

SPLIT

**Win32 computer program for analytic-based
modeling of single-layer groundwater flow
in heterogeneous aquifers with particle
tracking, capture-zone delineation,
and parameter estimation**

Version 3.2

by:

**Karl Bandilla
Raghavendra Suribhatla
Dr. Igor Janković**

**Department of Civil, Structural
and Environmental Engineering
University at Buffalo**

June 1, 2006

Contents

1	Contact	3
2	Version	3
3	Manual	3
4	Copyright, Distribution, User Rights, Disclaimer	3
5	Introduction	3
6	Where to Obtain and How to Use <i>Split</i>	4
7	Analytic Elements and Restrictions	5
8	Solution Algorithm	6
9	Particle Tracking and Restrictions	6
10	Parameter Estimation and Restrictions	7
11	Parallel Processing	7
12	Input ASCII File: <i>split.dat</i>	7
12.1	Input Control	7
12.2	Output Control	9
12.3	Aquifer and Model	10
12.4	Solution Algorithms	11
12.5	Particle Tracking and Capture Zones	11
12.6	Parameter Estimation	12
12.7	Analytic Elements	13
12.8	Precision	17
13	Maximum Model Sizes	17
13.1	Default Parameters	19
13.2	File Format	19
14	Input ASCII File: <i>head.dat</i>	21
15	Input Control File: <i>stop</i>	21
16	Output Binary File: <i>solution</i>	22
17	Output ASCII File: <i>debug.out</i>	22
18	Output ASCII File: <i>progress.out</i>	23
19	Output ASCII File: <i>extract.dat</i>	23
20	Output ASCII File: <i>extract.csv</i>	24

21 Output ASCII File: <i>errors.dat</i>	25
22 Output ASCII File: <i>invModel.dat</i>	26
23 Output ASCII File: <i>headerr.dat</i>	28
24 Output Layout Files	28
25 Output Particle Tracking Files	28
26 Output <i>Surfer</i> Grid Files	29
27 Output <i>ArcInfo</i> Grid Files	29
28 Output Contour Files	29
29 Output ASCII File: <i>TransectAnalysis.csv</i>	30
30 Output ASCII File: <i>ZoneBudget.csv</i>	30
31 About Global Constant	30
32 Acknowledgement	31
References	31

1 Contact

Dr. Igor Janković
Assistant Professor
Department of Civil, Structural, and Environmental Engineering
University at Buffalo
231 Jarvis Hall
Buffalo, NY 14260-4400
office phone: 716-645-2114 x 2328, fax: 716-645-3667
home phone: 716-639-7764, fax: 716-639-9734
e-mail: ijankovi@eng.buffalo.edu

2 Version

SPLIT version 3.2

3 Manual

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation. A copy of the license is included in the distribution.

4 Copyright, Distribution, User Rights, Disclaimer

Copyright (C) 2006 Karl Bandilla

Program *Split* consists of an executable file *split.exe*, fortran source files, example files, and manual files. This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License (English and Croatian version) along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

5 Introduction

Split is a Win32 computer program that uses the analytic element method to model single-layer groundwater flow in heterogeneous aquifers. Familiarity with the method is not a prerequisite to use *Split*. The analytic element method is based on the principle of superposition. The solution of a groundwater problem is obtained by adding the influences of individual analytic elements that correspond to aquifer features. The only input for an analytic-based model are physical aquifer features. For example, if the model consists of a well and two rivers, only three elements will

be present: one for the well and one for each river. The piezometric head at any location is obtained as sum of three terms: one for the well and one for each river. The discharge is obtained analytically in the same fashion. Computation of head and discharge does not require interpolation, numerical differentiation or similar procedures. The piezometric head and discharge are represented with continuous functions everywhere. The analytic element method is not a discretization-based method. The method does not use any grids that would make the solution scale-sensitive.

To illustrate the direct correspondence of analytic elements and aquifer features consider the following example. If the hydraulic conductivity in a model varies on a cell by cell basis, each cell will be treated as analytic element, referred to as inhomogeneity. Inhomogeneities in *Split* directly correspond to aquifer features; this is different from a finite element method where cells are also introduced to represent spatial distribution of piezometric head. It is not necessary to introduce inhomogeneities in *Split* in zones where aquifer is homogeneous. *Split* will allow you to do so, but the influence of each such inhomogeneity will be zero.

Analytic element method is internally mass conservative. Mass balance is perfectly satisfied in any model built by *Split*. Analytic element method also offers high precision. All the examples included in the distribution produce visually perfect results. This includes an example where hydraulic conductivity of neighboring inhomogeneities differs by two orders of magnitude. Sharp boundaries between geologic features do not require special attention.

The mathematical description and solutions for elements in *Split* can be found in ? and ?. The linear elements are also described in ?. Circular inhomogeneities are also available in ?. The description of area-sinks is also included in ?.

Split solution algorithm is block-iterative. Coefficients are computed for one element at a time using the coefficients from previous iterations for all other elements. The algorithm for computing the coefficients of an element is Gaussian elimination. This iterative algorithm is very useful when performing parameter estimation with *Split*, as efficiency of the procedure is greatly enhanced. The algorithm behind parameter estimation in *Split* is Levenberg-Marquardt algorithm, as described in ?. Parameter estimation is available for recharge/leakage (either constant or spatially varying), hydraulic conductivity of inhomogeneities, leaky walls and drains, conductivity of leaky beds of river elements, and extractions of reservoirs.

The particle tracking procedure in *Split* (used, for example, in capture-zone analysis) is based on the fourth-order Runge-Kutta method with variable time-step and a constant space-step. The numerical errors in particle tracking for the enclosed examples are typically in the seventh or eighth significant digit of the particle locations.

6 Where to Obtain and How to Use *Split*

To obtain the current version of *Split*, visit www.groundwater.buffalo.edu and go to the Software page.

Split does not contain a Graphical User Interface (GUI). *Split* is a mathematical engine only that takes one or two ASCII input files (which may be prepared either directly, or by a GUI) and prepares a number of output files which may be loaded in *Surfer*, in *ArcView*, in *ArcInfo*, or by a GUI. The input and output files used by *Split* are described later in this document. *Split* input and output files are designed to allow integration of *Split* and Geographical Information Systems (GIS).

ArcAEM, developed by Warit Silavisesrith (PhD Student, University at Buffalo) is an extension to *ArcGIS* (ESRI) that configures the input files for *Split* and features *OSTRICH* (Matott 2004), a model-independent calibration software along a variety of tools for automated configuration,

checking and simplification of complex groundwater models. *ArcAEM* and *OSTRICH* are available for download from www.groundwater.buffalo.edu, *ArcGIS* is developed and licensed by ESRI.

Brian Johnson, Justin Blum, and Rich Soule from Minnesota Department of Health have developed scripts that allow *Split* to be used as an add-on for *ArcView*. The scripts, named *ArcFlow*, are available free of charge upon request from Minnesota Department of Health. *ArcFlow* allows you to create, run, examine, and maintain your model without ever leaving *ArcView*.

A stand-alone Graphical User Interface (*Visual BlueBird*) that was designed for *Split* and *BlueBird* (a C++ based Analytic Element Method code) is also available from www.groundwater.buffalo.edu. *BlueBird* is closely coupled to the transport code *Cardinal*.

7 Analytic Elements and Restrictions

The most common element in *Split* has the geometry of a polygonal line, which is referred to as string. Strings can be head-specified, head and resistance-specified, extraction-specified, and discharge-specified. Strings are also used to model leaky walls and drains. The leaky wall is a linear feature with conductivity much smaller than the conductivity of the aquifer. The flow is perpendicular to the wall. The drain is a linear feature with conductivity much larger than the conductivity of the aquifer. The flow in the drain is parallel to the drain. Polygons (closed strings) are used for modeling area-sinks of specified recharge/leakage, and polygonal inhomogeneities in hydraulic conductivity. Discharge-specified wells, circular and elliptical inhomogeneities in hydraulic conductivity, and elliptical lakes are also included. Head-specified wells can be modeled as a short head-specified line.

Line segments must have a non-zero length. In general, the strings in *Split* should not be intersected. However, in most cases (e.g. a river, modeled as a head-specified string, crossing a polygonal inhomogeneity) *Split* will work properly even when intersections are present. In that case, the intersection point must be a vertex of both strings. If you do not specify this point as a vertex of BOTH strings, *Split* will diverge. If you are not sure whether your set-up is possible, try it, and see if the result makes sense. A nice feature of *Split* is that the solution is generally either perfect (or almost perfect) or, if something was wrong, way off.

In some cases, strings can share sides (one straight line-segment may belong to more than one string). If two strings share a side, they must also share the vertices of that side. If a side of a string is contained by a side of another string which is longer, the requirement can be met by introducing vertices on the longer side that correspond to the vertices on the shorter side. Leaky walls and polygonal inhomogeneities may share sides. Head-specified segments and polygonal inhomogeneities may share sides.

Strings that are used for area-sink boundaries are an exception of these rules. They may be intersected by all other strings (without introducing a vertex at the intersection point), and may share sides (without coinciding vertices) with all other strings.

The area-sinks, polygonal and circular inhomogeneities may be nested. If two inhomogeneities are nested, the hydraulic conductivity inside the smaller inhomogeneity is the conductivity of the smaller inhomogeneity. The zone between the inhomogeneities has the conductivity of the larger inhomogeneity. If area-sinks are nested, the leakage in the overlapping zone is the sum of leakages of overlapping area-sinks. Polygonal inhomogeneities may not be contained by circular inhomogeneities, but circular inhomogeneities may be contained by polygonal inhomogeneities. Polygonal inhomogeneities may share sides. Circular inhomogeneities may not be intersected.

The influences of individual elements in *Split* are obtained by truncating corresponding analytic solutions which are expressed as infinite series. The truncation level is directly related to the

precision: higher truncation level results in higher precision. Five precision levels are available. Each corresponds to a different truncation level, as will be described.

8 Solution Algorithm

The iterative algorithm behind *Split* is a string-based iterative algorithm; iterations are performed between various strings in the model. *Split* offers two iterative algorithms: a Gauss-Seidel algorithm and a Jacobi algorithm. In the Gauss-Seidel algorithm new coefficients are used as soon as they become available, whereas in the Jacobi algorithm only coefficients from the previous iteration step are used. On the element level, the coefficients are computed explicitly using Gaussian elimination with partial pivoting.

Head-specified elements can cause convergence problems when used with the Jacobi algorithm. In order to achieve convergence the fluxes of these head-specified elements need to be constrained during the computation of the element coefficients. The algorithm explicitly computes the flux of each elements needed to fulfill the boundary condition.

Superblocks are a method to reduce the number of element evaluations during the coefficient solution, gridding, and particle tracking steps. Elements get grouped together, and get represented by a single circle when viewed from a distance. For more information on the two superblock algorithms implemented here, please refer to ? and ?.

Convergence in *Split* is measured in terms of a dimensionless change of coefficients (of individual elements) between subsequent global iterations. The maximum (over all coefficients) value of this relative change, named 'Maxchange', is reported at the end of each iteration. The target value of the 'Maxchange' is **PotentialTolerance**; the global iterations will terminate when 'Maxchange' is less than **PotentialTolerance** and when **MinimumSolveIterations** iterations are performed. If 'Maxchange' is larger than **PotentialTolerance** after **MaximumSolveIterations** iterations, the iterations will be terminated regardless of the value of 'Maxchange'. The default values of **PotentialTolerance**, **MinimumSolveIterations**, and **MaximumSolveIterations** should be used most of the time.

9 Particle Tracking and Restrictions

The particle tracking in *Split* is based on the fourth-order Runge-Kutta method with a variable time step and a constant space-step. The particle locations are reported at rendezvous times, which are specified by the user (please see **Particle Tracking Parameters** section later on). Rendezvous times are not related to the internal time step size in *Split* which is not accessible by the user.

There are three restrictions of particle tracking in *Split*:

- The particle tracking in *Split* is two-dimensional particle tracking. Vertical positions of particles are not computed. This poses certain restrictions and limitations of using *Split*. Consider, for example, flow toward a river. If the river does not capture all the water (e.g. some water passes underneath), *Split* will let all the particles go through. In reality, particles that are close to phreatic surface would be extracted. *Split* accounts for this in an indirect manner by requiring the user to specify if all the particles are placed initially at the phreatic surface.
- The flow inside leaky walls and drains is not included in particle tracking. If flow and residence time inside these features is important, you should replace leaky walls and drains with polygonal inhomogeneities.

- Capture zone delineation (based on particle tracking) is not available for wells that dry-out the aquifer in the vicinity of the well.

10 Parameter Estimation and Restrictions

The algorithm behind parameter estimation in *Split* is Levenberg-Marquardt algorithm, as described in ?. Parameter estimation is available for recharge/leakage (either constant or spatially varying), hydraulic conductivity of inhomogeneities, leaky walls and drains, and bed resistance of river elements. The objective function that is minimized is the sum of the squared errors in piezometric heads, where error is defined as head computed by model minus measured head. Measured heads are provided by file *head.dat*.

11 Parallel Processing

The iterative approach used by *Split* to solve for the coefficients is very suitable for parallel processing. The elements are divided among the processors and the new coefficients are communicated at the end of each iterative step. The Message Passing Interface (MPI) is used for communication between the processors.

The currently distributed version of *Split* is compiled to run on a single processor. Versions for parallel processing are available on request.

12 Input ASCII File: *split.dat*

Every time *Split* is executed, an input file must be present in the same directory. When executing *Split.exe* from the command line, you may specify the name of the input file. If the name is omitted, *Split* will try to read the input from the file *split.dat*. The following is the description of the input file (e.g. *split.dat*) parameters. The file format and an actual example are presented later. The *Split* commands in this document are typeset in bold. The commands are case insensitive (e.g. **Precision** has the same effect as **preCision**). The commands are parsed (e.g. **Precision** has the same effect as **Pre**). Commands can be issued in any order. If a same command is issued several times with different parameters (e.g. **Porosity 1.0**; then, **Porosity 0.3**), only the last occurrence will count. This does not apply for commands that specify analytic elements (e.g. **Well**), command **Particle**, command **Time** and command **CaptureZone**. All occurrences of these commands will count.

12.1 Input Control

SolveOnly Solve the flow problem only (binary file *solution* is created).

GridOnly Grid only (create head, leakage and stream-function grid files), binary file *solution* required.

TrackOnly Track only (track particles and delineate capture zones), binary file *solution* required.

SolveAndGrid Solve, grid.

SolveAndTrack Solve, track.

SolveGridTrack Solve, grid, track.

Translate Takes the parameters **BbsToSol** and **SolToBbs**. After reading the *split.dat* either the *Split solution* file is converted to *BlueBird* format (**SolToBbs**), or vice versa (**BbsToSol**).

TranslateOnly Translate *solution* files only without solving, gridding or tracking. The **Translate** command is still needed to give the direction of the translation (*Split* to *BlueBird*, or vice versa).

Silent Solve progress printing mode; iteration summaries are not written on the screen.

Noisy Solve progress printing mode; iteration summaries are written on the screen.

ColdStart Starting mode; all coefficients are set to zero prior to solve.

WarmStart Starting mode; binary file *solution* created by a previous run is required. The coefficients available in file *solution* are used as initial estimates for the coefficients in the current run. This option is useful in sensitivity analysis; if you change the value of one (or a few) parameters by a small amount, the required number of iterations will be reduced if you use **WarmStart** in all runs excluding the initial run. If you add new elements or change the geometry of existing elements, do not use **WarmStart**. **WarmStart** is also useful if you need more iterations to converge than *Split* allows.

MaximumSolveIterations Maximum number of global iterations allowed to solve the problem.

MinimumSolveIterations Minimum number of global iterations for solving the problem.

RelaxationCoefficient Underrelaxation/overrelaxation coefficient.

LocalSolveIterations Number of local iterations. Please see explanation in **Solution Algorithm** section earlier in this document.

CompareHeadOn ASCII input file *head.dat* is required. Please see explanation later.

CompareHeadOff ASCII input file *head.dat* is not required.

Inverse Perform parameter estimation prior to solve. ASCII input file *head.dat* is required. *Split* module *Inverse* writes binary file *solution* regardless of the write solution mode and uses **WarmStart** regardless of the selected starting mode to reduce the required number of iterations.

Remark The remaining part of the line is ignored. Can not be used to disable elements or vertices of elements (please see the instructions in **Analytic Element Parameters** section if you want to disable, but not delete, analytic elements).

* Same as **Remark**.

EndInput Used to stop the input. If omitted, entire input file (e.g. *split.dat*) is read.

OldStyleSolution Enables *Split* to read solution files created by older *Split* versions.

12.2 Output Control

WriteSolutionOn Write solution mode; binary file *solution* will be written by *Split* module *Solve*.

WriteSolutionOff Write solution mode; binary file *solution* will not be written.

Formatted File *solution* will be written as ASCII. Default is binary. *Split* can read either ASCII or binary version of this file automatically, regardless of the format specifier.

BlueBirdsolution Write additional solution file (*solution.bbs*) in *BlueBird* format.

SurferLayoutOn Write the layout to Atlas Boundary ('bna') files (can be loaded by *Surfer*).

SurferLayoutOff Do not write the layout to Atlas Boundary files.

SurferGridsOn Write the grids to *Surfer* grid ('grd') files.

SurferGridsOff Do not write the grids to *Surfer* grid files.

SurferASCIIGrids *Surfer* grid files will be written as ASCII files. If this command is omitted, files will be written as binary.

AtlasTrackOn Write the particle tracks to Atlas Boundary file *tracks.bna*.

AtlasTrackOff Do not write the particle tracks to Atlas Boundary file *tracks.bna*.

AtlasContOn Write the contours to Atlas Boundary ('bna') files.

AtlasContOff Do not write the contours to Atlas Boundary files.

ASCIITrackOn Write the particle tracks to ASCII file *tracks.txt*.

ASCIITrackOff Do not write the particle tracks to ASCII file *tracks.txt*.

ArcInfoGenerateContOn Write the contours to *ArcInfo* generate ('gen') files.

ArcInfoGenerateContOff Do not write the contours to *ArcInfo* generate files.

ArcInfoGenerateTrackOn Write the particle tracks to *ArcInfo* generate file *tracks.gen*.

ArcInfoGenerateTrackOff Do not write the particle tracks to *ArcInfo* generate file *tracks.gen*.

ArcInfoGridsASCIION Write the grids to *ArcInfo* ASCII raster ('asc') files.

ArcInfoGridsASCIIOff Do not write the grids to *ArcInfo* ASCII raster files.

ArcInfoGridsBinaryOn Write the grids to *ArcInfo* binary raster ('ft') files.

ArcInfoGridsBinaryOff Do not write the grids to *ArcInfo* binary raster files.

ArcFlowOutput Sets output for *ArcFlow*. Equivalent to specifying **SurferLayoutOff**, **SurferGridsOff**, **AtlasTrackOff**, **AtlasContOff**, **ASCIITrackOn**, **ArcInfoGenerateContOn**, **ArcInfoGenerateTrackOn**, **ArcInfoGridsASCIIOff**, and **ArcInfoGridsBinaryOff**.

PlotQxQy Creates the grid files qx.grd and qy.grd for the specific discharges in x- and in y-direction.

Analysisface Gives the discharge normal and tangential to a line. Takes six parameters: numerical ID of the line (1,2,...), x- and y-coordinates of start point, x- and y-coordinates of end point, number of evaluation points along the line. Results in *TransectAnalysis.csv*.

Zonebudget Water balance across the sides of a polygon. Result in *ZoneBudget.csv*.

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex.

Line:

Line: enter & to end the vertex list. The polygon will then be closed (it is not necessary to enter the first vertex again).

The descriptions of various output files is included later in this document.

12.3 Aquifer and Model

Split does not require any particular set of units for aquifer and element parameters. You may choose any CONSISTENT set of units when using *Split*. For example, if hydraulic conductivity is measured in meters per second, you must measure distances in meters and time in seconds. Following is the list of aquifer and model parameters used by *Split*.

BaseAndThickness Aquifer base elevation [L] and thickness [L]. Thickness must be positive. Select large thickness if aquifer is unconfined.

Conductivity Aquifer background conductivity [L/T]. Must be positive.

Uniform Uniform flow X and Y components in [L^2/T]. These components are components of vertically integrated (over saturated thickness) specific discharge which is uniform in space and represents regional flow. If you are modeling on a larger (e.g. regional) scale, you should set the uniform flow components to zero. Uniform flow should be used only on a smaller scale (e.g., flow to a single well), to represents the regional flow in a simplified fashion.

Gradient Another way to specify uniform flow. You specify the magnitude of the regional head gradient (dimensionless), the direction of groundwater flow in degrees - the direction of decreasing heads (0 for flow from south to north, 90 for flow west to east, 180 for flow north to south, etc.), and average saturated thickness [L] (aquifer thickness if flow is confined); *Split* will compute the uniform flow components from these parameters and the background hydraulic conductivity of the aquifer. If the input file contains both **Uniform** and **Gradient** command, command **Uniform** will be ignored and the last occurrence of the command **Gradient** will be used.

ReferencePoint Specify X and Y coordinates and head [L] at a reference point. Head should be above the base of the aquifer. For further discussion about the reference point, please see the section **About Global Constant** presented later in this document.

NetExtraction Net extraction [L^3/T] of all elements. Can be specified if at least one head-specified string exists. Otherwise, it is ignored. For explanation of the net extraction, please see section **About Global Constant** later in this document.

Window Xcenter, Ycenter (center of gridding window), grid cell-size [L] (same in both X and Y direction), number of grid cells in X direction, number of grid cells in Y direction. Cell-size must be positive. Number of cells must be positive.

Precision Integer precision parameter. The lowest level (lowest precision) is 1, the highest level (highest precision) is 5. This command is used to define default precision: individual strings may have different precision levels. If a precision level for a string is not specified, the default precision defined here will be used. The detailed interpretation of this parameter is included later in this document.

12.4 Solution Algorithms

Jacobi Use the Jacobi algorithm, instead of the default Gauss-Seidel algorithm.

FluxControl Constrains the fluxes of head-specified elements. Takes number of control points along each head-specified element. **SVD** is used to pre-invert the explicit solution matrix, which is written to *inverse.dat*. This reduces the explicit flux computation to a simple matrix multiplication. Using less than 3 control points with high precision models (levels 4 and 5) can lead to divergence.

WarmSVD Allows the reuse of an existing *inverse.dat*. The matrix can only be reused if the number, order in *split.dat*, and geometry of the head-specified elements remains unchanged.

Superblocks Superblocks are used to increase the computational efficiency of line and circular elements. Takes nesting level (1-4), precision level (1-5), and updating scheme. Both the **Explicit** (?) and the **Implicit** (?) schemes are implemented.

12.5 Particle Tracking and Capture Zones

ForwardTracing Trace particles forward.

ReverseTracing Trace particles backward.

Porosity Must be a positive number not larger than 1.0.

Particle Takes x and y coordinates where a particle is introduced at time=0 and particle name (optional).

BoxOfParticles Takes xmin, xmax, ymin, ymax, nx, ny. Defines a rectangle where nx*ny equally spaced particles are released at time=0.

Time Takes time and a number of rendezvous times. Introduces a sequence of rendezvous times from time specified with previous **Time** command (zero if **Time** is called for the first time) to time specified by present **Time** command. The rendezvous times will be uniformly distributed between previous and present time.

CaptureZone Takes x and y coordinates and an integer number of particles. Finds a well of a positive discharge which is closest to the specified location. Introduces the specified number of equally spaced particles at the well radius, and sets **ReverseTracing**. If no wells of positive discharge exist, particles will be released in the center of the gridding window. Particles that are created by **CaptureZone** command are named. The name of each particle contains the name of the well where particle is released (if well was named) and a unique identifier.

CaptureAll Produces capture zones for all wells of positive discharge. Takes a number of particles. Introduces the specified number of equally spaced particles at the radius of all wells of positive discharge, and sets **ReverseTracing**. Particles that are created by **CaptureAll** command are named. The name of each particle contains the name of the well where particle is released (if well was named) and a unique identifier. If the total number of particles used by **Particle** commands, **CaptureZone** commands, and **CaptureAll** command exceeds the maximum allowed number of particles, the available number of particles (maximum minus particles assigned by **Particle** commands minus particles assigned by **CaptureZone** commands) will be evenly distributed between all wells of positive discharge by **CaptureAll** command.

TravelTime Creates the ASCII output file *ttime.dat*, which shows the start and end locations and the travel time for each particle.

AquiferTop Treats the particles as if they were at the top of the aquifer. This means particles get captured by linesinks regardless of flow across the linesink. If this command is not in *split.dat* the particles are regarded to be at the bottom of the aquifer.

HDWATER Takes x and y coordinate where headwater is located and flux [L^3/T] of water that is introduced by the headwater.

Pipeline Takes x and y coordinates of the start and endpoints of the pipeline. Pipelines are used to connect circular and elliptical lakes to the stream network. Pipelines only transport flux and have impact on the flow solution.

12.6 Parameter Estimation

Tolerance If the relative change of all parameters, that are being estimated, between two subsequent iterations is less than selected tolerance, the iterations of Levenberg-Marquard algorithm are terminated.

MaximumIterations Maximum number of iterations of Levenberg-Marquard algorithm. Minimum number of iterations can not be selected. It was set to 2.

CentralDerivatives Use central derivatives to obtain Hessian matrix.

SidedDerivatives Use sided derivatives to obtain Hessian matrix. In some cases this may be faster, yet less precise than **CentralDerivatives**.

FirstDerivatives Use first derivatives to approximate the Hessian matrix.

SecondDerivatives Use second derivatives to obtain the 'true' Hessian matrix. This is much slower than **FirstDerivatives**, but it may improve the estimate of variance-covariance matrix.

The analytic element parameters (described in the next section) that are optimized are specified with the initial guess and the range (minimum and maximum value). Initial guess must be within the range. If command **Inverse** is not present in the input file, the range is ignored and the initial guess is used in *Split* routine *Solve*.

12.7 Analytic Elements

Each analytic element type is assigned an identifier:

Well for a well

CircularInhomogeneity for a circular inhomogeneity in hydraulic conductivity

CirLake for a circular lake with specified head

EllipticalInhomogeneity for an elliptical inhomogeneity in hydraulic conductivity

EllipticalLake for an elliptical lake with specified head

AreaSink for an area-sink of spatially variable recharge/leakage

Inhomogeneity for a polygonal inhomogeneity in hydraulic conductivity

Head for a head-specified string

HWell for a horizontal well

Reservoir for a reservoir of specified extraction

River for a head and resistance-specified string

Extraction for an extraction-specified string

NormalDischarge for a string where normal discharge is specified on the 'left' side

LeakyWall for a string of leaky or impermeable walls

Drain for a string of drains or cracks

Each analytic element requires a group of lines in the input file. The first line in the group is element identifier. The remaining lines in the group depend on the element type. They are described below and presented in an example of the input file later in this document. If **Disabled** (e.g. **Head Disabled**) is present after the element identifier in the same line, the element will be ignored. Each element may be named. Name must appear after the element identifier in the same line. Name can contain multiple words (e.g. **Well number 1**). If element is disabled, the keyword **Disabled** can appear before element name (e.g. **Well Disabled number 1**), within element name (e.g. **Well number Disabled 1**), or after element name (e.g. **Well number 1 Disabled**). The character string containing the element identifier, name and, if applicable, keyword **Disabled** can not exceed 100 characters.

Well :

Line: x and y coordinates of the well, discharge [L^3/T], well radius. Radius must be positive. Discharge may be negative.

Line: enter & to end the element.

CircularInhomogeneity :

Line: x and y coordinates of the center, hydraulic conductivity [L/T], radius. Radius must be positive. Conductivity can not be negative.

Line: enter & to end the element and precision level (optional).

CirLake :

Line: x and y coordinates of the center, specified head [L], radius. Radius must be positive. Specified head can not be negative.

Line: enter & to end the element and precision level (optional).

EllipticalInhomogeneity :

Line: x and y coordinates of the center, hydraulic conductivity [L/T], long axis, short axis, angle. Conductivity can not be negative.

Line: enter & to end the element and precision level (optional).

EllipticalLake :

Line: x and y coordinates of the center, specified head [L], long axis, short axis, angle. Specified head can not be negative.

Line: enter & to end the element and precision level (optional).

AreaSink :

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex in counterclockwise direction.

Line:

Line: enter & to end the vertex list. The polygon will then be closed (it is not necessary to enter the first vertex again).

Line: x and y coordinates of the first leakage control point and leakage value. Initial guess, range.

Line: x and y coordinates of the second leakage control point and leakage value. Initial guess, range.

Line:

Line: enter & to end the group. You may enter just one leakage control point if constant-strength is desired. Leakage [L/T] will be interpolated using a multi-quadric interpolator (?) between leakage control points, where leakage is met exactly. Recharge is modeled as negative leakage.

Inhomogeneity :

Line: Hydraulic conductivity [L/T]. Can not be negative. Initial guess, range.

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex in counterclockwise direction.

Line:

Line: enter & to end the group and precision level (optional). The polygon will then be closed (it is not necessary to enter the first vertex again).

Head :

Line: x and y coordinates of the first vertex and head value. Head must be above the base.

Line: x and y coordinates of the second vertex and head value. Head must be above the base.

Line:

Line: enter & to end the group and precision level (optional). String is not closed automatically. Head $[L]$ will be interpolated linearly between vertices.

HWell :

Line: Well extraction $[L^3/T]$.

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex.

Line: enter & to end the group and precision level (optional).

Reservoir :

Line: Reservoir extraction $[L^3/T]$. Initial guess, range.

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex in counterclockwise direction.

Line:

Line: enter & to end the vertex list and precision level (optional). The polygon will then be closed (it is not necessary to enter the first vertex again). This element is internally represented as a combination of an area sink and a drain. The ideal drain ensures that the head is constant along the boundary of the reservoir, the area sink ensures that the requested extraction is produced. Head contours inside the reservoir will be created, but they are meaningless; they should be ignored.

River :

Line: Bed thickness and bed width. These parameters must be positive.

Line: Bed hydraulic conductivity $[L/T]$. Can not be negative. Initial guess, range.

Line: x and y coordinates of the first vertex, head value, river depth. Head must be above the base, depth must be positive.

Line: x and y coordinates of the second vertex, head value, river depth. Head must be above the base, depth must be positive.

Line:

Line: enter & to end the group and precision level (optional). String is not closed automatically. Head and depth $[L]$ will be interpolated linearly between vertices.

Extraction :

Line: x and y coordinates of the first vertex and extraction per unit length.

Line: x and y coordinates of the second vertex and extraction per unit length. The same extraction value is applied for both segments that share a vertex. If different values are necessary for different segments, make each segment a separate string.

Line:

Line: enter & to end the group and precision level (optional). String is not closed automatically. Extraction per unit length [L^2/T] will be interpolated linearly between vertices. Recharge is modeled as negative extraction.

NormalDischarge :

Line: x and y coordinates of the first vertex and discharge value.

Line: x and y coordinates of the second vertex and discharge value. Discharge value (vertically integrated specific discharge) [L^2/T] is for the left side of a segment from the first to the second vertex; positive if discharge vector is pointing away from the segment. The same discharge value is applied for both segments that share a vertex. If different values are necessary for different segments, make each segment a separate string.

Line:

Line: enter & to end the group and precision level (optional). String is not closed automatically. Discharge will be interpolated linearly between vertices.

LeakyWall :

Line: Wall thickness. This parameter must be positive.

Line: conductivity of the wall [L/T]. This parameter can not be negative. Initial guess, range.

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex.

Line:

Line: enter & to end the group and precision level (optional). String is not closed automatically.

Drain :

Line: Drain thickness. This parameter must be positive.

Line: conductivity of the drain [L/T]. This parameter must be positive. Initial guess, range.

Line: x and y coordinates of the first vertex.

Line: x and y coordinates of the second vertex.

Line:

Line: enter & to end the group and precision level (optional). String is not closed automatically.

Many circular inhomogeneities can be defined at once using **BoxofCircles** command that takes (in the same line) xmin, xmax, ymin, ymax, mean conductivity, standard deviation of conductivity, mean radius, standard deviation of radius, number of circles. The circular inhomogeneities will be randomly placed in the rectangle defined by xmin, xmax, ymin, ymax. Sizes and conductivities will be generated using a lognormal distribution with specified mean and standard deviation. If a circular inhomogeneity, generated by this random process, intersects any other previously generated inhomogeneity (or is fully contained by any previously generated inhomogeneity) it will be rejected and next trial will be performed. As many as 100,000 trials may be performed for each inhomogeneity.

12.8 Precision

Precision level	Number of terms retained in expansions			
	Area-sink boundaries	Other strings	Circular Inh.	Elliptical Inh.
1	5	2	2	2
2	5	5	10	10
3	10	10	20	20
4	10	20	50	50
5	10	30	100	100

13 Maximum Model Sizes

Split is distributed in three versions depending on the memory requirements. The *medium* sized model should run on machines with 256Mb RAM or more. *Split* versions with different parameters (e.g. larger sizes) are available on request.

Small Model Size	
Number of Wells	100
Number of Circular Elements	1,000
Number of Circular Lakes	25
Number of Elliptical Elements	1,000
Number of Elliptical Lakes	25
Number of Strings	100
Number of String Segments	100
Number of Segments in a String	100
Number of Control Points per linesink constraint	10
Number of Control Points per Area-Sink	100
Number of Particles	100
Number of Rendezvous-times	100
Number of Inverse Parameters	10
Number of Measured Heads	100
Grid Resolution	100
Number of Analysis Faces	10
Number of Zone Budgets	5

Medium Model Size	
Number of Wells	2,000
Number of Circular Elements	5,000
Number of Circular Lakes	100
Number of Elliptical Elements	5,000
Number of Elliptical Lakes	100
Number of Strings	1,000
Number of String Segments	1,000
Number of Segments in a String	1,000
Number of Control Points per linesink constraint	10
Number of Control Points per Area-Sink	100
Number of Particles	100
Number of Rendezvous-times	100
Number of Inverse Parameters	100
Number of Measured Heads	1,000
Grid Resolution	500
Number of Analysis Faces	50
Number of Zone Budgets	10
Large Model Size	
Number of Wells	5,000
Number of Circular Elements	15,000
Number of Circular Elements	500
Number of Elliptical Elements	15,000
Number of Elliptical Lakes	100
Number of Strings	2,500
Number of String Segments	1,000
Number of Segments in a String	1,500
Number of Control Points per linesink constraint	4
Number of Control Points per Area-Sink	100
Number of Particles	4,000
Number of Rendezvous-times	500
Number of Inverse Parameters	100
Number of Measured Heads	1,000
Grid Resolution	500
Number of Analysis Faces	100
Number of Zone Budgets	50

13.1 Default Parameters

control parameter	SolveGridTrack
solve progress printing mode	Noisy
starting mode	ColdStart
MaximumSolveIterations	200
MinimumSolveIterations	10
RelaxationCoefficient	1.0
LocalSolveIterations	1
PotentialTolerance	1.d-8
compare heads	CompareHeadOff
output control parameter	WriteSolutionOn
output control parameter	SurferLayoutOn
output control parameter	SurferGridsOn
output control parameter	AtlasTrackOn
output control parameter	AtlasContOff
output control parameter	ASCIITrackOff
output control parameter	ArcInfoGenerateContOff
output control parameter	ArcInfoGenerateTrackOff
output control parameter	ArcInfoGridsASCIIOff
output control parameter	ArcInfoGridsBinaryOff
BaseAndThickness	0. 1.
Conductivity	1.
Uniform	0. 0.
NetExtraction	0.
Window	0. 0. 0.2 100 100
Precision	5
particle tracking mode	ForwardTracing
Porosity	1.
Tolerance	1.d-4
MaximumIterations	20
central or sided derivatives	CentralDerivatives
first or second derivatives	FirstDerivatives

13.2 File Format

If a line contains more than one parameter, parameters must be separated by single or multiple spaces. Tabs can not be used. Once all parameters in a line are read, the remaining portion of the line is ignored. Real parameters can be entered with a period (e.g. 100.), without a period (e.g. 100), in single precision exponential format (e.g. 100.e-6) or double precision exponential format (e.g. 100.d-6). The following is an example of an input file (e.g. *split.dat*):

```
SolveGridTrack          control parameter
BaseAndThickness    0. 100.  aquifer base, thickness
Conductivity        1.d-5    aquifer hydraulic conductivity
Uniform             1.0d-5 1.0d-5 uniform flow components
```

```

Window      50. 50. 1. 100 100 xcenter, ycenter, grid size, number of grids
Precision           4 precision parameter
rem Precision      5 this line is ignored
* Precision        5 this line is also ignored
Jacobi            use Jacobi approach instead of default Seidel approach
Constraint 1 SVD A1S use flux constraint with one control point per line,
                  the SVD linear solver, and A1S algorithm

Well
75. 15. -0.03 .5d0      xc, yc, Q, radius
&
Well number Disabled 1 (this was name; the well will be disabled)
30. 70. -0.03 .5d0      xc, yc, Q, radius
&
Well
80. 80. -0.03 .5d0      xc, yc, Q, radius
&
Well
95. 95. -0.03 .5d0      xc, yc, Q, radius
&
CircularInhomogeneity
30. 13. 1.d-6 5.        xc, yc, k, radius
&
CircularInhomogeneity this one is impermeable
80. 70. 0.d0 5.        xc, yc, k, radius
&
Inhomogeneity outer one
2.d-6              conductivity
25. 25.            x, y
40. 25.            x, y
40. 50.            x, y
15. 70.            x, y
&
Inhomogeneity inner one
4.d-7              conductivity
27. 30.            x, y
38. 30.            x, y
34. 45.            x, y
&
AreaSink
10. 10.            x, y
70. 10.            x, y
50. 40.            x, y
75. 85.            x, y
10. 70.            x, y
&
20. 20. -.00002     x, y, leakage (control point)
60. 20. -.00003     x, y, leakage (control point)
40. 60. -.00001     x, y, leakage (control point)

```

```

30. 75.  -.00003          x, y, leakage (control point)
&
AreaSink smaller one with constant strength
45. 20.          x, y
70. 40.          x, y
45. 60.          x, y
&
80. 30.  -.00003          x, y, leakage (control point)
&
Leaky    it is really impermeable
0.d0          (conductivity of the wall) / (wall thickness)
30. 85.          x, y
80. 50.          x, y
&
Drain    this one is a crack
0.d0          1/(conductivity of the wall * wall thickness)
5.  5.          x, y
60. 5.          x, y
&
Head
90. 10.  100.          x, y, head
90. 90.  100.          x, y, head
10. 90.  100.          x, y, head
&
Normal
5.  2.  .0001          x, y, discharge
60. 2.  .0001          x, y, discharge
&
Particle 50 50
Time 10000000 100
EndInput

```

14 Input ASCII File: *head.dat*

The input file *head.dat* is used by *Split* to compare the piezometric heads computed by *Split* to measured heads (if **CompareHeadOn** is specified). The file is also used by module *Inverse* to compute model parameters based on measured heads. File *head.dat* contains an arbitrary number of lines. Each line must contain x and y location and piezometric head at that location. Each measurement may be named. If named, the name must appear in the same line, following the head value.

15 Input Control File: *stop*

If this file is present, the execution will stop. The stop will likely not occur instantaneously. *Split* will finish the present solution iteration/inverse iteration/particle track/grid line and then stop. This ensures that the solution and results obtained prior to stopping the engine are properly reported.

16 Output Binary File: *solution*

Split module *Solve* creates this binary file if **WriteSolutionOn** is specified. *Split* module *Inverse* creates this binary file regardless of the solution writing mode.

17 Output ASCII File: *debug.out*

Every time *Split* is executed, file *debug.out* is created in the same directory. If something went wrong (*code* not equal to 1), *debug.out* should be of some help. For example, if there was an error in reading the input file, line in the input file with offensive contents is reported. File *debug.out* also reports the string with the slowest convergence. This file also contains the summary of the convergence progress if **Noisy** option is selected. The first line of *debug.out* gives the error code. If code was not equal to 1, the remaining portion of the line gives the *Split* module (e.g. *Solve*) where *code* is generated. The number (*code*) in that line is an integer which should be interpreted as follows:

- 1 Everything was fine; you do not need to know about other codes which are described below.
- 0 *Split* error. System of equations singular. Save input file (e.g. *split.dat*) and call Igor.
- 1 Run-time error. The errors became too large during solve. Something is wrong with your model. For example, you are specifying head in a zone where conductivity is zero. Typically, the problem is not well posed mathematically.
- 2 Input error. You want more elements than *Split* can handle. Check *Maximum Model Sizes* and reduce the number. Or call Igor for a recompiled version of *Split* where you can choose the limits (free upon request).
- 3 Run-time warning. The maximum number of iterations is reached. If you need more iterations, simply run *Split* again with **WarmStart** selected or change the maximum number of iterations and start again. You must specify **WriteSolutionOn** in the initial run if you want to use **WarmStart**. You may also want to plot the head contours to identify the elements that did not converge. If this does not help, make sure that the problem is well posed. For example, if two head-specified strings share a vertex with different head-values assigned by each string, *Split* will not converge. If this does not help, save the input file (e.g. *split.dat*) and call Igor. File *debug.out* can also be helpful; it reports the convergence for each string and reports the string with the slowest convergence.
- 4 Input error. Syntax error during read, check *debug.out* (see below) for line number in the input file and explanation.
- 6 Input error. Binary solution file can not be read. Call *Solve* before *Grid* or *Track*.
- 7 Input error. File *head.dat* can not be used (file empty or syntax error encountered during read).
- 8 Input error. *Split* module *Inverse* was called, but no parameters are available for optimization.
- 9 Run-time error. The objective function has diverged in module *Inverse*; inverse procedure has failed. Make sure that the problem is well posed.
- 10 Run-time error. The initial guess of parameters can not be used by module *Inverse*.

- 11 *Split* error. Contouring routine has failed. Save input file (e.g. *split.dat*) and call Igor.
- 12 Error occurred while reading *split.dat*
- 13 A required file could not be opened.
- 14 The maximum number of iterations was reached by the linear solver of the flux constraining step.
- 15 problems with superblocks (too many superblocks, too many nestlevels, too high order)

18 Output ASCII File: *progress.out*

Every time *Split* is executed, file *progress.out* is created in the same directory. This file contains the information about the progress of the solution process in the modules *Inverse*, *Solve*, *Grid*, *Track* and *Matrix Inversion*. Each of these modules writes a keyword to the progress file. The keyword and the information are:

solve:

Split is performing iterations to solve for element coefficients.

inverse:

Inverse modeling is being performed.

grid:

The results are being grided.

track:

Particles are being tracked.

SVDinvert:

Split is inverting the matrix for constraining the flux, using the SVD algorithm.

19 Output ASCII File: *extract.dat*

Split modules *Solve* and *Inverse* create this ASCII file. Extractions of line-sinks, area-sinks, circular and elliptical lakes and wells are reported. Net (total) extraction is also reported.

An example follows:

```

.....
HEAD LAKE Theme: Dposlksimp.shp          ID: 2
  0.1765E+03 extraction of side          1
  0.2398E+03 extraction of side          2
  0.1566E+03 extraction of side          3
  0.8486E+02 extraction of side          4
  0.1562E+01 extraction of side          5
-0.6290E+01 extraction of side          6
  0.9240E+02 extraction of side          7
  0.7453E+03 total extraction for HEAD LAKE Theme: Dposlksimp.shp          ID: 2
.....

```

```

HEAD LAKE Theme: Dposlksimp.shp          ID: 3
-0.7176E+03 extraction of side          1
-0.1098E+04 extraction of side          2
-0.1224E+04 extraction of side          3
-0.1439E+04 extraction of side          4
-0.1125E+04 extraction of side          5
-0.4267E+03 extraction of side          6
-0.2280E+03 extraction of side          7
-0.6375E+03 extraction of side          8
-0.4469E+03 extraction of side          9
-0.4948E+03 extraction of side         10
-0.7839E+04 total extraction for HEAD LAKE Theme: Dposlksimp.shp          ID: 3

```

```

-----
the total (net) extraction is -0.7093672E+04
-----

```

20 Output ASCII File: *extract.csv*

Split modules *Solve* and *Inverse* create this ASCII file. Extractions per unit length [L^2/T] are reported at all the control points of each segment which is modeled using **Head**, **River**, **Extraction**, or **NormalDischarge** segment. The file may be imported in *ArcInfo*, *Excel* and in *Surfer* to view the distribution of extraction along each segment and between the segments. For example, this output may be used to differentiate the gaining and losing sections of rivers.

Cumulative extractions [L^3/T] are also reported for same elements. This may be used, for example, to compute flow in a river system due to groundwater discharge to the system. In order to compute correct cumulative extraction (flow in the river), you need to specify nodes in a string used to model the river in the direction of water flow: first node should be the spring, last node the end of the river (e.g. point of discharge to a lake or ocean). If two rivers come together, the same rule applies for both rivers. You may, for example, specify nodes in one of the rivers from its spring all the way to the end, and nodes in the other river from its spring to the point where the rivers come together. If you do this, *Split* will take into account contributions from both river branches and compute correct cumulative flow. The number of various river branches, defined using these rules, is not limited.

If a river originates from a lake, you need to specify nodes in a string that is used to model the lake (e.g. a **Head** string) starting and ending at the node where the river originates. The cumulative discharge (flow) at that location will equal the groundwater discharge to the lake. If this lake receives flow from other rivers, strings that are used to model these rivers can be terminated at any node of the string used to model the lake. Contributions from all rivers will be taken into account for a river system downstream from the lake.

Please note, the current version of *Split* does not terminate the river if the cumulative extraction/flow is zero. *Split* will continue computations even if the flow is negative. Future versions of *Split* will take this condition into account and terminate the rivers once they get dry.

21 Output ASCII File: *errors.dat*

Split modules *Solve* and *Inverse* create this ASCII file. Errors along boundaries of circular inhomogeneities and strings are reported. These errors should be used to evaluate the quality of solution obtained by *Split*, that is, to evaluate the ability of *Split* to solve your model. The appropriateness of your model CAN NOT be evaluated using these errors. To evaluate your model, you should, for example, compare piezometric heads and extractions obtained by *Split* to observed heads and extractions. To reduce errors in *Split* you can increase the **Precision** level, and/or divide long linear segments into a number of smaller ones. The former approach usually improves the solution along segments; the latter improves the solution near vertices.

The reported errors are the largest absolute errors (sampled over a set of control points) along boundaries of elements. Errors along boundaries of elements are defined as:

CircularInhomogeneity : discontinuity in head across the boundary of the inhomogeneity. If conductivity of the inhomogeneity is zero, the error is the discharge potential inside the boundary (which should be zero).

AreaSink : discontinuity in discharge potential for doublets, discontinuity in integrated normal component of discharge for linesinks. For more information about area-sinks, please see ?.

Inhomogeneity : discontinuity in head across the boundary of the inhomogeneity. If conductivity of the inhomogeneity is zero, the error is the discharge potential inside the boundary (which should be zero).

Head : head specified minus head obtained by the model.

River : difference in head between the river and the aquifer minus the product of flow (per unit length) from the river to the aquifer and bed resistance parameter defined earlier. If the head in the aquifer is below the river bottom, the error is computed as a difference between the head elevation in the river and the elevation of river bottom minus the product of flow from the river to the aquifer and bed resistance parameter.

Extraction : extraction specified minus extraction obtained by the model.

NormalDischarge : normal discharge specified minus normal discharge obtained by the model.

LeakyWall : error in normal discharge, computed using:

$$Q_n + \frac{k_w}{kb_w} \Delta\Phi \quad (21.1)$$

where Q_n is the normal discharge, k_w is the hydraulic conductivity of the wall of thickness b_w , and k is the background conductivity. $\Delta\Phi$ is the jump in discharge potential across the wall. For details, see ?.

Drain : error in tangential discharge, computed using:

$$Q_t + \frac{k}{k_d b_d} \Delta\Psi \quad (21.2)$$

where Q_t is the tangential discharge, k_d is the hydraulic conductivity of the drain of thickness b_d . $\Delta\Psi$ is the jump in stream function across the drain. For details, see ?.

The maximum (over all elements) error ('Maxerror') is reported at the end of each iteration. This is typically of limited value, since errors from different types of elements are measured in different units. 'Maxerror' is a very useful measure if all the elements are of the same type.

An example of this file follows:

```

.....
HEAD LAKE Theme: Dposlksimp.shp          ID: 2
 0.1209E-01 largest error for side      1
 0.1215E-01 largest error for side      2
 0.7689E-02 largest error for side      3
 0.4299E-02 largest error for side      4
 0.1895E-02 largest error for side      5
 0.1746E-02 largest error for side      6
 0.6833E-02 largest error for side      7
 0.1215E-01 largest error for all sides in HEAD LAKE Theme: Dposlksimp.shp      ID: 2
.....

HEAD LAKE Theme: Dposlksimp.shp          ID: 3
 0.4629E-01 largest error for side      1
 0.6225E-01 largest error for side      2
 0.6391E-01 largest error for side      3
 0.1071E+00 largest error for side      4
 0.1110E+00 largest error for side      5
 0.5664E-01 largest error for side      6
 0.5918E-01 largest error for side      7
 0.6604E-01 largest error for side      8
 0.4084E-01 largest error for side      9
 0.4274E-01 largest error for side     10
 0.1110E+00 largest error for all sides in HEAD LAKE Theme: Dposlksimp.shp      ID: 3
.....

-----
the largest error for all elements is    0.1109503
-----

```

22 Output ASCII File: *invModel.dat*

Split module *Inverse* creates this ASCII file. The file contains fitted parameters, estimate of variance-covariance matrix and estimate of the correlation matrix.

An example follows:

```

-----
Iteration: 0 Change:.000E+00 Error:.260E+03
Iteration: 1 Change:.000E+00 Error:.650E-03
Iteration: 2 Change:.134E+00 Error:.979E-04
Iteration: 3 Change:.109E+00 Error:.431E-06
Iteration: 4 Change:.809E-02 Error:.266E-08
Iteration: 5 Change:.462E-03 Error:.163E-09
Iteration: 6 Change:.902E-04 Error:.104E-09
Inverse model started at: 15:00:25
Inverse model finished at: 15:01:03
Time spend on inversion : 0: 0:38

```

```

-----
Parameters fitted by the inverse model
-----

```

```

-----
Parameter ID 1; Value -0.20001E-05; Leakage number 1; from AreaSink...
Parameter ID 2; Value -0.29998E-05; Leakage number 2; from AreaSink...
Parameter ID 3; Value -0.10002E-05; Leakage number 3; from AreaSink...
Parameter ID 4; Value -0.39998E-05; Leakage number 4; from AreaSink...
-----

```

```

-----
Estimate of the variance-covariance matrix
-----

```

```

-----
ID          1          2          3          4
-----
1 | 0.14E-21 -0.52E-21 0.53E-21 -0.63E-21
2 | -0.52E-21 0.44E-20 -0.45E-20 0.42E-20
3 | 0.53E-21 -0.45E-20 0.56E-20 -0.62E-20
4 | -0.63E-21 0.42E-20 -0.62E-20 0.78E-20
-----

```

```

-----
Estimate of the correlation matrix
-----

```

```

-----
ID          1          2          3          4
-----
1 | 0.10E+01 -0.66E+00 0.60E+00 -0.60E+00

```

```

2 | -0.66E+00  0.10E+01 -0.91E+00  0.72E+00
3 |  0.60E+00 -0.91E+00  0.10E+01 -0.94E+00
4 | -0.60E+00  0.72E+00 -0.94E+00  0.10E+01

```

23 Output ASCII File: *headerr.dat*

The output file *headerr.dat* is created by the module *Inverse* and by **CompareHeadOn** command. Following the disclaimer, the lines in *headerr.dat* correspond to lines in *head.dat*. Each line contains the x and y location, computed and measured piezometric head at that location. If measurement was named, the name will appear in the same line. The last line in *headerr.dat* contains the number of measurements, sum of the absolute values of errors (measured minus computed heads), and sum of the squared errors. Head measurements that are located inside well-screens are not included.

24 Output Layout Files

Split module *Grid* creates following ASCII Atlas Boundary files if **SurferLayoutOn** is specified:

area.bna: area-sinks.

circle.bna: circular inhomogeneities.

ellipse.bna: elliptical inhomogeneities

bound.bna: gridding window.

lines.bna: line elements.

wells.bna: wells.

heter.bna: polygonal inhomogeneities.

Element names (if specified) are reported. The files may be loaded in *Surfer*.

25 Output Particle Tracking Files

Split module *Track* creates following files:

tracks.bna: Atlas Boundary file created if **AtlasTrackOn** is selected. Particle names (if specified) are reported.

tracks.gen: Particle tracks in *ArcInfo* generate file format, created if **ArcInfoGenerateTrackOn** is selected.

tracks.txt: Similar format as *tracks.bna*; file is created if **ASCIITrackOn** is selected. Time and particle index are printed next to particle locations. Locations and time are separated by commas. Particle names are not printed. The first line in the file is the header. File may be loaded into *ArcView* directly, and in *Surfer* as a classed-post file.

26 Output *Surfer* Grid Files

Split module *Grid* creates following *Surfer* grid files if **SurferGridsOn** is selected:

head.grd: piezometric head, available where conductivity is not zero.

stream.grd: stream function, available where leakage is zero.

leakage.grd: leakage.

27 Output *ArcInfo* Grid Files

Split module *Grid* creates following *ArcInfo* grid files:

head.asc: ASCII raster file (piezometric head, available where conductivity is not zero)

stream.asc: ASCII raster file (stream function, available where leakage is zero)

leakage.asc: ASCII raster file (leakage)

head.ftt: binary raster file (piezometric head, available where conductivity is not zero)

stream.ftt: binary raster file (stream function, available where leakage is zero)

leakage.ftt: binary raster file (leakage)

head.hdr: header file

stream.hdr: header file

leakage.hdr: header file

These files may also be loaded in *ArcGIS* if you have the *Spatial Analyst* extension. The ASCII files are created if **ArcInfoGridsASCIIOn** is selected; the binary and the header files are created if **ArcInfoGridsBinaryOn** is selected.

28 Output Contour Files

Split has the ability to produce contour files directly. Creating contour files directly with *Split* (rather than creating grid files that can be used to produce any set of contours) is useful if you do not have a contouring software (e.g. *Surfer* or *ArcView* with *SpatialAnalyst*). The contouring routine in *Split* follows the *Grid* routine. *Split* does not read the grid files (which are used to produce contours). Hence, if you need contours, you must grid first. By default, *Split* will produce 30 contours from minimum to maximum. If this is not satisfactory, you may specify your own contour levels in *head.lvl*, *stream.lvl* and *leakage.lvl*. The format of these files is simple; each line should contain only one real value: the contour level. Any number of contour levels can be specified. If these files do not exist, they will be created and 30 contour levels (from minimum to maximum) will be written there. The contours are written to the following files:

head.gen: *ArcInfo* generate file (piezometric head, available where conductivity is not zero)

stream.gen: *ArcInfo* generate file (stream function, available where leakage is zero)

leakage.gen: *ArcInfo* generate file (leakage)

head.bna: Atlas Boundary file (piezometric head, available where conductivity is not zero)

stream.bna: Atlas Boundary file (stream function, available where leakage is zero)

leakage.bna: Atlas Boundary file (leakage)

ArcInfo generated files are created if **ArcInfoGenerateContOn** is selected. Atlas Boundary files are created if **AtlasContOn** is selected. The format of *ArcInfo* generate files and Atlas Boundary files (both are ASCII files) is very simple. For example, you can use these files to view contours in *Microsoft Excel*.

29 Output ASCII File: *TransectAnalysis.csv*

The **AnalysisFace** command creates the output *TransectAnalysis.csv*. This file is comma delimited to work well with Microsoft Excel. Each transect will generate several lines of output. The first line for each transect starts with the numerical ID, followed by the coordinates of the start and end point, the number evaluation points along the transect and the maximum flux along the transect. Then one line for each evaluation point is printed. Each of these lines contains the following information: evaluation point ID (1 to number of evaluation points), x and y coordinate of evaluation point, head at evaluation point, flux normal to transect, and flux tangential to transect.

```
Transect, 2, 0.300E+02, 0.300E+02, 0.400E+02, 0.400E+02, 10, 0.70351619E+02
  1, 0.305E+02, 0.305E+02, 0.19076100E+03, -0.11327834E+02, 0.89040543E+01
  2, 0.315E+02, 0.315E+02, 0.19076100E+03, -0.28473948E+01, 0.12830285E+02
  3, 0.325E+02, 0.325E+02, 0.19076100E+03, 0.75590072E+01, 0.17903793E+02
  4, 0.335E+02, 0.335E+02, 0.19076100E+03, 0.21330710E+02, 0.22447409E+02
  5, 0.345E+02, 0.345E+02, 0.19076100E+03, 0.37101651E+02, 0.22665875E+02
  6, 0.355E+02, 0.355E+02, 0.19076100E+03, 0.49988708E+02, 0.18705208E+02
  7, 0.365E+02, 0.365E+02, 0.19076100E+03, 0.58824792E+02, 0.14019150E+02
  8, 0.375E+02, 0.375E+02, 0.19076100E+03, 0.64713601E+02, 0.10063184E+02
  9, 0.385E+02, 0.385E+02, 0.19076100E+03, 0.68445172E+02, 0.69593417E+01
 10, 0.395E+02, 0.395E+02, 0.19076100E+03, 0.70351619E+02, 0.43752318E+01
```

30 Output ASCII File: *ZoneBudget.csv*

The file *ZoneBudget.csv* contains the output from the command **Zonebudget**. Each line represents one zone budget. The following information is given: the numerical ID of the zone budget (0 to number of zone budgets - 1), the flux into the zone, the flux out of the zone, and the water budget of the zone (outflow - inflow).

31 About Global Constant

If you are familiar with the analytic element method, you are likely aware of the 'reference point' phenomenon. Essentially, an additive constant (referred to as the *global constant*) is a solution of the Laplace equation, and can be added to the solution of a groundwater flow problem. If the flow problem contains, for example, only discharge specified wells, the only effect the global constant has

is the shift of the discharge potential everywhere by the same amount. To select a unique solution, piezometric head is specified at one location, which is then referred to as the 'reference point'. If the aquifer is unconfined, the selection of the head at the reference point affects computation of the saturated aquifer thickness everywhere and thus the computation of the SPECIFIC discharge. Discharge is not affected.

However, if model contains head-specified elements, things are quite different. A global constant does not shift the discharge potential everywhere by the same amount. Global constant enters into computation of each head-specified element. The flow problem can be solved (all boundary conditions met) with any value of the global constant. However, a different selection of the constant will yield different 'far-field' behavior of your solution. That is, the solution (model) will be different on scales that are larger than the scale where model is developed. This is because each value of the global constant will yield a different value of the net extraction. The net extraction is the sum of extractions of all wells, line-sinks and area-sinks. The current value of net extraction ('NetExtract') is reported at the end of each iteration.

I prefer specifying the net extraction as a parameter from which global constant is computed, rather than the reference point location and head. I believe that the reference point concept makes sense only when the only effect of the global constant is a shift in discharge potential. If you are modeling a large regional scale, the obvious choice for net extraction is zero (which is default in *Split*). If you are not modeling the large regional scale, but your domain of interest is surrounded by head-specified features, the net extraction value will not change piezometric heads in your domain of interest, but will change discharges of head-specified line-sinks that surround your domain. Net extraction of zero is then as good as any. If you are not modeling the large regional scale, and your domain of interest is not surrounded by head-specified features, good luck. Some of you will likely want to play with the net extraction. This is the reason why net extraction can be specified.

The only penalty of the net extraction concept is that it does not apply (and can not be used) if no head-specified line elements exist (e.g. the only elements are discharge specified wells). The reference point concept should be used then. To make this possible, I have also included the 'traditional' reference point in *Split*. Unless you specify the location of the reference point and the head, the head at the reference point is set to 10 times the aquifer thickness and the location is selected some distance away from the modeled area in the direction perpendicular to the uniform flow. *Split* also allows you to assign the reference point location and head in the case where head-specified elements do exist. However, on a number of occasions when the reference point was specified inside the modeled area (in the presence of head-specified elements), *Split* has diverged. In that case, you must specify the reference point far from your modeled area, or exclude it.

32 Acknowledgement

The authors wish to thank the National Science Foundation (project EAR-0218914; IGERT award DGE-987066) and the Environmental Protection Agency (NCER STAR grant R 82-7961) for providing partial funding for development of this software. The software has not been subjected to any EPA review and therefore does not necessarily reflect the views of the Agency, and no official endorsement should be inferred.