417201

ANALYSIS OF THE GENERATION OF TRANSMISSIVITY FIELDS FOR THE CULEBRA DOLOMITE, WPO # 40517

Approval Page

Author:	A. Marsh Lavenue INTERA Inc.	A. Marsh Lavenue	12/13/96
		Signature	Date
Approved by:	D.R. Anderson Department Manager Organization 6849 WIPP Performance Assessment	D R Anderson	12/13/96
		Signature	Date
Associated Analysis Plan	AP-018 Groundwater Modeling Analysis Plan for the Generation of Transmissivity Fields for the Culebra Flow and Transport Calculations		

December 6, 1996

Information Only

ANALYSIS OF THE GENERATION OF TRANSMISSIVITY FIELDS FOR THE CULEBRA DOLOMITE, WPO # 40517

PI:	A. Marsh Lavenue/INTERA
WBS:	1.2.07.4.1
Associated Analysis Plan:	AP-018 Groundwater Modeling Analysis Plan for the Generation of Transmissivity Fields for the Culebra Flow and Transport Calculations
Date Work Performed:	July, 1996
Date of Package:	Friday, December 6, 1996

page intentionally left blank

TABLE OF CONTENTS

TABLE OF CONTENTS (cont.)

LIST OF FIGURES

FIGURE **PAGE**

LIST OF FIGURES (cont.)

FIGURE **PAGE**

LIST OF TABLES

PREFACE

This analysis package is one of eight packages documenting analyses performed in support of the Compliance Certification Application (CCA) for the Waste Isolation Pilot Plant (WIPP). The following background and overview of the analyses is provided to assist the reader in understanding this analysis package and the overall strategy and framework of these analyses. The reader is also referred to the glossary for further information regarding terms used in this analysis package.

P.1 Background

The WIPP is a geologic repository operated by the U.S. Department of Energy (DOE) for disposal of transuranic radioactive wastes. The repository is located approximately 650 meters underground in the Salado Formation, and is connected to the surface by four shafts which will be sealed after waste emplacement is completed. The geologic formations immediately above and below the Salado are the Rustler and Castile Formations, respectively. The Rustler is considered important because it contains the most transmissive units above the repository; the most significant of these is considered to be the Culebra Dolomite Member. The Castile contains areas of pressurized brine (brine pockets); it is not known whether any such pockets are located under the repository. The area surrounding the shafts and surface facilities and the underlying subsurface are controlled by the DOE.

In October 1996, the DOE submitted the CCA to the U.S. Environmental Protection Agency (EPA) in accordance with the requirements of Title 40 of the Code of Federal Regulations (40 CFR) Parts 191 and 194. The containment requirements in 40 CFR 191.13(a) specify that the disposal system is to be designed to provide a reasonable expectation that radionuclide releases to the accessible environment during 10,000 years are not likely to exceed certain limits (the limits are based on the radionuclide inventory in the repository). The demonstration of having a reasonable expectation is to be based on a performance assessment. Performance assessment (PA) is defined in 40 CFR 191.12:

> *Performance assessment* means an analysis that: (1) Identifies the processes and events that might affect the disposal system; (2) examines the effects of these

> processes and events on the performance of the disposal system; and (3) estimates the cumulative releases of radionuclides, considering the associated uncertainties, caused by all significant processes and events. These estimates shall be incorporated into an overall probability distribution of cumulative release to the extent practicable.

The PA process used in the CCA fulfills these requirements through 6 major steps, listed below.

- (1) Collecting data, characterizing the site and disposal system, and developing the modeling system.

- (2) Constructing scenarios (combinations of possible future events), with and without human activities.

- (3) Estimating the probability that various scenarios will occur.

- (4) Analyzing the consequences of the various scenarios (deterministic futures) which have sufficiently high consequences and probability of occurrence. There are four basic scenarios considered: (1) undisturbed performance (the absence of human intrusion); (2) the E1 intrusion scenario (a borehole which penetrates both the repository and an underlying pressurized brine reservoir in the Castile Formation); (3) the E2 intrusion scenario (a borehole which penetrates the repository); and (4) multiple intrusions (for example, an E2 intrusion followed by an E1 intrusion - E2E1). Each of these scenarios is considered with and without the effects of mining potash located in the Salado within the controlled area.

- (5) Calculating cumulative radionuclide releases and comparing them to regulatory standards in 40 CFR Part 191. The releases are calculated using the consequences of each scenario and their combinations in various (probabilistic) futures. The releases are expressed as complementary cumulative distribution functions (CCDFs), the probability distribution of exceeding normalized cumulative radionuclide releases.

(6) Performing sensitivity analyses to identify the most significant factors.

The PA calculations described in these analysis packages (and in this overview) complete the fourth and fifth steps: analysis of scenario consequences and calculation of CCDFs, respectively. The other steps are addressed elsewhere.

P.2 PA Calculation Strategy

Because of the large number of complex calculations that are required to produce CCDFs, it is not practical, nor is it necessary, to model the total system in a single calculation. Instead, disposal system components and subsystems are modeled (in six separate tasks) to calculate consequences for the undisturbed scenario and for the E1, E2, and E2E1 human intrusion scenarios (with and without mining). Each of these tasks is performed for a set of reference conditions, which include specific intrusion scenarios at certain times. The reference conditions are designed to allow the results of the first six tasks to be incorporated into the CCDF calculations in a seventh task.

To perform the first six tasks, several major computer codes are used to simulate relevant features of the disposal system and calculate scenario consequences. An additional computer code is used to construct the CCDFs in the seventh task. The seven tasks are described in the Analysis Plan for the Performance Assessment Analyses Supporting the Compliance Certification Application (AP-AAD), dated March 8, 1996. They are summarized here, together with their major computer codes. The computer codes and disposal system components addressed in the first six tasks are also shown schematically in Figure P-1.

Task 1

In the first task, overall flow of brine and gas is calculated for undisturbed conditions and for human intrusion scenarios. The flow of brine and gas is calculated in the repository, in the sealed shafts, in the Salado Formation (where the repository is located), and in the human intrusion boreholes. Brine flow in other formations is also calculated (except for the Culebra, which is addressed separately in Task 3 because of its significance as a pathway for long-term releases). Processes which are coupled to brine and gas flow are also included in this task: gas generation in

the repository, disposal room closure and consolidation, brine flow, and effects on the rock surrounding the repository. Creep closure within the waste regions in the repository is represented in this task using a porosity surface describing porosity as a function of time and pressure. These calculations are performed for the set of system reference conditions, and provide results that are used in subsequent disposal system models (Tasks 2 through 6) and also in CCDF construction (Task 7).

The brine and gas flow and coupled repository processes are modeled using version 4.00 of the computer code BRAGFLO. The porosity surface describing closure of the modeled disposal room is generated using the code SANTOS. The codes and disposal system components are shown in Figure P-1. There are two analysis packages associated with this task: *Analysis Package for the Salado Flow Calculations (Task 1) of the Performance Assessment Analyses Supporting the Compliance Certification Application* (WPO#40514) and *Final Porosity Surface Data* (WPO#35697).

Task 2

This task is calculation of the overall long-term transport and radioactive decay of radionuclides from the waste in brine in the Salado and in the overlying Rustler Formation (except for the Culebra, which is addressed in Task 3). The brine flow fields and disposal system model geometry are those calculated in Task 1, and the transport calculations are performed for undisturbed conditions and for human-intrusion scenarios. The radionuclide source concentrations in the brine (the actinide source term) in the repository are the modeled solubilities of the radionuclides contained in the waste. These calculations are performed for the system reference conditions.

The overall transport and decay are calculated using the computer code NUTS for the undisturbed, E1, and E2 scenarios. In simulations of the E1 scenario, NUTS also tracks brine originating in the underlying Castile brine reservoir, including the fraction of Castile brine that has flowed out from the human-intrusion borehole into the waste in the repository. The code PANEL calculates radionuclide concentrations in brine and also radionuclide transport to the Culebra for the E2E1 scenario. In all scenarios, the quantity of brine flowing up the shafts or a degraded exploratory

borehole to the Culebra calculated by BRAGFLO (Task 1), together with the concentration of radionuclides in that brine calculated by NUTS or PANEL (Task 2), is used to determine the quantity of radionuclides released to the Culebra (the Culebra is addressed in Task 3). The radionuclide concentration in brine calculated by PANEL is also used to determine the quantity of radionuclides released to the surface in Task 4. The codes and disposal system components are shown in Figure P-1. The analysis package for this task is *Analysis Package for the Salado Transport Calculations (Task 2) of the Performance Assessment Analysis Supporting the Compliance Certification Application* (WPO# 40515).

Task 3

Detailed fluid flow and radionuclide transport in the Culebra for each scenario are modeled in Task 3. The fluid flow calculations use transmissivity fields that are generated for the Culebra to represent the spatial heterogeneity in flow characteristics which has been observed experimentally. Each scenario may occur with or without potash mining in the Salado in the controlled area; this mining affects the transmissivity of the Culebra. Detailed movement of radionuclides is also calculated using a modeled double-porosity medium for the Culebra, accounting for flow in fractures, diffusion in the matrix, retardation, and radioactive decay. The transport is calculated using a unit source of radionuclides. These calculations are performed for the system reference conditions.

The computer code SECOFL2D calculates fluid flow in the Culebra, using transmissivity fields calculated by the code GRASP-INV (one field in each simulation). The code SECOTP2D calculates radionuclide transport in the Culebra. In Task 7, transport of the unit radionuclide source *in* the Culebra (from this task) is combined with the release *to* the Culebra (calculated in Task 2 using brine flows calculated in Task 1) to determine whether any radionuclides are actually released to the Culebra and subsequently transported through it for each scenario. The codes and disposal system components are shown in Figure P-1. There are two analysis packages associated with this task: *Analysis Package for the Culebra Flow and Transport Calculations (Task 3) of the Performance Assessment Analyses Supporting the Compliance Certification Application* (WPO#40516) and *Analysis of the Generation of Transmissivity Fields for the Culebra Dolomite* (WPO# 40517).

Task 4

Drilling intrusions into the repository (the E1, E2, and E2E1 scenarios) have immediate consequences: they lead to direct releases of material containing radionuclides to the accessible environment at the surface. These consequences are calculated for the system reference conditions in Tasks 4, 5, and 6. The radionuclide content of the materials released is dependent on the time of intrusion and is calculated separately using the system reference conditions.

Task 4 addresses brine containing dissolved radionuclides in the repository that may reach the surface if it is sufficiently pressurized. Short-term flow in the repository is modeled on a scale which includes repository features such as panel closures to calculate brine and gas flow (gas released to the surface is addressed in Task 6). The radionuclide concentration in the brine is calculated in Task 2. The short-term flow in the repository is modeled using version 4.01 of the code BRAGFLO (also referred to as BRAGFLO_DBR to differentiate it from the BRAGFLO code used in Task 1). The modeled geometry in Task 4 is different from the geometry used in the BRAGFLO code in Task 1, to account for the repository features. The initial conditions for Task 4 are provided by the long-term repository conditions calculated in Task 1. The code and modeled system components are shown in Figure P-1. The analysis package for this task is *Analysis Package for the BRAGFLO Direct Release Calculations (Task 4) of the Performance Assessment Analysis Supporting the Compliance Certification Application* (WPO# 40520).

Task 5

Task 5 addresses cuttings and cavings - the second direct release pathway associated with drilling intrusions into the repository (the E1, E2, and E2E1 scenarios). Cuttings and cavings are solid material carried to the surface by the drilling fluid during the process of drilling the borehole: cuttings are materials removed directly by the drill bit, and cavings are materials eroded from the walls of the borehole by the circulating drilling fluid. The code CUTTINGS_S calculates the quantity of material transported to the surface as cuttings for the system reference conditions. The radionuclide content of the materials released is dependent on the time and location of the intrusion; the content is calculated separately (using the results from the reference conditions) during construction of the CCDFs in Task 7. The code and modeled system components are shown in

Figure P-1. The analysis package for this task is *Analysis Package for the Cuttings and Spallings Calculations (Tasks 5 and 6) of the Performance Assessment Analysis Supporting the Compliance Certification Application* (WPO# 40521).

Task 6

Task 6 addresses spallings - the third direct release pathway associated with drilling intrusions into the repository (the E1, E2, and E2E1 scenarios). Spallings are solid materials carried up the borehole by pressurized gas which may be present in the repository at the time of intrusion. The repository pressure and conditions are calculated in Task 1. The code CUTTINGS_S calculates the quantity of material transported to the surface as spallings for the system reference conditions. The radionuclide content of the materials released is dependent on the time of intrusion and is calculated separately (using the results from the reference conditions) during construction of the CCDFs in Task 7. The code and modeled system components are shown in Figure P-1. This task is discussed together with Task 5 in *Analysis Package for the Cuttings and Spallings Calculations (Tasks 5 and 6) of the Performance Assessment Analysis Supporting the Compliance Certification Application* (WPO# 40521).

Task 7

The final task is construction of CCDFs representing futures of the repository and calculation of cumulative releases (this task represents Step 5 in the performance assessment process described in the previous section). There are three parts in this task: (1) determine futures (random sequences of future events that may occur over the next 10,000 years at the WIPP site); (2) estimate the radionuclide releases resulting from these random sequences of future events, using the results of the calculations for each scenario and the reference conditions; and (3) construct a CCDF for each future. In order to efficiently calculate the consequences of multiple futures without repeating Tasks 1 through 6 for each history, the radionuclide releases for each future are calculated by scaling the reference-condition results from the first six tasks.

The computer code CCDF_GF is used to perform the steps in this task, using the results from all the previous tasks and associated computer codes. Task 7 does not address a component of the

disposal system, therefore the CCDF_GF code is not shown in Figure P-1. The analysis package for this task is *Analysis Package for the CCDF Construction (Task 7) of the Performance Assessment Analysis Supporting the Compliance Certification Application* (WPO# 40524).

P.3 PA Computer Calculations

The major computer codes used in the analyses (including CCDF_GF) and the flow of information among them are illustrated in Figure P-2. Combined, Figures P-1 and P-2 illustrate the flow of information through the codes and the relationship between the codes and the physical system being simulated. In the PA calculations, the codes shown in the figures are executed under the requirements of the software configuration management system (CMS or SCMS), which creates and maintains a complete record of the input data and results of each calculation, together with the exact codes and scripts (commands for executing the codes) used to create those results.

Figures P-1 and P-2 show only those codes that perform the bulk of the computational effort related to simulating the significant physical processes occurring within the disposal system. In addition to these codes, a variety of additional codes are used in this performance assessment. These additional codes are used for the transfer of data between codes, preparation of input data and files, model output processing, and similar tasks. Many of these additional codes are also executed within the CMS, and all are qualified for use in these analyses under applicable SNL WIPP quality assurance procedures.

As shown in Figure P-2, there are three major calculation steps in analyzing the consequences of various scenarios (Tasks 1 through 6 in the previous section):

- Preparation of input from submodels (GRASP-INV and SANTOS),
- Latin hypercube sampling (LHS) of the variables in the parameter database that represent subjective uncertainty (such as spatial variability in a disposal system component property or processes), and

- Execution of the codes within the "deterministic futures" box indicated by dashed lines in Figure P-2.

The parameter database is the initial element in the calculation process. The database includes the values of parameters used in performance assessment codes that pertain to the technical aspects of disposal system performance. Parameters pertaining only to the execution of the computer codes (for example, convergence criteria for Newton-Raphson numerical solvers) are generally not included in the database but are recorded in input files and are traceable through the CMS. The parameters in the database fall into two categories: those that are assigned fixed values, and those that are uncertain and are therefore assigned a range of values according to a cumulative distribution function (CDF).

For the analyses of scenario consequences (Tasks 1 through 6), vectors (sets) of parameter values are created from the variable parameters representing subjective uncertainty by LHS of each variable for the set of simulations in the analyses. Each of the fixed parameter values from the database and a vector of sampled parameter values are combined to form a realization (a set of input parameters that are used in one or more of the codes). Each set of input parameters is then propagated through Tasks 1 through 6 (that is, the codes are executed) under four code sequence configurations, one each for the undisturbed performance scenario, the E1 scenario, the E2 scenario, and the E2E1 scenario. In each configuration, the codes are executed sequentially, as shown in Figure P-2.

In this performance assessment, subjective uncertainty is addressed using a LHS sample size no less than a third larger than the number of uncertain parameters: there are 57 sampled parameters (used in one or more of the codes) that represent subjective uncertainty, and they are sampled to create 100 vectors. The entire process (LHS of uncertain parameters, creation of vectors, and evaluation of scenario consequences through execution of the codes) is repeated three times (each time comprises a replicate which is independent of the other replicates) to achieve confidence in the results.

Once the consequences of various scenarios are calculated, there are two major steps in evaluating consequences of probabilistic futures (Task 7):

- Random sampling of parameters which address stochastic uncertainty (such as location of an intrusion borehole), and

- Execution of the code in the "probabilistic futures" box (CCDF_GF) in Figure P-2, in which the releases for the futures are calculated using the results of the calculations for each scenario and the reference conditions, and a CCDF is constructed for each future.

This sequence of two steps is repeated once for each of the 100 vectors of uncertain parameters (that is, all the random sequences of future events that may occur over the next 10,000 years at the WIPP site are considered for each of the vectors). This yields a group (family) of 100 CCDFs (one for each of the vectors). The family arises from the fact that fixed, but unknown, quantities are needed in the estimation of each CCDF (these quantities are the uncertain parameters in each vector).

Each individual CCDF displays the effects of stochastic uncertainty in that the stepwise shape of the CCDF reflects the fact that a number of different occurrences have a real possibility of taking place. The variations between the individual CCDFs in the family display the effects of subjective uncertainty. The distribution of CCDFs in the family thus provide a complete display of both stochastic and subjective uncertainty.

In the final step, the family of CCDFs for each replicate is compared to the regulatory standard in 40 CFR 191.13(a) to determine compliance.

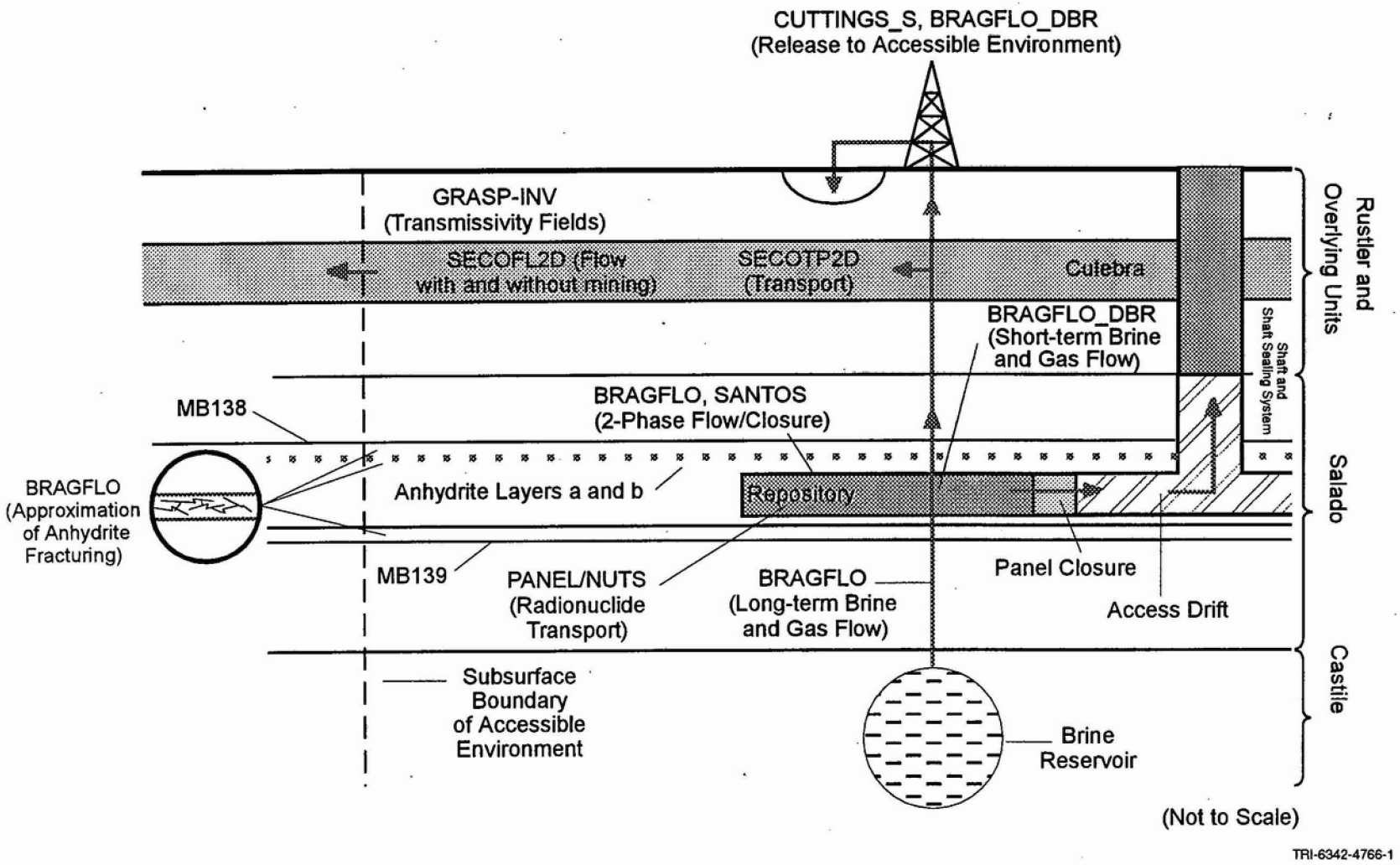

Figure P.1. Schematic side view of the disposal system associating Performance Assessment codes with the components of the disposal system each code simulates.

```
 END;
END; {loadconfigfile}


 PROCEDURE OpenWindowFile;
{============================}
{Global variables: windowfile, windowfilename }

VAR pointpos,err_flag        : INTEGER;

BEGIN

 {check whether filename specified}
 IF Length(windowfilename)<1 THEN BEGIN
   WriteLn(prot,'+++ Error: Windowfile not specified ...',lfc,lfc);
   HALT;
 END;

 {open file}
 Assign(windowfile, windowfilename);
 {$I-} Reset(windowfile); {$I+}
 IF IOResult<>0 THEN BEGIN
  WriteLn(prot,'+++ Error: Unable to open Windowfile ...',lfc,lfc);
  HALT;
 END;

 {setup initial time window}
 time.min  := 0.0;
 time.mean := 0.0;

END; {openwindowfile}


 PROCEDURE GetSigma;
{==================}
{Get Sigma from command line if specified (4. parameter)}
VAR err_flag : INTEGER;
   sigma   : REAL;
BEGIN
 IF ParamCount>3 THEN BEGIN
   err_flag:=0;
   Val(ParamStr(4),sigma,err_flag);
   IF err_flag=0 THEN BEGIN
```

```
      value.sigma:=sigma;
      WriteLn(prot,lfc,'PONS: USING SIGMA SPECIFIED IN COMMAND LINE : ',
              sigma:10:2,lfc );
    END ELSE BEGIN
      WriteLn(prot,lfc,'+++ Warning: Specified uncertainty no value !');
      WriteLn(prot,   '          Default value will be used ... ',
              lfc,lfc,lfc);
    END;
  END;
END;



 PROCEDURE ReadData;
{==================}
{ Global Variables:   wellname,time,value}
VAR ia    :INTEGER;
   iwchar : CHAR;
   swap_buffer : REAL;
BEGIN
 wellname:='';
 time.new:=0.0;
 value.new:=0.0;

 FOR ia:=1 TO wellname_length DO BEGIN
   Read(inputfile,iwchar);
   wellname:=wellname+iwchar;
 END;
 {$I-} ReadLn(inputfile,time.new,value.new);  {$I+}
 IF IOResult<>0 THEN BEGIN
   WriteLn(prot,lfc,'+++ ERROR: in Inputfile ...');
   WriteLn(prot,   '        unable read to record ',stat.nrdata+1:7);
   WriteLn(prot,lfc,lfc,lfc,lfc);
   HALT;
   Close(inputfile);
   Close(outputfile);
 END;

 IF transpose_flag = 'Y' THEN BEGIN
   swap_buffer := time.new;
   time.new   := value.new;
   value.new  := swap_buffer;
   {WriteLn(Con,time.new:10:2,value.new:10:2); }
 END;
```

```
Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

END; {ReadData}


 PROCEDURE WriteData;
{==================}
{Global Variables: wellname,time,value,stat}
VAR printflag : INTEGER;

BEGIN
 {prepare output}
 IF time.cmode='M' THEN time.mean:=time.mean/stat.ndata;
 value.mean:=value.mean/stat.ndata;
 value.min :=value.min-value.sigma;
 value.max :=value.max+value.sigma;

 {clipping}
 printflag:=1;
 IF clip.flag=1 THEN BEGIN
  IF value.min < clip.minvalue THEN value.min :=clip.minvalue;
  IF value.mean< clip.minvalue THEN value.mean:=clip.minvalue;
  IF value.max < clip.minvalue THEN printflag := 0;

  IF value.min > clip.maxvalue THEN printflag := 0;
  IF value.mean> clip.maxvalue THEN value.mean:=clip.maxvalue;
  IF value.max > clip.maxvalue THEN value.max :=clip.maxvalue;

  IF time.mean < clip.mintime  THEN printflag := 0;
  IF time.mean > clip.maxtime  THEN printflag := 0;
 END;

 IF printflag=1 THEN BEGIN
   stat.nwdata:=stat.nwdata+1;
   WriteLn(Outputfile,time.mean:15:2,value.max:12:2,value.min:12:2,
           value.mean:12:2,stat.ndata:10,'  ',wellname);
 END;
 stat.ndata:=0;
END; {writedata}


 PROCEDURE CheckWellName;
{======================}

BEGIN
```

```
  {catch different wellnames}
  IF old_wellname<>wellname THEN BEGIN

    {special case: first record }
    IF old_wellname='initial value' THEN BEGIN
      WriteLn(prot,lfc,'PONS: READING WELLDATA OF ',wellname);

    {different well names}
    END ELSE BEGIN
      WriteLn(prot);
      WriteLn(prot,'+++ WARNING: Record ',stat.nrdata:10);
      WriteLn(prot,'    new wellname: ',wellname);
    END;

    {Store new wellname}
    old_wellname:=wellname;
  END;
END; {CkeckWellName}



 PROCEDURE PrintStack;
{====================}
{Global Variable : stack,stacklen}
VAR  ia :INTEGER;
    t,year,month,day,hour,min,sec : REAL;
    marker : STRING[3];
BEGIN
  WriteLn(prot,       ' Time (s)    ( dd-mm-yy  hh:mm:ss )      Value',lfc);
  FOR ia:=stacklen DOWNTO 1 DO BEGIN

    {transform time}
    t    := stack[ia].time+time.base+time.offset ;
    year := INT(t/31536000.0);                {365  days/year}
    t    := t-31536000.0*year;
    month:= int(t/2635200.0);                 {30.5 days/month}
    t    := t-2635200.0*month;
    day  := INT(t/86400.0);
    t    := t-86400.0*day;
    hour := INT(t/3600.0);
    t    := t-3600.0*hour;
    min  := INT(t/60.0);
    sec  := t-60.0*min;

    {print data record}
```

```
   marker:='  ';
   IF (ia=stackcheck) AND (stack[ia].flag>0) THEN BEGIN
     marker      :=' * ';
     stack[ia].flag:=0;
   END;
   WriteLn(prot,stack[ia].time:12:2,'  ( ',day:2:0,'-',month:2:0,'-',year:2:0,
                          '  ', hour:2:0,':',min:2:0,':', sec:2:0,
                  ' )  ',stack[ia].value:12:2,marker:3);
 END;
END; {PrintStack}



 PROCEDURE PrintOverlap;
{=======================}
{Global Variables: stack,stat}
VAR  irecord :INTEGER;
BEGIN
  irecord:=stat.nrdata-stackcheck+1;
  WriteLn(prot,lfc,lfc,'+++ WARNING: Record ',irecord:10);
  WriteLn(prot,       '   New time smaller than previous time',lfc);
  PrintStack;
END; {PrintOverlap}



 PROCEDURE CatchOffshot;
{=======================}
{Global Variables: stack,stat}

{ Chi-Square Table for 99% confidencial limit  }
{ Ref: M.R.Spiegel (1975), Appendix E, page 347}
CONST chi_square_table : ARRAY[1..25] OF REAL =
                  ( 0.000, 0.010, 0.0717, 0.207, 0.412,
                    0.676, 0.989, 1.34, 1.73, 2.16,
                    2.60, 3.07, 3.57, 4.07, 4.60,
                    5.14, 5.70, 6.26, 6.84, 7.43,
                    8.03, 8.64, 9.26, 9.89, 10.5);
    ny_max    = 25;
    stddev_min = 0.1;

VAR ia,ny,irecord,stackfill :INTEGER;
   valuedev,sum,mean,stddev,stddev_max,chi: REAL;

BEGIN
```

```
 {establish situation in stack and set pointers}
 stackfill:= stat.nrdata;
 IF stackfill>stacklen THEN stackfill:=stacklen;

 {perform ckeck only if there are at <stackcheck> values in stack }
 IF stat.nrdata>=stackcheck THEN BEGIN

   {calculate sum and mean, except record to be checked }
   sum:= -stack[stackcheck].value;
   FOR ia:=1 TO stackfill DO sum:=sum+stack[ia].value;
   mean := sum/(stackfill-1);

   {calculate standard deviation, except record to be checked}
   valuedev :=  Abs(stack[stackcheck].value-mean);
   stddev   := -Sqr(valuedev);
   FOR ia:=1 TO stackfill DO stddev:=stddev+Sqr(stack[ia].value-mean);
   stddev:=sqrt(stddev)/(stackfill-1);

   {estimate max. possible standard deviation using chi-square}
   ny:=stackfill-2;
   IF ny>ny_max THEN ny:=ny_max;
   chi:=sqrt(chi_square_table[ny]);
   stddev_max:= stddev*sqrt(stackfill-1)/chi;

   {use 2.58 standard deviations as criterion (99% confidence limit)}
   stddev_max:= stddev_max *2.58;

   {set lower detection limit}
   IF stddev_max<stddev_min THEN stddev_max:=stddev_min;

   IF valuedev>stddev_max THEN BEGIN
     irecord:=stat.nrdata-stackcheck+1;
     WriteLn(prot,lfc,lfc,'+++ WARNING: Record ',irecord:10);
     WriteLn(prot,      '  Value outside 99% confidence interval',lfc);
     WriteLn(prot,      '  Mean value            : ',mean:10:3);
     WriteLn(prot,      '  2.58 Standard deviations : ',stddev_max:10:3);
     WriteLn(prot,      '  Point Deviation       : ',valuedev:10:3,lfc);

     stack[stackcheck].flag:=1;
     PrintStack;
   END;
 END;
END; {catchoffshot}
```

```
 PROCEDURE NewTimeWindow(VAR time:TimeType);
{=========================================}
{ Setup a new timewindow}

BEGIN WITH time DO BEGIN

 IF wmode='E' THEN BEGIN  {externally fixed time window}

   {read records from window file until upper window limit
   larger than last input time encountered}
   REPEAT BEGIN
    min := max;

    {catch EOF in windowfile}
    IF EOF(windowfile) THEN BEGIN
     WriteLn(prot,lfc,'+++ ERROR: unexpected end of ');
     WriteLn(prot,   '   the window file encountered ... ');
     WriteLn(prot,lfc,lfc,lfc,lfc);
     Close(inputfile);
     Close(outputfile);
     HALT;

    {read next record from window file}
    END ELSE BEGIN
     {$I-} ReadLn(windowfile,max);  {$I+}
    END;

    {catch error during read}
    IF IOResult<>0 THEN BEGIN
     WriteLn(prot,lfc,'+++ ERROR: unable to read next ');
     WriteLn(prot,   '   time step from window file ... ');
     WriteLn(prot,lfc,lfc,lfc,lfc);
     Close(inputfile);
     Close(outputfile);
     HALT;
    END;
   END UNTIL max > new;

   {user message with new window}
   WriteLn(prot,lfc,'PONS: New external time window (',
              min:12:2,' - ',max:12:2,')');
 END;

 IF wmode='F' THEN BEGIN  {internal fixed time window}
```

```
    min  := window * Trunc(new/window);
    max  := min + window;
  END;

  IF wmode='V' THEN BEGIN  {internal variable time window}
    min  := new;
    max  := new+window;
  END;

  IF cmode='B' THEN mean := min;              {backward timing}
  IF cmode='C' THEN mean := 0.5*min+0.5*max;  {centered timing}
  IF cmode='F' THEN mean := max;              {foreward timing}
  IF cmode='M' THEN mean := new;              {average  timing}

END; {with}
END; {NewTimeWindow}



 PROCEDURE StoreData;
{====================}
{Global Variables: stat,time,value,wellname,old_wellname}

VAR ia : INTEGER;
BEGIN {storedata}

  {store new data in stack}
  FOR ia:=stacklen DOWNTO 2 DO BEGIN
    stack[ia]:=stack[ia-1]; END;
  stack[1].time  :=time.new;
  stack[1].value :=value.new;
  stack[1].flag  :=0;

  {catch overlapping data}
  IF (time.new<time.old) AND (stat.ndata>0) THEN BEGIN
    stack[1].flag :=1;  {mark data in stack}
    WriteData;          {close actual time window}
  END;

  {print overlapping data}
  IF stack[stackcheck].flag=1 THEN PrintOverlap;

  {catch obvious offshots}
  CatchOffshot;
```

Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

```
  {ckeck new wellname}
  CheckWellName;

  {store new time}
  time.old:=time.new;


  IF stat.ndata>0 THEN BEGIN

   IF time.new>time.max THEN BEGIN
     WriteData;
   END ELSE BEGIN
     IF time.cmode='M' THEN time.mean:=time.mean+time.new;
     value.mean:=value.mean+value.new;
     IF value.new<value.min THEN value.min:=value.new;
     IF value.new>value.max THEN value.max:=value.new;
     stat.ndata:=stat.ndata+1;
   END;
 END;

 {new timewindow}
 IF stat.ndata=0 THEN BEGIN
   stat.ndata:=1;
   NewTimeWindow(time);
   value.min :=value.new;
   value.max :=value.new;
   value.mean:=value.new;
  END;

 {last input line read}
 IF EOF(inputfile) AND (stat.ndata>0) THEN BEGIN
    WriteData;
    stat.ndata:=-1;
 END;
END; {storedata}


{Main Variables}
{==============}
VAR ia : INTEGER;


BEGIN {main}

 {initialize default user output: DOS standard handle #1}
```

```
protstr := 'Out:';
InitOutput(protstr);
WriteLn(prot,lfc,'DATA ABRIDGING UTILITY CODE PONS');
WriteLn(prot,   '================================');
WriteLn(prot,    Codeversion,lfc);

{open datafiles using command line parameters}
OpenFiles;

{load configuration file}
LoadConfigFile;

{open file with external time windows}
IF time.wmode='E' THEN OpenWindowFile;

{get over ride sigma from command line}
GetSigma;

{setup initial values}
stat.nrdata:=0;
stat.nwdata:=0;
stat.ndata :=0;
old_wellname:='initial value';

FOR ia:=1 TO stacklen DO BEGIN
  stack[ia].time :=0.0;
  stack[ia].value:=0.0;
  stack[ia].flag :=0;
END;

{read inputfile}
WHILE NOT EOF(inputfile) DO BEGIN
  ReadData;
  stat.nrdata:=stat.nrdata+1;
  time.new:=time.new - time.offset;
  StoreData;
END;
StoreData;
close(inputfile);
close(outputfile);

WriteLn(prot,lfc,lfc,'PONS: NORMAL TERMINATION');
WriteLn(prot,lfc,'    Records read    : ',stat.nrdata:7);
WriteLn(prot,   '    Records written : ',stat.nwdata:7,lfc,lfc);
```

END. {main}

Input files for PONS utility code, WIPP PA 96

Pons.cnf Input File (for FW Head data prior to 1995), WIPP PA 96

CMAXTIME 2.E+10
CMAXVALUE 2.E+10
CMODE F
OFFSET 1.262304E+08
WMODE E
WNAME TSIDAT.96

Pons2.cnf Input File (for 1995 and 1996 FW Head data), WIPP PA 96

CMAXTIME 2.E+10
CMAXVALUE 2.E+10
CMODE F
OFFSET 0.0
WMODE E
WNAME TSIDAT.96

Reduce.bat Dos Batch File, WIPP PA 96

PONS H-1P.DAT * PONS.CNF
PONS TH-1P.DAT * PONS.CNF
PONS H-2CP.DAT * PONS.CNF
PONS TH-2CP.DAT * PONS.CNF
PONS H-3B1P.DAT * PONS.CNF
PONS TH-3B2P.DAT * PONS.CNF
PONS H-6BP.DAT * PONS.CNF
PONS TH-6BP.DAT * PONS.CNF
PONS H-11B2P.DAT * PONS.CNF
PONS TH11PT1P.DAT * PONS.CNF
PONS H-14P.DAT * PONS.CNF
PONS H-15P.DAT * PONS.CNF
PONS H-17P.DAT * PONS.CNF
PONS H-18P.DAT * PONS.CNF
PONS P-17P.DAT * PONS.CNF

Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

```
PONS TP-17P.DAT   * PONS.CNF
PONS W-12P.DAT    * PONS.CNF
PONS TW-12P.DAT   * PONS.CNF
PONS W-13P.DAT    * PONS.CNF
PONS TW-13P.DAT   * PONS.CNF
PONS W-18P.DAT    * PONS.CNF
PONS TW-18P.DAT   * PONS.CNF
PONS W-19P.DAT    * PONS.CNF
PONS TW-19P.DAT   * PONS.CNF
PONS W-21P.DAT    * PONS.CNF
PONS TW-21P.DAT   * PONS.CNF
PONS W-22P.DAT    * PONS.CNF
PONS TW-22P.DAT   * PONS.CNF
PONS W-30P.DAT    * PONS.CNF
PONS DOE-1P.DAT   * PONS.CNF
PONS TDOE-1P.DAT  * PONS.CNF
PONS DOE-2P.DAT   * PONS.CNF
PONS TDOE-2P.DAT  * PONS.CNF
PONS ER-9P.DAT    * PONS.CNF
PONS TER-9P.DAT   * PONS.CNF
PONS CB-1P.DAT    * PONS.CNF
PONS D-268P.DAT   * PONS.CNF
PONS  doe1_95.dat * PONS2.CNF
PONS  doe2_96.dat * PONS2.CNF
PONS  er9_95.dat  * PONS2.CNF
PONS  h11b1_95.dat * PONS2.CNF
PONS  h15_95.dat  * PONS2.CNF
PONS  h17_95.dat  * PONS2.CNF
PONS  h18_96.dat  * PONS2.CNF
PONS  h19b0_95.dat * PONS2.CNF
PONS  h1_95.dat   * PONS2.CNF
PONS  h3b2_95.dat * PONS2.CNF
PONS  w21_95.dat  * PONS2.CNF
PONS  wp12_96.dat * PONS2.CNF
PONS  wp13_96.dat * PONS2.CNF
PONS  wp18_96.dat * PONS2.CNF
PONS  wp19_96.dat * PONS2.CNF
PONS  wq1_96.dat  * PONS2.CNF
PONS  wq3_96.dat  * PONS2.CNF
PONS  wq4_95.dat  * PONS2.CNF
PONS  wq5_95.dat  * PONS2.CNF
ppons obsgrasp.96 ppons.wll tsidat.96
```

Appendix 10.
PPONS utility code for building OBSGRASP file

Purpose: PPONS is a post processor for PONS (Appendix 9). Its purpose is to provide a link between the PONS output files (labeled as *.ABR) and the input file OBSGRASP of GRASP-INV. GRASP-INV requires that the mean data values for all the wells be grouped together under each time period. Thus PPONS reads the PONS output files, extracts the mean data values from selected time intervals, rearranges the records such that all mean values for a given period are grouped together, and writes this information to the OBSGRASP format file.

Input: PPONS reads the output files of PONS (one for each well) as specified by the user, and also requires an additional configuration file that designates the files to be read. The input file for PPONS lists all of the files used in assembling the OBSGRASP file. The steady-state data, listed in dataset 1 of OBSGRASP.96, was edited into the file since the steady-state heads were picked from the hydrographs (see Analysis WriteUp). This first dataset corresponds to time=0.0 in the WIPP application (i.e, steady-state).

The PPONS command-line parameters are as follows:

PPONS <outputfile> <wellfile> <windowfile> [<skip>]

outputfile of PPONS. It must be a valid DOS filename. If the file already exists when starting PPONS, it will be overwritten. Otherwise a new file will be created.

wellfile (.WLL) must contain the names of the PONS output files (.ABR) to be used by PPONS. In addition, the file must also contain the grid block indices of the corresponding wells (as used in the SWIFT/GRASP model). Each record must be formatted as follows:

<datafile> <i> <j> <k> [<any string>]

datafile denotes a PONS output file (.ABR). It must be a valid DOS file name. Its maximum length is 12 characters. The parameters *i*, *j*, and *k* represent the grid block indices of the corresponding well (Format: integer values. starting at column 13 or later, separated by one or more blanks). The grid block indices may be followed be any string which will be ignored be PPONS.

windowfile The window file must contain the time intervals for which the mean data values are to be extracted by PPONS from the PONS output files (.ABR). The well file must be a standard ASCII file. The format of the individual records must be as follows:

<time value> [<any string>]

The time value denotes the time interval (Format: any valid numeric value, followed by one or more blanks). The blank after the time value may be followed be any string which will be ignored be PPONS.

skip must be an integer value. It defines whether every time interval (skip=1) or only every n^{th} time interval (skip > 1) specified in the windowfile will be used. If missing, a default value of 1 will be used. In any case, the first time window will be used (e.g., for skip = 5, the used time intervals are 1, 6, 11, 16, ...).

The command line for PPONS application to WIPP PA 96 is as follows:

ppons obsgrasp.96 ppons.wll tsidat.96

Obsgrasp.96 and ppons.wll are listed later in this appendix. Tsidat.96 is summarized in Table 3.4 in the main report.

Output: Extracts the mean data values of selected time intervals, rearranges the records such that all the mean values of a given time interval are grouped together, and writes this information to a single output file. The output file can then be used directly as the OBSGRASP file.

Verification: The PPONS code will read the *.ABR files and combine them, as described above. Table 8 lists the input file to PPONS, PPONS.INP, for the test case continued from PONS. It contains the name of H-18's and D-268's *.ABR data filename and the model indicies associated with the H-18 and D-268 locations. Compare the average pressures in Table 6 and 7 to those in Table 9, which with the exception of less than 1 Pa round off are identical. Also note that only times 1, 4, and 5 have entries, which agrees with the time periods in the previous appendix discussion of PONS. This demonstrates that the PPONS code correctly assembles the H-18 and D-268 ABR data into the appropriate format as needed by GRASP-INV's OBSGRASP.INP file.

Table 8. PPONS Test Case Input file, PPONS.INP

```
H-18P.ABR   26 66  1
D-268P.ABR  25 20  1
```

Description: Borehole Abridged Data Filename, I,J,K Model Indicies of Borehole Location

Table 9. Output file from PPONS, PPONS.OUT
(Output file contains, Data Set Number and Time Step Number - Line 1)
(I,J,K Indices, Mean Observed Pressure During Time Interval, and Label - Line 2)
(0 (to end data set) - Line 3)

```
 1   1
26   66   1    1.034146E+06    H-18P.ABR
 0
 2   4
26   66   1    1.018368E+06    H-18P.ABR
25   20   1    3.114345E+05    D-268P.ABR
 0
 3   5
25   20   1    3.109940E+05    D-268P.ABR
 0
 0
```

Since the steady-state data was added to the file manually the following check calculation is provided. Pressure head (which is the state variable in SWIFT II) is computed from formation elevation and total freshwater head as:

P = fluid column height*gravity*density

where the center elevation of the Culebra is 826.13 m (Cauffman et al., 1990), gravity is 9.792 m/s^2, and density is that of freshwater 1000 kg/m^3

P = (921.6 m - 826.13 m) (9.792 m/s^2) (1000 kg/m^3) = 934842.2, which is identical to the entry for H-1 in OBSGRASP.96. By inspecting Figures 2.2 and 3.5 in the main text the grid indices of H-1 also correct.

Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

PPONS Source Listing

```
{ ============}

{ POSTPROCESSOR FOR DATA ABRIDGING UTILITY CODE PONS
 ================================================

 Code Evolution:
 ---------------
 Development Version   03-Mar-89    QA-CIN 097B-24C-001B


 Code Custodian: Andreas Haug, INTERA Technologies Inc., Austin, TX
 ---------------


}
{Compiler Directives}
{====================}
{$U+}  {Interrupt Option on}


{Global Constants}
{=================}

CONST  codeversion = '     Sandia QA CIN 097B-24C-001B';

    lfc       = #13#10;      {line feed character}
    lm        = '   ';     {left margin}

    max_window    = 500;    {maximum number of time windows used}
    max_well      = 100;    {maximum number of wells used}
    max_namelength =  12;    {maximum length of well names}

{Global Types}
{=============}

TYPE  Str80       = STRING[80];

   WindowType   = RECORD
                  pos  : ARRAY[1..max_window] OF INTEGER;
                  timc : ARRAY[1..max_window] OF REAL;
                  flag : ARRAY[1..max_window] OF INTEGER;
                  act  : INTEGER;
```

```
            last : INTEGER;
          END;

  WellType    = RECORD
          name  : ARRAY[1..max_well] OF STRING[max_namelength];
          i,j,k : ARRAY[1..max_well] OF INTEGER;
          act   : INTEGER;
          last  : INTEGER;
          END;

  BufferType  = RECORD
          name  : STRING[max_namelength];
          time  : REAL;
          head  : REAL;
          i,j,k : INTEGER;
          END;



{Global Variables}
{================}

VAR bufferfile          : FILE OF BufferType;
  outputfile          : TEXT;  {output file}
  wellfile            : TEXT;  {list of wells}
  windowfile          : TEXT;  {external windows}
  User                : TEXT;  {user interface}

  User_string         : Str80; {string specifying user output}

  window              : WindowType;
  well                : WellType;

{-------------------------------------------------------------------}

 PROCEDURE Help;
{==============}
BEGIN
 WriteLn(User,lfc,lfc,lm,'HELP:');
 WriteLn(User,   lm,'-----');
 WriteLn(User,   lm,'The Command-Line Parameters of PPONS are:');
 WriteLn(User,lfc,lm,'PPONS <outputfile> <wellfile> <windowfile> [<skip>] ');
 Close(User);
 HALT;
END; {help}
```

Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

```
{-------------------------------------------------------------------}

 PROCEDURE CombineWell(VAR well:WellType; VAR window:WindowType);
{=====================================================================}
VAR datafile         : TEXT;
   ia              : INTEGER;
   i,j,k           : INTEGER;
   name              : STRING[max_namelength];
   dummy,head,time    : REAL;
   found_flag        : INTEGER;
   buffer           : BufferType;

BEGIN

  {setup buffer record}
  buffer.name := well.name[well.act];
  buffer.time := 0.0;
  buffer.head := 0.0;
  buffer.i    := well.i[well.act];
  buffer.j    := well.j[well.act];
  buffer.k    := well.k[well.act];


  {open data file}
  Assign(datafile,buffer.name);
  {$I-} Reset(datafile); {$I+}
  IF IOResult<>0 THEN BEGIN
    WriteLn(User,lfc,lfc,'+++ Error: Unable to open Datafile ',buffer.name,' ...');
    Close(User);
    HALT;
  END;

  {reset pointer to first time interval}
  window.act:=1;
  {read all data from data file}
  WHILE NOT EOF(datafile) DO BEGIN
    {read record}
    ReadLn(datafile,time,dummy,dummy,head);
    {conmpare record time with time window in window stack}
    found_flag:=0;
    IF time=window.time[window.act] THEN found_flag:=1;

    {search for matching time in time window}
    IF found_flag=0 THEN BEGIN
```

```
    {reset pointer to first time interval}
    IF time<window.time[window.act] THEN window.act:=1;

    {search foreward}
    WHILE (time>window.time[window.act]) AND (window.act<window.last) DO
     window.act:=window.act+1;
   END;
   IF time=window.time[window.act] THEN found_flag:=1;

   {write record to file}
   IF found_flag = 1 THEN BEGIN
    buffer.time := time;
    buffer.head := head;
    window.flag[window.act] := 1;
    Write(bufferfile,buffer);
   END; {if}

 END; {while}
 Close(datafile);
END; {combinewell}

{-------------------------------------------------------------------}

 PROCEDURE GenerateTimeSteps(VAR setnumber: INTEGER; VAR window: WindowType);
{=================================================================================
}
VAR time          : REAL;
   buffer         : BufferType;


BEGIN
 {rewind bufferfile}
 Reset(bufferfile);

 {first line of time step for ADJT}
 WriteLn(outputfile,setnumber:5,window.pos[window.act]:5);
 {localize time to search for}
 time := window.time[window.act];

 {scan buffer file for record with the specified time}
 WHILE NOT EOF(bufferfile) DO BEGIN
  Read(bufferfile,buffer);

  {wite to outputfile if times match}
  IF buffer.time=time THEN BEGIN
```

```
      WriteLn(outputfile,buffer.i:5,buffer.j:5,buffer.k:5,'    ',
           buffer.head:12,'    ',buffer.name);
   END;
  END;

  {write last line of time step}
  WriteLn(outputfile,'   0');

END; {genereatetimesteps}

{-------------------------------------------------------------------}

 PROCEDURE InitOutput(User_string:Str80);
{=========================================}
{Initialize I/O channel for user output}
BEGIN
 Assign(User,User_string);
 ReWrite(User);
END; {InitOutput}

{-------------------------------------------------------------------}

 PROCEDURE LoadTimeWindows(VAR window: WindowType);
{=================================================}
VAR ia                    : INTEGER;
   parampos, iskip,fskip      : INTEGER;
   iread                  : INTEGER;
   error_flag             : INTEGER;
   time_buffer            : REAL;
   windowfilename             : Str80;

BEGIN
 {initialize Window}
 FOR ia:=1 TO max_window DO BEGIN
   window.pos[ia]  := 0;
   window.time[ia] := 0.0;
   window.flag[ia] := 0;
 END;
 window.act  := 0;
 window.last := 0;

 {open windowfile}
 parampos:=3;
 IF ParamCount>=parampos THEN BEGIN
   windowfilename:=ParamStr(parampos);
```

```
 END ELSE BEGIN
   WriteLn(User,' +++ Error: Windowfile not specified ...');
   Help;
 END;
 Assign(windowfile,windowfilename);
 {$I-} Reset(windowfile); {$I+}
 IF IOResult<>0 THEN BEGIN
   WriteLn(User,' +++ Error: Unable to open Windowfile ...');
   Help;
 END;

 {get skipping factor}
 parampos:= parampos+1;
 fskip := 0;
 IF ParamCount>=parampos THEN BEGIN
  error_flag:=0;
  Val(ParamStr(parampos),fskip,error_flag);
  IF error_flag<>0 THEN BEGIN
    WriteLn(User,' +++ Error: specified skipping factor not a numeric value ...');
    Help;
  END;
 END;

 {load windows from windowfile}
 WriteLn(User,lfc,lm,'PPONS: Loading windowfile ....');

 iread := 0;
 iskip := fskip;  {use always first record}

 WHILE NOT EOF(windowfile) DO BEGIN
  {read next record}
  iread := iread+1;
  iskip := iskip+1;
  ReadLn(windowfile,time_buffer);

  {store record unless skipping specified}
  IF iskip>=fskip THEN BEGIN
    iskip := 0;
    window.last := window.last +1;
    IF window.last > max_window THEN BEGIN
      WriteLn(User,lfc);
      WriteLn(User,' +++ Error: More than ',max_window:5, 'time windows ...',lfc);
      Close(User);
      HALT;
    END;
```

```
    window.pos[window.last] := iread;
    window.time[window.last] := time_buffer;
   END;
 END; {while}

 {print stored time windows}
 WriteLn(User,lfc,lm,'  Set  Step      Stored Time');
 WriteLn(User,    lm,'-----------------------------');
              {   12345 12345   123456789012345 }

 FOR ia:=1 TO window.last DO BEGIN
   WriteLn(User,lm,ia:5,window.pos[ia]:6,'   ',window.time[ia]:15:2);
 END;

END; {loadtimewindows}


{-------------------------------------------------------------------}

 PROCEDURE LoadWellData(VAR well: WellType);
{=========================================}
VAR ia, parampos, iwell        : INTEGER;
   wellfilename               : Str80;

BEGIN

 {initialize well stack}
 FOR ia:=1 TO max_well DO BEGIN
   well.i[ia] := 0;
   well.j[ia] := 0;
   well.k[ia] := 0;
   well.name[ia] := '';
 END;
 well.act  := 0;
 well.last := 0;

 {open wellfile}
 parampos:=2;
 IF ParamCount>=parampos THEN BEGIN
    wellfilename:=ParamStr(parampos);
 END ELSE BEGIN
    WriteLn(User,lfc,lfc,' +++ Error: Wellfile not specified ...');
    Help;
 END;
 Assign(wellfile,wellfilename);
```

Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

```
 {$I-} Reset(wellfile); {$I+}
 IF IOResult<>0 THEN BEGIN
   WriteLn(User,lfc,lfc,' +++ Error: Unable to open Wellfile ...');
   Help;
 END;

 {load well data from wellfile}
 WriteLn(User,lfc,lfc,lm,'PPONS: Loading well data ....');

 iwell := 0;
 WriteLn(User,lfc,lm,' No.   Filename    Grid Block Indices');
 WriteLn(User,   lm,'----------------------------------------');
            {   12345  <1234567890.12> 12345 12345 12345}


 {read all records}
 WHILE NOT EOF(wellfile) DO BEGIN
  iwell := iwell+1;
  IF iwell > max_well THEN BEGIN
    WriteLn(User,lfc);
    WriteLn(User,' +++ Error: More than ',max_well:5, 'wells ...',lfc);
    Close(User);
    HALT;
  END;

  ReadLn(wellfile, well.name[iwell],well.i[iwell],well.j[iwell], well.k[iwell]);

  {convert letters to uppercase}
  FOR ia:=1 TO max_namelength DO BEGIN
    well.name[iwell][ia] := UpCase(well.name[iwell][ia]);
  END;

  {print well record}
  WriteLn(User,lm,iwell:4,'  <',well.name[iwell],'>',
             well.i[iwell]:5,well.j[iwell]:6, well.k[iwell]:6);
 END; {while}
 well.last := iwell;
END; {loadwelldata}

{-------------------------------------------------------------------}

 PROCEDURE OpenFile;
{==================}
VAR parampos       : INTEGER;
   outputfilename  : Str80;
```

Culebra Transmissivity Fields
Analysis Package
WBS:
SWCF-A:1.2.07.4.1:WA:QA:AP-018

```
BEGIN
 parampos:=1;
```