported into ArgusONE as a data layer for the bottom elevation of the unoxidized late Wisconsinan till. Figure 4.2 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 4.3 shows the image of the Upper Skunk surface contours in Iowa.

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 to a meter above the surface.

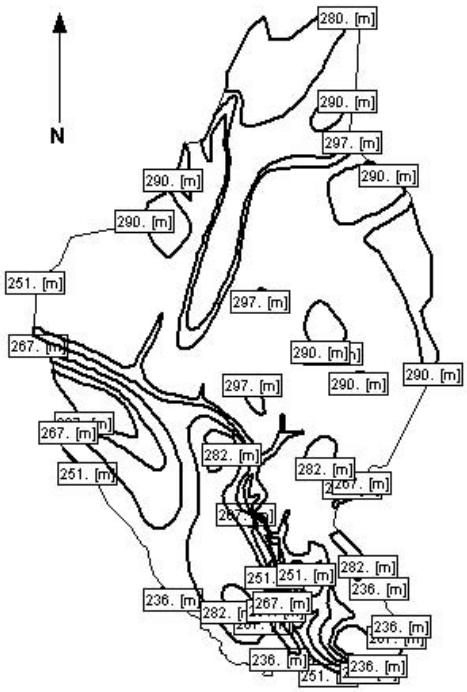

Figure 4.2 Upper Skunk DNR **bedrock** contour topology (m-msl) in Iowa

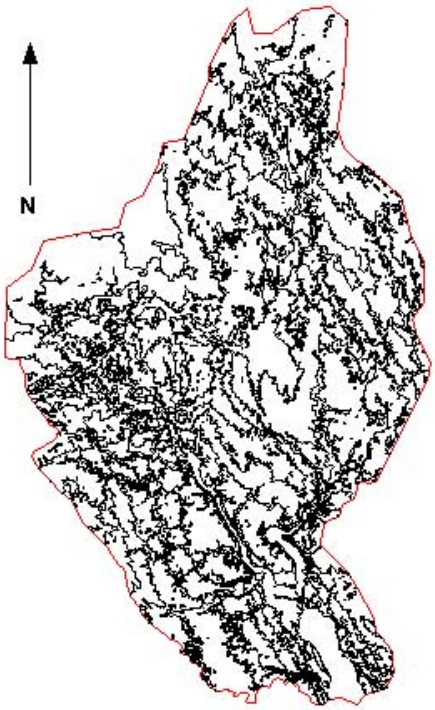

Figure 4.3 Upper Skunk DNR **surface** contour topology image in Iowa

4.2 UPPER SKUNK GROUNDWATER MODEL DATA

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. 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 that ranges between 1E-04 m/s and 1E-05 m/s. The unoxidized layer hydraulic conductivity ranges between 1E-08 m/s and 1E-09 m/s. The entire basin is an unconfined system.

The top late Wisconsinan oxidized layer varies in thickness from one to nine meters. Most of the one to three meters is immediately surrounding the rivers. Apart from the rivers the late Wisconsinan oxidized layer varies from four to nine meters. Five, six, and seven meter top thicknesses were used in the Walnut Creek inverse model. There was no difference in the computed forward parameter for six and seven meter oxidized layer thickness. A value of seven meters was used in the top oxidized layer for the Upper Skunk river basin.

Given the surface and bedrock hypsography and the top thickness, the model automatically computes the elevation of the bottom of the oxidized layer by subtracting the top thickness from the surface elevation. The top and bottom layer thickness were computed for each grid cell.

4.3 UPPER SKUNK GROUNDWATER MODELING RESULTS

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.

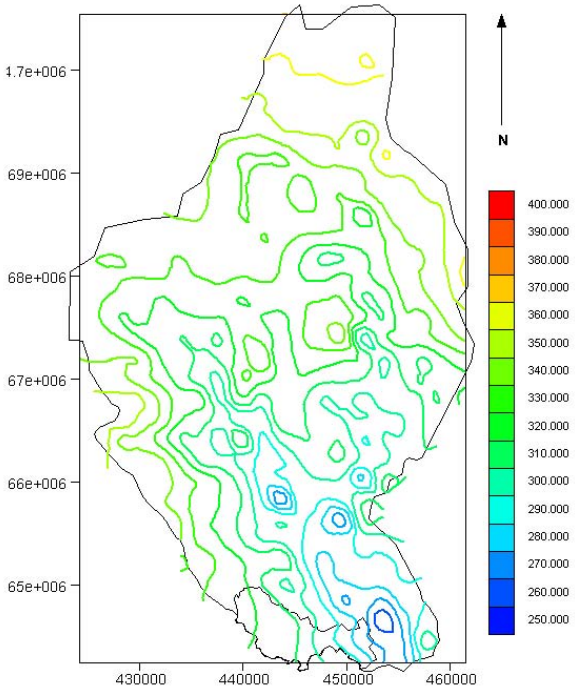

Figure 4.4 Modflow Top Thickness = 7 m Top K = 7.8E-9 Bottom K = 1.7E-4

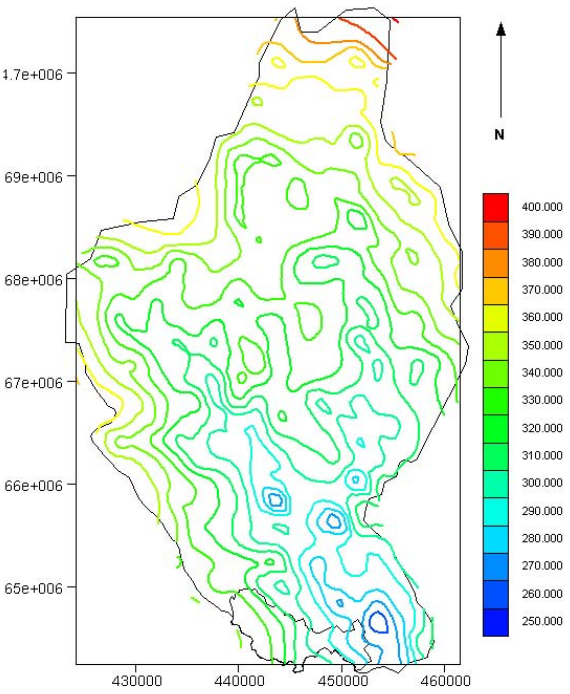

Figure 4.5 LAD-EM Top Thickness = 7 m Top K = 6.84E-04 Bottom K = 1E-10

Figure 4.4 shows the results of the groundwater modeling using hydraulic conductivity (K) values computes by the LSG method. Figure 4.5 shows the results of the groundwater modeling using hydraulic conductivity (K) values computes by the LAD-EM method.

4.4 SUMMARY OF THE UPPER SKUNK GROUNDWATER MODELING

The timing and recharge rates from Walnut Creek groundwater model were used as input to the Upper Skunk groundwater model.

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.

The USDA ARS National Soil Tilth Laboratory is NAD83 datum. The Iowa Department of Natural Resources is NAD27 datum. 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.

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

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

5.0 CONCLUSION

5.1 ANALYSIS

The NSTL GIS data had been spot checked with GPS data to an accuracy of one meter. In addition, kriging of the contours provided an additional level of accuracy for the Walnut Creek GIS data. The IOWA NAD27 GIS data did not have the benefit of such close scrutiny. Both horizontal positional accuracy and the vertical accuracy of the hypsography data are essential to large basin modeling. The ArgusONE interface is a good tool for resolving discrepancies.

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.

The use of the Walnut Creek rainfall data to model the entire Upper Skunk river basin is a stretch because of the differences in the size of the drainage areas. An alternate recommended procedure for basin analysis is to analyze all the rain gages in the basin, analyze the rainfall for similar linear variations, and apply these as areal distributions throughout the basin. However, both Walnut Creek watershed and the larger encompassing Upper Skunk river basin are within a low relief landscape not affected by orographic effects. In addition, short term (i.e. single storm events) effects are mitigated by the use of the long term (i.e. 8 ½ years) record of data in the groundwater models.

5.2 RESULTS

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 matched observed heads in the upland Walnut Creek area. The LAD-EM model hydraulic conductivity parameters were consistent with the values published by USGS (Buchmiller, 1995).

6.0 APPENDICES

6.1 C LISTING

```
/*
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)
  )
 {
  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 */
 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
```

```
        (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);
}
```

6.2 EXP2SHP LISTING

```
 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:
```

```
#ifdef USE_STRICMP
#define CASE_INSENSITIVE_STR_CMP  stricmp
#else
#define CASE_INSENSITIVE_STR_CMP  strcasecmp
#endif
double atof();
int getline(FILE *fp, char s[] )
{  int c, i;

   i=0;
   while ( (c=fgetc(fp))!=EOF && c!='\n' )
     s[i++]=c;
   if(i>0 && s[i-1] == '\r') s[i-1]='\0';
   s[i]='\0';
   return c;
}

void print_version(FILE *file)
{
   fprintf(file,"Exp2Shp version " VERSION "\n");
#ifdef DEBUG
   fprintf(file,"compiled with option: DEBUG\n");
#endif
}

static DBFHandle LaunchDbf (  const char *fname ) {
   DBFHandle  hDBF;
   char    dbffname[STR_BUFFER_SIZE];
   char    fieldname[STR_BUFFER_SIZE];

   sprintf(dbffname, "%s.dbf", fname);
   sprintf(fieldname, "%s-id", fname);

   hDBF = DBFCreate( dbffname );
   if( hDBF == NULL ) {
     fprintf(stderr, "DBFCreate(%s) failed.\n", fname );
     exit(ERR_DBFCREATE);
   }

   if (DBFAddField( hDBF, fieldname, FTInteger, 11, 0 ) == -1) {
  fprintf(stderr,"DBFAddField(hDBF,%s,FTInteger,11,0)failed.\n",
     fieldname);   exit(ERR_DBFADDFIELD);
   }

   DBFClose( hDBF );

   hDBF = DBFOpen( dbffname, "r+b" );
   if( hDBF == NULL ) {
     fprintf(stderr, "DBFOpen(%s,\"r+b\") failed.\n", dbffname );
     exit(ERR_DBFOPEN);
   }

   return hDBF;
}

static SHPHandle LaunchShp(  const char *fname,
         int ObjectType ) {
   SHPHandle  hSHP;
   SHPObject  *psShape;
   char    shpfname[STR_BUFFER_SIZE];

   sprintf(shpfname, "%s.shp", fname);

   switch (ObjectType) {
     case OBJECTTYPE_POINT:
       hSHP = SHPCreate( shpfname, SHPT_POINT );
       break;
     case OBJECTTYPE_ARCS:
       hSHP = SHPCreate( shpfname, SHPT_ARC );
       break;
     case OBJECTTYPE_LINE:
       hSHP = SHPCreate( shpfname, SHPT_ARC );
       break;
     case OBJECTTYPE_POLYGON:
       hSHP = SHPCreate( shpfname, SHPT_POLYGON );
       break;
     default:
       fprintf(stderr, "internal error: "
         "unknown ObjectType=%d\n", ObjectType);
       exit(ERR_OBJECTTYPE);
   }

   if( hSHP == NULL ) {
     fprintf(stderr, "SHPOpen(%s, shape_type) failed.\n", shpfname );
     exit(ERR_SHPOPEN);
   }

   return hSHP;
}

static void WriteDbf (  DBFHandle hDBF,
       int rec,
       int id ) {
   if (! DBFWriteIntegerAttribute(hDBF, rec, 0, id)) {
     fprintf(stderr, "DBFWriteIntegerAttribute(hDBFs,%d,1,%d)
     failed.\n", rec, id );
exit(ERR_DBFWRITEINTEGERATTRIBUTE);
   }
}

static void WritePoint(  SHPHandle hSHP,
       int rec,
       double x,
       double y ) {
   SHPObject  *psShape;

   psShape = SHPCreateObject( SHPT_POINT, rec, 0, NULL,
NULL,
                              1, &x, &y, NULL, NULL );
   SHPWriteObject( hSHP, -1, psShape );
   SHPDestroyObject( psShape );
}

static void WriteLine(  SHPHandle hSHP,
       int rec,
       int coords,
       double * x,
       double * y ) {
   SHPObject  *psShape;

   psShape = SHPCreateObject( SHPT_ARC, rec, 0, NULL, NULL,
     coords, x, y, NULL, NULL );
   SHPWriteObject( hSHP, -1, psShape );
   SHPDestroyObject( psShape );
}

static void WritePolygon(  SHPHandle hSHP,
         int rec,
         int coords,
         double * x,
         double * y,
         int nparts,
         int * partstarts) {
   SHPObject  *psShape;

DEBUG_OUT1("WritePolygon: rec = %d\n", rec);
DEBUG_OUT1("WritePolygon: nparts = %d\n", nparts);
DEBUG_OUT1("WritePolygon: coords = %d\n", coords);
```

```
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: }
```

7.0 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., Evaluating Least Absolute Deviation Regression as an Inverse Model in Groundwater Flow Calibration, PhD Dissertation Colorado State University, Fort Collins, CO, 2004.
21. Huddleston, J.M. and B.A. Shafer, Streamflow Analysis of the Colorado River, AGU Infiltration Conference, Utah, 1987.

22. 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
23. 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.
24. McLaughlin, D. B. and L.R. Townley, A reassessment of the groundwater inverse problem, Water Resources Research, 32(4), 1996. pp. 1131-1161
25. 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
26. 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
27. Mielke, P.W. and K. J. Berry, Permutation Methods A Distance Function Approach, Springer-Verlag, New York, 2001.
28. Phillips, R. F., Least absolute deviations estimation via the EM algorithm, Statistics and Computing, No. 12, 2002. pp. 281-285
29. 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
30. 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.
31. 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
32. Rao, C. R., Methodology based on the L_1-norm, in statistical inference, Sankhya, A 50, 1988. pp 289-313
33. Sauer, P.A. and J.L. Hatfield, Walnut Creek Watershed Research Protocol Report, 94-1, 1994.
34. 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.
35. 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.
36. Sheynin, O.B., R. J. Boscovich's work on probability, Archive for History of Exact Sciences, 9: 1973. pp. 306-324
37. Sun, N.Z., Inverse problems in groundwater modeling, Volume 6 of Theory and Applications of Transport in Porous Media, Kluwer Academic Publishers, 1994. 337p.
38. 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
39. 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
40. 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.
41. Thompson, C.A., Groundwater Resources, Boone County, Open File Report 82-8 WRD, Iowa Geological Survey, 1982.
42. Thompson, C.A., Groundwater Resources, Story County, Open File Report 82-85 WRD, Iowa Geological Survey, 1982.
43. Todd, D. K., Ground-Water Hydrology, Wiley, New York, 1959.

About Your Paper

2005 International ESRI Conference Paper 1199

Status: **Accepted**

Groundwater Applications

Tuesday July 26, 2005 - 8:30 AM

Room 26-A

Contact Information

John M. Huddleston, PE, PhD

USDA NRCS

2150A Centre Avenue

Fort Collins, CO 80526 USA

970-295-5485

jhuddleston@itc.nrcs.usda.gov