ESTEL-3D version 5.6
User Manual

JP Renaud
University of Bristol

December 4, 2006

Chapter 1

Introduction

ESTEL-3D is a numerical model for flow and contaminant transport in porous media. **ESTEL-3D** is developed by EDF, where it is used to look at the viability of a deep nuclear waste repository, in collaboration with the University of Bristol, where **ESTEL-3D** is used for subsurface hydrology research.

ESTEL-3D is part of the **TELEMAC** system. **TELEMAC** is a state-of-the-art hydroinformatics system comprising several modules for water flow, sediment transport, water quality and waves. In particular, the **TELEMAC** system contains the **BI**bliothèque **É**léments **F**inis (BIEF), a powerful finite element library shared by all modules. The **BIEF** library handles numerical operations. Like all other modules in the **TELEMAC** system, **ESTEL-3D** is written in FORTRAN 90.

In version v5p6, **ESTEL-3D** contains a prototype module solving the advection - dispersion equation for solute transport using the random walk method. This module is not feature conmplete and will not be described in this user manual. Please read chapter 11 if you wish to use this experimental feature.

ESTEL-3D solves the partially saturated Richards equation in its mixed form:

$$\frac{\partial \theta}{\partial t} = \underline{\nabla}.\underline{\underline{K}}.\underline{\nabla}\,(h+z)$$

θ is the moisture content, $\underline{\underline{K}}$ the conductivity tensor, h the pressure head, z the elevation, and t the time. Richards' equation is usually written with an additional source/sink S term but this term is not activated in version v5p6.

ESTEL-3D solves equation by the finite element method on an unstructured mesh of tetrahedra. It is a partial differential equation and requires both initial and boundary conditions.

Chapter 2

The steering file

An **ESTEL-3D** simulation is controlled via a steering file. The steering file is a text file which contains a list of keywords. All keywords available in **ESTEL-3D** are described in this chapter. The keywords have been grouped by function (input files, output files, numerical parameters, physical parameters etc...).

For each keyword, the syntax used in this manual is:

KEYWORD= example - type - default value Description of the keyword.

The keywords can be of **type** integer (**int**), real (**real**), boolean (**bool**) or character string (**char**), written inside quotes: **"char"**. The **default** value is the value which will be used by **ESTEL-3D** if the keyword is not found in the steering file. When **none** is written as the **default** value, it means that the keyword has to be included in the steering file. Note that in the steering file, the equal signs '**=**' could be subsitituted with colons '**:**'. Therefore you might find that some people use the syntax:

KEYWORD: value

Note that some keywords (e.g. see **SPACE AND TIME UNITS**) expect more than one value. In that case, the values should given separated by a semi-colon '**;**' and the type in the the description will contain the term **list**.

The following section describe the keywords used in a typical simulation. The keywords have been grouped by function (input files, output files, numerical parameters, physical parameters etc...)

2.1 Input files

This first set of keywords tells **ESTEL-3D** where to find information such as the geometry of the domain for example.

FORTRAN FILE= 'file.f90' - char - none This keyword is used to give the name of the Fortran file (extension .f90). The Fortran file is compiled when the simulation starts and contains a Fortran module. This module contains all subroutines that the user can modify for his simulation. An in-depth description of the Fortran file is given in chapter 3.

ICEM LOG FILE= 'file.log' - char - none This keyword is used to give the name of the log file (usual extension .log) for the mesh in UNiVersal format (see below). This file is created automatically by the ANSYS ICEM CFD software when generating a mesh. The log file contains summary information about the mesh such as number of nodes elements, etc... Although it is a text file, this file should *NOT* be edited manually.

UNIVERSAL FILE= 'file.unv' - char - none This keyword is used to give the name of the mesh in UNiVersal format (usual extension .unv). This file is usually created using the ANSYS ICEM CFD software. Although it is a text file, it should *NOT* be edited manually.

FILE CONTAINING THE SOIL DATABASE= 'sdb' - char - none This keyword is used to give the name of the file which contains the soil data base file which is described in more details in section 6. Note that the soil database can be included in the sterring file.

2.2 Output files and variables

This set of keywords is used to tell **ESTEL-3D** which variables to save in the results files and which file names to use.

3D RESULTS FILE = 'file.dat' - char - none This keyword is used to give the name of the results file. The 3D results file is written in either binary (usual extension .plt) or ASCII (usual extension .dat) Tecplot datafile format. See THE **FILE FORMAT FOR THE 3D OUTPUT FILES** keyword below for a description of the keyword to switch between binary and ASCII output. Section 10.1 contains a more in-depth description of the 3D results file.

VARIABLES FOR 3D GRAPHICAL PRINTOUTS = 'H,VX,VY,VZ' - char - 'H,THETA,IMAT' This keyword is used to list the variables to output in the 3D results file. The character string contains a list of variables separated by commas ','. The key is:

- Example: **NAME**="Full Name Of Variable" - (Unit) - Discretisation
- **H**="Pressure Head" - (L/T) - P1
- **THETA**="Volumetric Moisture Content" - (L3/L3) - P0
- **KXX**="Hydraulic Conductivity Kxx" - (L/T) - P0
- **KYY**="Hydraulic Conductivity Kyy" - L/T) - P0
- **KZZ**="Hydraulic Conductivity Kzz" - (L/T) - P0
- **KUNSAT**="Relative Unsaturated Hydraulic Conductivity" (-) - P0
- **SAT**="Effective Saturation" - (-) - P0
- **CAP**="Hydraulic Capacity" - (1/L) - P0
- **VX**="Darcian Velocity Vx" - (L/T) - P0

- **VY**="Darcian Velocity Vy" - (L/T) - P0
- **VZ**="Darcian Velocity Vz" - (L/T) - P0
- **ZG**="Elevation" - (L) - P1
- **HH**="Hydraulic Head" - (L) - P1
- **IMAT**="Soil Type" - (-) - P0

See section 10.1 for more information about the discretisation types. Note that the soil type **IMAT** variable, as well as the coordinate X, Y and Z are always output in the 3D results file. Also note that although this keyword lists the variables to output, it is not a "keyword list"; the syntax is that of single character string where variables are separated by commas rather than a list of character strings.

FILE FORMAT FOR THE 3D OUTPUT FILES= 2 - int - 1 This keyword specifies whether the 3D results file and the mesh results file will be written in ASCII or binary mode. The choices are:

- **1**="TECPLOT ASCII"
- **2**="TECPLOT BINARY "

In the official/commercial version of **ESTEL-3D**, the default value is ASCII output, value 1. This is due to concerns with the redistribution of the Tecplot library. It is very likely that this default value has been changed to binary on your platform. Check with your distributor. Note that binary output files take a lot less file space than ASCII files. However, they can only be read with the Tecplot software version 10 and above. An ASCII output file can be read with any file editor.

TITLE= 'Title of the simulation' - char - No title in the steering file The title is a simple character string which will be written on the screen when **ESTEL-3D** runs and will be written in the results file.

STOP AFTER CREATING THE MESH RESULTS FILE = yes - bool - no Sometimes, it is difficult to parameteris an **ESTEL-3D** simulation as the geometry is so complex that the external faces are difficult to visualize. It is now possible to ask ESTEL-3D to interpret the mesh save the results in the mesh results file (see below) and then stop the simulation. This allow a direct visualisation of complex meshes as interpreted by **ESTEL-3D**.

3D MESH RESULTS FILE = 'file.dat' - char - none This keyword is used to give the name of a file which will contain a version of the mesh after it has been interpreted by **ESTEL-3D** (see above). A more detailed description of this file is given in section 10.2.

2.3 Simulation duration and time control

This section lists the time-related keywords for a **ESTEL-3D** simulation.

SPACE AND TIME UNITS = 'm';'s' - char - 'm' ; 's' This keyword is used to define respectively the length and time units. Each unit is a character string of four characters at most. Note that the units defined in the steering file *have to* match the units defined in the soil database file (see section 6) so that the conductivity value is matched to the mesh dimensions. If they do not match, the simulation will stop.

INITIAL TIME = 0. - real - 0. Initial time for the simulation, the initial conditions will be for the time specified with this keyword.

FINAL TIME = 5. - real - 1. Final simulation time. The simulation will stop when this time is reached.

TIME STEP = 1. - real - 1. The time step is time increment used by **ESTEL-3D** to move forward in time. For instance, if the time usint is the second, initial time is 0s the final time 5s and the time step 1s, then **ESTEL-3D** will use initial conditions at time 0s and compute a solution of Richards equation for the timess 1s, 2s, 3s, 4s and 5s.

PRINTOUT PERIOD = 2. - real - 1. Simulation time period between output in the 3D results file.

INITIAL PRINTOUT TIME = 0. - real - 0. Simulation time of the first printout.

STEADY STATE HYDROLOGY = no - bool - no This keyword specifies whether ESTEL-3D is to be run in steady state or transient mode. In steady state mode, Richards equation simplifies a lot as the time derivative disappears.

CONSTANT INITIAL CONDITIONS = 1. - real - 0. When running in **ESTEL-3D** in transient mode, Richards equation contains time derivatives. Therefore it requires initial conditions. This keyword can be used to impose a constant value to all nodes in the mesh as initial conditions. This is often not sufficient and more complex initial conditions can be programmed in the Fortran file. See chapter 7 for more detail.

Note that to facilitate the interpretation of the results, it is important that the **PRINTOUT PERIOD** be a multiple of the time step and that **INITIAL PRINTOUT TIME** is of the form **INITIAL TIME** + N * **TIME STEP** where N is any integer.

2.4 Domain geometry

NUMBER OF EXTERNAL FACES = N - int - none This keyword defines the number of external faces which constitute the boundary of the computational domain. For instance for a cube N would be equal to six. This number is required to fully parameterise an **ESTEL-3D** simulation so when using a complex geometry, it is useful to write down the number of external faces created by ANSYS ICEM.

PRIORITY FOR EXTERNAL FACES = 5 ; 6 ; 2 ; 3 ; 4 ; 1 - list int - none When a node belongs to two or more external faces, it needs to be assigned a specific face number that will be re-used when setting up the boundary conditions (see chapter 7). The approach used in ESTEL-3D consists of defining a priority list for the external faces and use this list to determine which face number should be used. For instance, if a node belongs to faces 1 and 3, the example priority list above will assign the face number 3 to that node. Note that the mesh results file can help you finding how the external face numbers are distributed on the mesh. This is described in section 10.2. This section describes keywords which describes the physics of the flow to be modelled.

2.5 Extra physical parameters

INDEX OF THE UNIFORM SOIL = N - int - 1 This keyword defines which soil of the soil data base (by default soil 1) is to be used by default for all elements of the mesh. More information about soils is given in chapter 6 for the soil database and chapter 5 for the soil distribution.

3D DIRECTION OF GRAVITY = 0. ; 0. ; -1. - list real - 0. ; 0. ; -1. This keywords expects a list of three reals which define the components of a vector gravity $\underline{g}$ on the axis x, y and z. The modulus of the vector does not matter and only its direction is used, i.e. (0.;0.;-1) and (0.;0.;-45.) produce the same result. However, using (0; 0., +12.) would define an upward gravity vector.

INFLUENCE OF GRAVITY = yes - bool - yes The influence of gravity can be switched on or off using this keyword. Note that switching the gravity off has no physical meaning in a 3D code and this option is usually used only to compare the results with the analoguous heat transfer equation for which many analytical solutions exist.

2.6 Numerical parameters

This final set of keywords is used to control how the equation is actually solved. The values of these keywords can be modified to match particular requirements. However, the values used in this document (usually the default values) are a good starting point for any simulation.

2.6.1 Iterative scheme

As Richards' equation is highly non-linear in unsaturated conditions, **ESTEL-3D** buses an iterative scheme to solve it. This section describes the keywords related to the iterative scheme.

SCHEME FOR SOLVING THE RICHARDS EQUATION = 1 - int - 2 Two schemes are available, the classic Picard scheme and the modified Picard scheme. The modified scheme is meant to be more performant (more stable and requires less iterations) but users have had problems with it, hence the classic Picard scheme is still the default scheme.

- **1**="MODIFIED PICARD SCHEME"

- **2**="CLASSIC PICARD SCHEME"

CONVERGENCE CRITERION OF THE ITERATIVE SCHEME = 2 - int - 2
Different types of convergence criteria can be used to stop the scheme iterating. The dictionnary cites three choices:

- **0**="ALWAYS CONVERGENCE"
- **1**="RELATIVE CONVERGENCE CRITERION"
- **2**="ABSOLUTE CONVERGENCE CRITERION"

However, in ESTEL-3D v5p6, **only the absolute convergence criterion is available**. Do **NOT** use option **0** or **1**, they are only provided for debugging purposes at this stage.

ACCURACY OF THE ITERATIVE SCHEME FOR THE RICHARDS EQUATION = 1.E-4 - real - 1.E-4 This keyword sets a value for the convergence criterion defined above. Assuming you are using the absolute convergence criterion (and you should!) the accuracy is expressed in the length unit, i.e. **m** by default. When the offset between two successive iterations of the iterative scheme is less than the required accuracy, the scheme stops.

MAXIMUM NUMBER OF ITERATIONS FOR THE ITERATIVE SCHEME = 30 - int - 40 This keywords sets an upper limit to the number of iterations allowed before convergence of the iterative scheme. Note that if this maximum number of iterations is reached, the simulation stops.

IMPLICITATION FOR THE PRESSURE HEAD = 0.85 - real - 0.55 A semi-implicit time formulation is used throughout **ESTEL-3D**. This keyword is used to set the level of implicitation. With a value of 1. , the scheme is fully implicit. With a value of 0., it is fully fully explicit. A value slightly higher than 0.5 is usually best. Note that if you experience problems, you can safely increase its value to get closer to 1. It usually helps.

2.6.2 Numerical solver

ESTEL-3D uses solvers from the **BIEF** library to handle the numerics. All methods offered by **BIEF** are offered in **ESTEL-3D**. Read the **BIEF** documentation for a detailed description of the relevant options. This section merely lists the relevant keywords.

SOLVER FOR THE RICHARDS EQUATION = 1 This keyword is used to choose which solver is used to solve Richards equation. The list of available solvers is available from the BIEF documentation.

ACCURACY OF THE SOLVER FOR THE RICHARDS EQUATION = 1.E-12 This keyword is used to fix the accuracy of the solver for Richards equation.

MAXIMUM NUMBER OF ITERATIONS FOR THE SOLVER FOR THE RICHARDS EQUATION = N The solvers are iterative and this keyword can be used to stop the iterative process after a N iteration, even if the accuracy has not been reached. N = 200 is usually a good guess.

Chapter 3

The Fortran file

3.1 Background

All numerical models in the **TELEMAC** system have the option of having part of the program recompiled at the beginning of a simulation. This is a very powerful feature as it means that the user is not limited to the situation thought of by the developer and can use its own code. The subroutines that can be recompiled at runtime are called the user subroutines. In **ESTEL-3D**, these subroutines are all put inside one single module called **M_ESTEL3D_USERPROCS**. This chapter lists all available subroutines and each subroutine is described in detail in the following chapters.

To make use of the user subroutines, the file M_ESTEL3D_USERPROCS.f90 needs to be copied to the simulation folder. It is then usually renamed something like fortran.f90. And the new name given to the **FORTRAN FILE** keyword in the steering file. Now, the whole module will be recompiled at the beginning of a simulation. Each subroutine inside this module is blank and the sections that can be edited are marked as follows:

```
!-------------------------------------------------------------------------
!********************** START USER CODE *********************************
!-------------------------------------------------------------------------
! INSERT YOUR CODE HERE!
!-------------------------------------------------------------------------
!************************ END USER CODE *********************************
!-------------------------------------------------------------------------
```

3.2 List of user subroutines

user_correct_xyz is used to modify the mesh. This can be useful to manipulate the coordinates created by the mesh generator, for instance to change scale by converting metres to kilometres etc... This subroutine is described in more details in section 4.

user_correct_soils is used to distribute the soil types on the mesh. This subroutine is described in more details in section 5. The soil types themselves are described in chapter 6

user_boundary_conditions is used to define the boundary conditions. This subroutine is described in more details in chapter 7.

user_initial_conditions is used to define the initial conditions. This subroutine is described in more details in chapter 8.

3.3 Particle tracking subroutines

Note that in the module **M_ESTEL3D_USERPROCS**, two extra subroutines are listed: **user_new_particles** and **user_dispersion**. These subroutines are used by the particle tracking module and not described in this manual. If you want to use them, please refer to section 11.2.

3.4 Required Fortran knowledge

Basic knowledge of the Fortran language is required to use ESTEL-3D.You should be comfortable with `if` statements and array structures. Extra knowledge of Fortran would help you setting up more complex cases.

Chapter 4

Definition of the mesh

Generating the mesh is probably the most complicated step of setting up an **ESTEL-3D** simulation. Because unstructured meshes of tetrahedra are very complex, it is often not convenient to use Fortran subroutines to modify spatially distributed input and these operations have to be done during the mesh generation phase. Therefore, **the user needs to have a good idea of the soil distribution and the boundary conditions when building the mesh with the software ANSYS ICEM CFD** so that he can define a specific "colour" to the soil types and boundary conditions.

ESTEL-3D requires two files from **ANSYS ICEM CFD**: the universal file which contains the mesh definition and the log file which contains a summary of the information contained in the universal file, such as number of points, number of elements etc...

The mesh created with **ANSYS ICEM CFD** can be slightly modified at the beginning of an **ESTEL-3D** simulation. This can be used to convert the length unit or perform simple manipulations such as moving the origin of the coordinate system. These modifications can be done in the subroutine **user_correct_xyz**.

The user can modify the arrays `x`, `x` and `x` for each of the `npoin` points in the mesh.:

```
  integer,          intent(in)    :: npoin    ! Number of points

  double precision, intent(inout) :: x(npoin) ! X coordinates
  double precision, intent(inout) :: y(npoin) ! Y coordinates
  double precision, intent(inout) :: z(npoin) ! Z coordinates
```

For instance, to scale the mesh and change the origin of the coordinate system (translation), one could use:

```
integer :: ipoin

do ipoin=1,npoin

          ! Scale the mesh (from m to km for instance)
```

```
  x(ipoin) = x(ipoin) / 1000.
  y(ipoin) = y(ipoin) / 1000.
  z(ipoin) = z(ipoin) / 1000.

  ! Move the origin 50 m up (50 m = 0.05 km)
  z(ipoin) = z(ipoin) - 0.05

enddo
```

Note that **user_correct_xyz cannot** be used to modify the mesh connectivity table, i.e. the nodes can be moved around the but the segments still link the same nodes. Obviously, if the user code entered in **user_correct_xyz** creates intersections between segment or faces, the simulation will stop as the mesh is not valid anymore.

Chapter 5

Distribution of the soil types on the mesh

The distribution of the soil types on the grid is done via the user subroutine **user_correct_soils**. The soil type for each element of the mesh is contained in the Fortran array `imat`:

```
    integer, intent(in)    :: nelem       ! Number of mesh elements
    ! Final mesh distribution
    integer, intent(inout) :: imat(nelem)
```

The user has to give an integer value to each element of the mesh. This integer value corresponds to a soil type of the soil database (see chapter 6).

To achieve this, the user should capitalize as much as possible on the work done during the mesh generation and use the element colours defined in **ANSYS ICEM CFD**. These colours are automatically read **by ESTEL-3D** and are made accessible to the user via the Fortran array `imat`:

```
  ! Element colours in the UNV file
  integer, intent(in)      :: nsol(nelem)
```

Then, `imat` can be from `nsol` using simple Fortran statement, for instance:

```
      integer :: ielem

      do ielem = 1,nelem
        imat(ielem) = nsol(ielem)
      enddo
```

Obviously, the ICEM colours can be overidden if necessessary, for instance, to use soil number 1 for the colours 1 and 2 and soil number 2 for the colour 3, one could use:

```
      integer :: ielem

      do ielem = 1,nelem
        if (nsol(ielem) .eq. 1) imat(ielem) = 1
```

```
    if (nsol(ielem) .eq. 2) imat(ielem) = 1
    if (nsol(ielem) .eq. 3) imat(ielem) = 2
  enddo
```

More complex operations can be done on the array `imat`, using the mesh coordinate for example, but as manipulating unstructured 3D data is complex, we recommend using the **ANSYS ICEM CFD** colours as much as possible.

Note that if one soil type only is required in the simulation, ESTEL-3D will use the soil number 1 from the soil database by default. The index of the default soil can be changed with the keyword **INDEX OF THE UNIFORM SOIL**. In any case, the index of the unifor soil is given to all elements *before* calling **user_correct_soils**. Therefore the keyword can be used to give a default value to all elements and only those who require modification are modified in **user_correct_soils**.

Chapter 6

Definition of soil properties

6.1 Syntax

The soil properties are kept in the soil database file which is a text file whose name is specified in the steering file via the keyword **FILE CONTAINING THE SOIL DATABASE**. The syntax of the file is similar to that of the steering file, i.e. it consists of a list of keywords. The soil database can be in a standalone file or whitin the steering file itself, at the user's convenience.

Below is an extract the file containg the soil database from the "tracy3d" test case:

```
SPACE AND TIME UNITS = 'm';'s'

NUMBER OF SOILS = 2

/                               Soil 1   Soil 2 (dummy)
/                              (Tracy)   (VG)

 SOIL MOISTURE MODEL          =  12     ; 3

 SATURATED MOISTURE CONTENT   =   0.45  ; 0.75
 RESIDUAL MOISTURE CONTENT    =   0.15  ; 0.22
 SATURATED CONDUCTIVITY KS111 =  10.    ; 1.e-12
 SATURATED CONDUCTIVITY KS222 =  10.    ; 1.e-6
 SATURATED CONDUCTIVITY KS333 =  10.    ; 1.e-09

 SOIL MODEL PARAMETER SP1     =  10.    ; 10.
 SOIL MODEL PARAMETER SP2     =   0.    ; 1.e-8
 SOIL MODEL PARAMETER SP3     =   0.    ; 1.
 SOIL MODEL PARAMETER SP4     =   0.    ; 100.
 SOIL MODEL PARAMETER SP5     =   0.    ; 100000.
```

The **SPACE AND TIME UNITS** contains again the space and time units for the soil database. These units **have to** be identical to those specified in the steering file or the simulation will stop. This check has been implemented to avoid typical end-user mistakes where the soil database is written for units different to those of the mesh or the time step written in the steering file. Note that if all the soil database keywords are put directly in the steering file, there is no need to repeat the **SPACE AND TIME UNITS** twice.

The **NUMBER OF SOILS** is then required. It will be used to dimension all other keywords which expect lists of values according to the following pattern:

```
NUMBER OF SOILS = N
/           Soil 1   Soil 2   Soil 3   ... , Soil N
 PROPERTY = Value1 ; Value2 ; Value3; ... ; ValueN
```

The **SOIL MOISTURE MODEL** relates to the type of moisture curve which will be used for the relationship between moisture content, relative conductivity and pressure head. A detailed description of each model is given in section 6.2. In version v5p6, only three soil models are available:

- **1**="Constant Storage"
- **3**="Van Genuchten"
- **10**="Tracy"

Moreover, the "Tracy" model should not be used for most simulations, see section 6.2.

The significance of the **SATURATED MOISTURE CONTENT** and **RESIDUAL MOISTURE CONTENT** keywords is self explanatory.

The saturated conductivity tensor used by **ESTEL-3D** has three components **SATURATED CONDUCTIVITY KSxxx**. This allows for anisotropy but the tensor has to be *diagonal*. Therefore the general form of the saturated conductivity tensor can be written

$$\underline{\underline{K_s}} = \begin{pmatrix} K_{s_{111}} & 0 & 0 \\ 0 & K_{s_{222}} & 0 \\ 0 & 0 & K_{s_{333}} \end{pmatrix}$$

Therefore the main axis of anisotropy have to be parallel to the coordinate axis. This is a limitation when using ESTEL-3D in stratified media where the stratification is diagonal.

The following five **SOIL MODEL PARAMETER SPx** are used to define the soil moisture curve. See the section 6.2 for their significance. Parameters not used in the soil moisture curve relationships are simply ignored by **ESTEL-3D**.

6.2 Soil models

6.2.1 Constant storage

The constant storage model is usually used in saturated conditions. A small value is given to the slope of the moisture cuurve, allowing for transient flow in saturated conditions. The se slope is related to the compressibility of both the water and the porous matrix. The mathematical formulation is:

$$\frac{\partial\theta}{\partial h} = S_s$$

In the equation above, the specific storage S_s takes its value from the parameter **SP1** from the soil database. The parameters **SP2** to **SP5** are ignored when using the constant storage model.

6.2.2 Van Genuchten

The Van Genuchten model is a very popular soil model for unsaturated flow. Its mathematical formulation is:

$$\frac{\theta - \theta_r}{\theta_s - \theta_r} = \frac{1}{(1 + (\alpha h)^n)^{1-\frac{1}{n}}}$$

In the equation above, α and n take their value from the parameters **SP1** and **SP2**, n is dimensionless and α is in the inverse of a distance (L^{-1}). The parameters **SP3** to **SP5** are ignored when using the constant storage model.

6.2.3 Tracy model

A third model, the Tracy model is also available in ESTEL-3D. However, thios model has only been implemented to validate the results of the model with an analytical solution derived by Tracy. The Tracy soil model does not represent a natural soil and should **not** be used for simulations other than the Tracy test case.

Chapter 7

Boundary conditions

7.1 Background

Because Richards equation contains spatial derivatives, boundary conditions are required to close the problem and solve the equation. A boundary condition is some information about the unknown of the problem at the boundary of the computational domain. In **ESTEL-3D**, the unknown is the pressure head h. Boundary condition can consist of a value for h at the boundary or a value for the gradient of h. Two types of boundary conditions are available in **ESTEL-3D**:

Dirichlet The Dirichlet boundary condition represents an imposed head, i.e. the value of the pressure head h is know at the boundary.

Neumann The Neumann boundary condition is an imposed flux. It is used to specify how much water enters the domain though a section of the boundary. This is a condition on the gradient of h. Note that an impermeable or zero flux boundary is actually a Neumann condition with a flux value of 0.

These boundary conditions need to be specified for each boundary point. Again, because unstructured 3D data is complex to manipulate, it is advantageaous to use the data contained in the universal file created by ANSYS ICEM CFD as much as possible, in a similar way to what can be done for the soil types in section 6.

7.2 Defining the boundary conditions

The boundary conditions are defined with the user subroutine **user_boundary_conditions**. Three Fortran arrays are used to hold the boundary conditions in **ESTEL-3D**:

```
integer,          intent(inout)   :: lihbor(nptfr)
double precision, intent(inout)   :: hbor(nptfr)
double precision, intent(inout)   :: qbor(nptfr)
```

`nptfr` is the number of boundary points and `lihbor` is used for the type of boundary condition. To select a Neumann boundary condition for the boundary point `iptfr`, just use

```
lihbor(iptfr) = kneu
```

where `kneu` stands for Neumann. Similarly, `kdir` is used for Dirichlet points.

Dirichlet points require an imposed pressure head, this is done with the array `hbor`. Neumann points require an imposed flux, this is done with the array `qbor`. So to define an imposed flux of 1.e-6 m/s at the boundary point `iptfr` one could use:

```
lihbor(iptfr) = kneu
qbor(iptfr)   = 1.e-6
```

This assumes that the **TIME AND SPACE UNITS** are 'm' and 's' in the steering file.

Because locating boundary points on the 3D mesh is complicated, it is advantageous to use the faces defined during the mesh generation as much as possible. This can be done by using the array `FACE_FOR_BNODE` which contains a face number for each boundary node. When a boundary node belongs to several faces, the face with the highest priority is used. The priorities for the external faces is defined by the keyword **PRIORITY FOR EXTERNAL FACES**, see section 2.4. This allows the user to impose face-constant boundary conditions very quickly. For instance, for a cube, one could use the following code to impose an impermeable boundary on 4 faces and an imposed head on two faces:

```
! Variable for the loops
integer :: iptfr

! Put the whole boundary to no flux
! (LIHBOR = Neumann and QBOR = 0.)

do iptfr = 1,nptfr
   lihbor(iptfr) = kneu
   qbor(iptfr) = 0.d0
enddo

! Set faces 5 and 6 to imposed head
! LIHBOR = Dirichlet and HBOR = imposed values

do iptfr = 1,nptfr
   if (    (face_for_bnode(iptfr).eq.5) ) then
      lihbor(iptfr) = kdir
      hbor(iptfr)   = imposed_head_face_5

   else if ( (face_for_bnode(iptfr).eq.6)  ) then
      lihbor(iptfr) = kdir
      hbor(iptfr)   = imposed_head_face_6
   endif
 enddo
```

This example shows that if particular care is given to the boundary faces numbering during the mesh generation, the information can be used very easily when parameterising **ESTEL-3D**. More complex boundary conditions could be defined using for instance the spatial coordinates to select a set of nodes.

Chapter 8

Initial conditions

Because Richards equation contains time derivatives in transient mode, initial conditions are required to close the problem and solve the equation. Initial conditions consist of some information about the unknown of the problem at the start of the simulation. In **ESTEL-3D**, the unknown is the pressure head h and the initial conditions are values of the pressure head at each node in the mesh at the start of the simulation.

For simple cases, the keyword **CONSTANT INITIAL CONDITIONS**. Its value will be given to all nodes in the mesh. In most cases, it is required to use the user subroutine **user_initial_conditions**. The array to use is called `hn`:

```
integer,          intent(in)    :: npoin     ! Number of mesh nodes
double precision, intent(inout) :: hn(npoin) ! Initial head distribution
```

This subroutine has to be modified so that each node in the mesh has the relevant value of pressure head. Note that the space coordiantes are available in this subroutine to select sets of nodes and set their `hn` value accordingly.

Note that if you run ESTEL-3D in steady state mode, initial conditions are not theoretically required but a well chosen initial condition could speed up solver convergence.

Chapter 9

Running ESTEL-3D

Running ESTEL-3D is done at the command line in a similar way to all other modules of the TELEMAC system. Just call '`estel3d`' with the name of the steering file. Note that if no steering file name is given, ESTEL-3D uses by default a steering file called '`cas`':

```
estel3d name_of_steering_file
```

ESTEL-3D will then run in the shell window where it has been started and print running messages to the screen.

Chapter 10

A bit more about the output files...

10.1 ESTEL-3D results file

ESTEL-3D writes its results in a file named by the keyword **3D RESULTS FILE**. It is written in **Tecplot 10** format (either ASCII or binary) with one zone for each time output. Refer to the **Tecplot** documentation for information on how to use the software to explore the **ESTEL-3D** results and a description of the file format. However, here are a couple of things that can make your life easier:

Display the time If you use **Tecplot 360** or above, load the 3D results file use the "File"-¿"Load Data File" menu (not at the command line), then Tecplot becomes time aware. You can then anuimate the simulation by using the time box in the left hand panel. To display the time, open a text box and input the following text: `TIME=&(SOLUTIONTIME}` and **Tecplot 360** will show the current time. This trick is a lot more complicated with **Tecplot 10**.

Variable discretization In section 2.2, the notion of output variable discretisation was introduced. A P0 variable has one value per element (tetrahedron). A P1 variable has one value per node. By default, Tecplot show each contour map with values interpolated on the nodes. This is sometimes inconvenient for the P0 variables, for instance when visualising soil types. You can get around this by selecting the option "Primary Value Flood" as the "Countour Type". Tecplot will then display a "mosaic" of solid tetrahedra/triangles.

10.2 Mesh results file

The **MESH RESULTS FILE** is created at the beginning of a simulation and contains the mesh as interpreted by **ESTEL-3D**. It can be used as a debugging tool when dealing with complex geometries. The mesh results file contains two zone: the "Domain mesh" and the "Border mesh". On both meshes, two variables are available:

ID_NODE contains the node colour as interpreted by **ESTEL-3D**. So for instance visualising =fID_NODE on the surface mesh will display explicitely how the external face priorities have been interpreted by **ESTEL-3D**. This is in fact a visualisation of the array `face_for_bnode` used in **user_boundary_conditions**.

ID_ELEM contains the element colour as interpreted by **ESTEL-3D**. So for instance visualising **ID_ELEM** on the domain mesh can help distributing the soil types in the domain as you actually visualise the array `nsol` used in **user_correct_soils**.

Chapter 11

More information and Feedback

11.1 This document is out-of-date...

Most things move on... Printed documentation does not.

The latest version of this document and other related documentation is available in PDF format at the URI:
http://source.ggy.bris.ac.uk/wiki/Estel_Documentation.

A working copy in LaTeX format is available from the Subversion repository at the University of Bristol at the URI:
http://source.ggy.bris.ac.uk/subversion/estel-doc/branches/REL-v5p6/

Note that you will need a username and a password to download this document in editable form. If you don't have one, ask for one. See Contacting people (11.4) below.

11.2 The particle tracking module

A prototype of the particle tracking module is present in **ESTEL-3D** v5p6. However, it is still evolving a lot and we do not recommend using it yet. If you really want to use it, look at the text case `cube\_part}`in the `test.gb/}`directory. You should really be using v5p7 or later though...

The relevant user subroutines are **user_new_particles** and **user_dispersion**.

user_new_particles is used add new particle on a per elment basis.

user_dispersion is used to define the values of the dispersion tensor.

The test case `cube\_part` shows how these two subroutines are used.

11.3 Bugs...

By the time you read this user manual, **ESTEL-3D** v5p7 or later will be in heavy development. If you encounter a bug or think a feature does not work as advertised, it is likely that this has been addressed in the next version of **ESTEL-3D**. However, you are highly encouraged to get in touch with the developers so that your feedback helps them make **ESTEL-3D** (even) better for the next release...

11.4 Contacting people

If you need to report a bug, have a question about some particular feature, please use the following contacts:

JP Renaud, University of Bristol `j.p.renaud@bristol.ac.uk`
Main developer at the University of Bristol. Contact Jean-Philippe with any general question about **ESTEL-3D**, any bug report or feature request.

Regina Nebauer, EDF `regina.nebauer@edf.fr`
Main developer and in charge of **ESTEL-3D** at EDF. Contact Regina for any general question, any bug report or feature request and also purchasing/licensing information.