Implementation and testing of the CODESA-3D model for density-dependent flow and transport problems in porous media

Giuditta Lecca
Environment Area

July 2000

Centre for Advanced Studies, Research and Development in Sardinia
VI Strada Ovest, Z. I. Macchiareddu, C.P. 94, I-09010 Uta, CA, Italia

Contents

1 Introduction

The report describes the implementation and test phases of the computer code CODESA-3D (COupled variable DEnsity and SAturation 3-Dimensional model), whose mathematical and numerical models are described in all details in the tenth chapter of the book [15].

CODESA-3D is a three-dimensional finite element simulator for flow and solute transport in variably saturated porous media on unstructured domains. The flow and solute transport processes are coupled through the variable density of the filtrating mixture made of water and dissolved matter (salt, pollutants). The flow module simulates the water movement in the porous medium, taking into account different forcing inputs: infiltration/evaporation, recharge/discharge, withdrawal/injection, etc., while the transport module computes the migration of the salty plume due to advection and diffusion processes.

Typical applications of the model are so-called *density-dependent* problems in subsurface hydrology; in particular the model has been applied to the saltwater intrusion problem of coastal aquifesr [24, 27, 22] and brine movement in a radionuclide polluted aquifer [29]. Denser-than-water non-aqueous phase liquids (DNAPLs), such as chlorinated organic contaminants, are other examples of density-dependent contaminants, which can be modeled with CODESA-3D.

The CODESA-3D code is born from the integration and extension of two *parent* codes:

- SATC3D: SATurated Coupled flow and transport 3-Dimensional model;
- FLOW3D: variably saturated FLOW 3-Dimensional model.

These two computer codes were developed, during the last ten years, by the Department of Applied Mathematics of the University of Padova in collaboration with the Environment Group of CRS4. Code manuals (in Italian) and related publications are [11, 13, 14] and [10, 12], respectively.

The report represents the first version of the CODESA-3D *developer and user's manual* describing:

- the **mathematical and numerical models**. Only a brief summary of them is included here for cross-referencing reason; refer to [15] for the complete description of CODESA-3D model;
- the **implementation model**. It is largely based on the parent SATC3D and FLOW3D routines, written in Fortran 77, but in addition allows a more flexible data management and enhances software modularity via Fortran 90 features such as user-derived data types and dynamic memory allocation. The code listings of the most representative routines are also included for a rapid but comprehensive instruction/data flow checking;
- the **application model**. It is described the set up and the discussion of the results of *case studies* developed for didactic, test and application purposes.

Appendices A, B and C report respectively:

- the expressions of the physical parameters of the mathematical model and those of matrices and vectors of the numerical model. In this way, the CODESA-3D extensions to the parent codes (SATC3D and FLOW3D) and the planned extensions to build up the final version of model [15] are summarized. The input and output of the model are discussed, as well;
- the description and usage of all the code variables (listed in the `codesa3d_header.h` file) and the data structure hierarchy.
- the complete input/output dataset for a representative case study [16] and the CPU timings of the run over CRS4 available machines running Fortran 90 compilers.

2 Brief description of the model

In this section we briefly describe the mathematical and numerical model of CODESA-3D, described in details in [15]. The coupled flow and transport model is developed for the case of *variably saturated* porous medium, applicable both to the unsaturated (soil) and the saturated (groundwater) zone, and of *variably dense* filtrating solution, assuming mixing [1], between freshwater and dissolved salts.

2.1 Mathematical model

The CODESA-3D mathematical model is based on two coupled equations assessing, for a fixed control volume immersed in the flow domain (Figure 1), the *mass conservation principle* [2] both for water (equation 1) and dissolved salt (equation 2). The first mass balance equation is referred further as *flow equation* and the second one as *transport equation* [3].

In what follows $[x, y, z]^T$ is the Cartesian spatial coordinate vector [4], with z vertical coordinate directed upward, and t is the time.

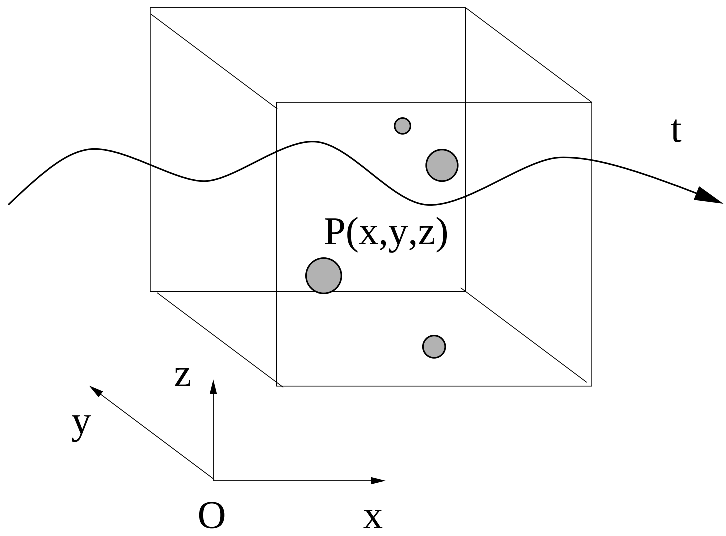

Figure 1: *Control volume centered at point $P(x, y, z)$ of domain Ω and crossed by the water flow at time t. The salt dissolved in water is represented by gray particles.*

[1]As opposed to *sharp-interface* models, which consider an abrupt interface between salt- and freshwater, approximately assumed as immiscible water bodies.

[2]The excess of mass inflow over outflow across the control volume boundaries is equal to the increase of mass of the control volume per unit volume of the porous medium and per unit time.

[3]Also known as the *advection-dispersion* equation.

[4]The superscript T denotes the transpose operator.

The mathematical model is formulated in terms of **two unknowns**:

$$\begin{aligned} \psi(x,y,z,t) &= \frac{p}{\rho_o g} && \textit{equivalent freshwater pressure head} \\ c(x,y,z,t) &= \frac{\tilde{c}}{\tilde{c}_s} && \textit{normalized concentration} \text{ of salt} \end{aligned} \qquad (2)$$

The *equivalent freshwater pressure head* is defined as $\psi = p/(\rho_o g)$ $[L]$, where p $[ML^{-1}T^{-2}]$ is the water pressure, ρ_o $[ML^{-3}]$ is the freshwater density and g $[LT^{-2}]$ is the gravitational constant. A derived variable is the *equivalent freshwater hydraulic head* $h = \psi + z$ $[L]$. The *normalized concentration* [5] c $[/]$ is an adimensional variable $(0 \leq c \leq 1)$ defined as the ratio between actual $\tilde{c}$ and maximum $\tilde{c}_s$ absolute concentrations of salt in the (water) solution. The maximum absolute concentration $\tilde{c}_s$ is a characteristic parameter of the numerical application at hand: for saltwater intrusion problems $\tilde{c}_s$ is usually in the range $25 \div 35 \times 10^{-3}$ grams per liter (g/l) [6], which corresponds to an average salt concentration of seawater.

In this model the *variable density* ρ of the solution is expressed as a linear function of the normalized salt concentration c:

$$\rho = \rho_o(1 + \epsilon c)$$

where $\epsilon = (\rho_s - \rho_o)/\rho_o$, is the *density difference ratio*, typically $\epsilon \ll 1$, with ρ_s the solution density at the maximum concentration (c =1): $\rho_s = \rho_o(1+\epsilon)$. Also the dynamic viscosity μ $[ML^{-1}T^{-1}]$ of the solution is assumed linearly dependent on c: $\mu = \mu_o(1+\epsilon' c)$, with $\epsilon' = (\mu_s - \mu_o)/\mu_o$ the *viscosity difference ratio*, and μ_o and μ_s solution viscosities at $c = 0$ and $c = 1$, respectively.

The *real pressure head* in the variably dense water is $\psi^r = p/(\rho g) = \psi/(1+\epsilon c) \leq \psi$. Analogously the *real total head* is $h^r = z + \psi^r \leq h$.

With these definitions, the **coupled system of variably saturated flow and miscible salt transport** equations is:

$$\begin{aligned} \sigma \frac{\partial \psi}{\partial t} &= -\nabla \cdot \mathbf{v} - \phi S_w \epsilon \frac{\partial c}{\partial t} + \frac{\rho}{\rho_o} q \\ \phi \frac{\partial (S_w c)}{\partial t} &= -\nabla \cdot (c\mathbf{v}) + \nabla \cdot (D \nabla c) + q c^* + f \end{aligned} \qquad (\textit{mathematical model}) \qquad (4)$$

All terms in equations (4) are time inverses $[T^{-1}]$ [7] and the symbol ∇ is the gradient operator, e.g.: $\nabla z = [0,0,1]^T$ is the unit vertical vector.

In the first equation of system (4) the term $-\nabla \cdot \mathbf{v}$ express the divergence of water flux in the control volume; in the second equation the term $-\nabla \cdot (c\mathbf{v})$ expresses the *advective* flux, i.e. the flux carried by the water at its average velocity, while the term $\nabla \cdot (D \nabla c)$ expresses the

[5] Also called *relative concentration*.

[6] About 23÷35 grams per cubic meter (g/m^3).

[7] The continuity equations are formulated per unit volume of porous medium.

dispersive flux, linearly proportional to the gradient of concentration, which takes place from high concentration to low ones. *Dispersive* flux is the macroscopic effect resulting from local velocity fluctuations, accounting both for mechanical dispersion and molecular diffusion [3].

In the *flow equation* (first equation):

- $\mathbf{v}$ is the *Darcy velocity vector* $[L/T]$:

$$\boxed{\mathbf{v} = -\mathbf{K} \cdot [\nabla\psi + (1+\epsilon c)\nabla z]} \tag{5}$$

  with $\mathbf{K} = k_{rw}\mathbf{K}'_s$ the *variably saturated* hydraulic conductivity tensor $[L/T]$, with k_{rw} $[/]$ the relative permeability and $\mathbf{K}'_s = \rho g k/\mu$ the saturated hydraulic conductivity tensor $[L/T]$, being k is the intrinsic medium permeability $[L^2]$. Incorporating constitutive equations for density and viscosity, $\mathbf{K}'_s$ becomes:

$$\mathbf{K}'_s = \frac{(1+\epsilon c)}{(1+\epsilon' c)}\mathbf{K}_s$$

  with $\mathbf{K}_s$ the saturated hydraulic conductivity tensor at reference conditions μ_0 and ρ_0;

- $\sigma(\psi, c)$ is the overall storage coefficient $[L^{-1}]$;
- ϕ is the porosity $[/]$;
- S_w is the water saturation $[/]$ i.e. the ratio between the volume of water and the volume of voids in the *representative elementary volume* (REV) of porous medium. Obviously $S_w = 1$ in a water saturated porous medium;
- q is the injected (positive)/extracted (negative) volumetric [8] flow rate $[T^{-1}]$.

In the *transport equation* (second equation):

- $\mathbf{D}$ is the hydrodynamic dispersion tensor $[T^{-1}]$;
- c^* is the normalized concentration of salt in the injected(positive)/extracted(negative) fluid $[/]$;
- f is the volumetric rate of injected (positive)/extracted (negative) solute, in a limited quantity not capable to affect the flow field $[T^{-1}]$.

The expressions of the physical parameters listed above are given in the Appendix A. A detailed description of flow and solute transport processes can be found in classical subsurface hydrology textbooks [3, 7, 23].

Coupling in system (4) is due to the concentration terms that appear in the flow equation and, conversely, the head terms that appear in the transport equation via the Darcy velocities. As can be seen the coupling terms contain *nonlinear* expressions of the unknowns both in the flow and transport equations. An additional source of nonlinearity is introduced in the flow equation by the coefficients σ and k_{rw} that are highly nonlinear function of the pressure head ψ, when the water flow develops in the unsaturated zone [9] where ψ, now called the *suction head*, becomes lesser than zero, due to the capillarity effect.

[8] Per unit volume of porous medium.

[9] Where the void space of the porous medium is filled with both air and water.

Initial conditions (IC's) and Dirichlet, Neumann, or Cauchy boundary conditions (BC's) must be added to complete the mathematical formulation of the density-dependent flow and transport problem expressed in (4).

The **flow boundary conditions** are:

$$
\begin{array}{rcll}
\psi(x,y,z,t=0) & = & \psi_o(x,y,z) & \text{on } \Omega \quad \textit{(flow IC's)} \\[1em]
\psi(\overline{x},\overline{y},\overline{z},t) & = & \overline{\psi}(\overline{x},\overline{y},\overline{z},t) & \text{on } \Gamma_1 \quad \textit{(flow BC's)} \\
\mathbf{v}\cdot\mathbf{n} & = & -q_n(\overline{x},\overline{y},\overline{z},t) & \text{on } \Gamma_2
\end{array}
\qquad (7)
$$

having denoted Ω as the whole 3-D computational domain and Γ as its boundary, ψ_o is the prescribed pressure head at the initial time (zero) for all the points belonging to the volume Ω, $\overline{\psi}$ is the prescribed pressure head on the Dirichlet boundary segment Γ_1 of the surface Γ and q_n is the prescribed flux across the Neumann boundary segment Γ_2, whose outward normal unit vector is $\mathbf{n}$.

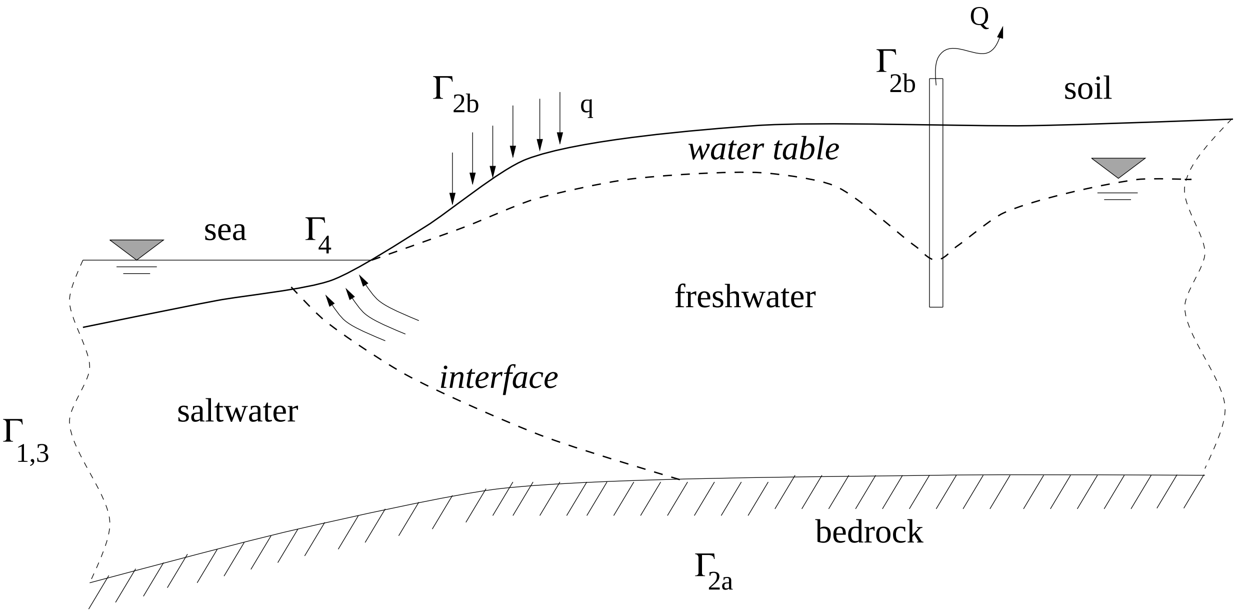

Figure 2: *Cross-section of the flow domain Ω representing a coastal aquifer, contaminated by saltwater intrusion, with its boundaries: Γ_1 at the sea side with a prescribed distribution of seawater pressure heads, Γ_{2a} on the aquifer bottom with zero water flux ($q_n = 0$), Γ_{2b} on the aquifer surface with prescribed distributed water inflow q (infiltration) and prescribed concentrated water outflow Q (withdrawal); Γ_3 at the sea side with prescribed salt concentrations and above it a freshwater outlet face Γ_4, where zero concentration gradients $\nabla c = 0$ are usually assumed. The drawn interface between salt- and freshwater is actually to be considered as a transition zone, where relative concentration c continuously varies from unity to zero when proceeding from the sea to the land.*

The **transport boundary conditions** are:

$$
\begin{aligned}
c(x,y,z,t=0) &= c_o(x,y,z) && \text{on } \Omega \quad \textit{(transport IC's)} \\
c(\overline{x},\overline{y},\overline{z},t) &= \overline{c}(\overline{x},\overline{y},\overline{z},t) && \text{on } \Gamma_3 \quad \textit{(transport BC's)} \\
D\nabla c \cdot \mathbf{n} &= q_d(\overline{x},\overline{y},\overline{z},t) && \text{on } \Gamma_4 \\
(\mathbf{v}c - D\nabla c)\cdot \mathbf{n} &= -q_c(\overline{x},\overline{y},\overline{z},t) && \text{on } \Gamma_5
\end{aligned}
\qquad (9)
$$

where c_o is the initial concentration at time zero, $\overline{c}$ is the prescribed concentration on the Dirichlet boundary segment Γ_3, q_d is the prescribed *dispersive flux* across the Neumann boundary segment Γ_4 and q_c is the prescribed *total flux* (advective plus dispersive) of solute across the Cauchy boundary segment Γ_5.

Appendix A also includes the description of the physical parameters appearing in the IC and BC formulations (7) and (9).

Figure 2 shows typical boundary conditions for a coastal aquifer, contaminated by seawater intrusion due to inland overpumping. We consider prescribed pressure (on Γ_1) and concentration (on Γ_3) Dirichlet boundaries at the sea-side, impervious boundary (no water flux) at the aquifer bottom (Γ_{2a}), assumed distributed influx q (infiltration) along soil surface (Γ_{2b}) and prescribed exploitation Q at the wells (Γ'_{2b}). Since the model assumes hydrodynamic dispersion, the drawn interface between salt- and freshwater is actually to be interpreted as a *transition zone*, where relative concentration continuously varies from unity to zero moving landward. Along the coast, accordingly to a dynamic water balance budget, is to be considered also a seepage face Γ_4 allowing the lighter freshwater to discharge into the sea. Miscible models usually assume along this outlet window Γ_4 no spatial variation of salt concentrations ($\nabla c = 0$).

2.2 Numerical model

The numerical model CODESA-3D is a standard finite element (FE) Galerkin scheme, with tetrahedral finite elements and linear basis functions, complemented by a weighted finite difference (FD) scheme for the discretization of the time derivatives.

With reference to a 3D domain composed by N nodes and N_e finite elements, the solutions sought $\psi(x,y,z,t)$ and $c(x,y,z,t)$ are expressed as linear combination of the nodal discretized unknowns $\hat{\boldsymbol{\psi}} = [\hat{\psi}_1, \hat{\psi}_2, \ldots, \hat{\psi}_N]^T$ and $\hat{\mathbf{c}} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_N]^T$ through linear shape functions N:

$$\psi(x,y,z,t) \approx \hat{\psi} = \sum_{i=1}^{N} N_i(x,y,z)\hat{\psi}_i(t)$$

and

$$c(x,y,z,t) \approx \hat{c} = \sum_{i=1}^{N} N_i(x,y,z)\hat{c}_i(t)$$

The expression of linear basis functions N_i for tetrahedral element can be found in classical FE text-books [34, 18]. The complete FE formulation of the CODESA-3D can be found in [15].

The FE discretization of system (4) finally yields the following system of ordinary differential equations (ODEs) in the nodal discretized unknowns $\hat{\psi}$ and $\hat{c}$, which represents the CODESA-3D **numerical semi-discretized model**:

$$\begin{aligned} \mathbf{P}\frac{d\hat{\boldsymbol{\psi}}}{dt} + \mathbf{H}\hat{\boldsymbol{\psi}} + \mathbf{q}^* &= \mathbf{0} \\ \mathbf{M}\frac{d\hat{\mathbf{c}}}{dt} + (\mathbf{A}+\mathbf{B}+\mathbf{C})\,\hat{\mathbf{c}} + \mathbf{r}^* &= \mathbf{0} \end{aligned} \qquad \textit{(ODE system)} \qquad (11)$$

In the above system, for the flow equation:

- $\mathbf{P}(\hat{\boldsymbol{\psi}}, \hat{\mathbf{c}})$ is the flow mass (or capacity) matrix;
- $\mathbf{H}(\hat{\boldsymbol{\psi}}, \hat{\mathbf{c}})$ is the flow stiffness (or flux) matrix;
- $\mathbf{q}^*(\hat{\boldsymbol{\psi}}, \hat{\mathbf{c}})$ is a vector accounting for the prescribed boundary fluxes, withdrawal or injection rates, the gravitational gradient term, and the time variation of the concentration,

and for the transport equation:

- matrices $\mathbf{A}(\hat{\boldsymbol{\psi}}, \hat{\mathbf{c}})$, $\mathbf{B}(\hat{\boldsymbol{\psi}}, \hat{\mathbf{c}})$ and $\mathbf{C}(\hat{\boldsymbol{\psi}}, \hat{\mathbf{c}})$ represent advective, dispersive and the Cauchy-BC-condition contributions to the overall transport stiffness matrix, that in the following will be shortly indicated as matrix $\mathbf{E} = (\mathbf{A}+\mathbf{B}+\mathbf{C})$;
- $\mathbf{M}(\hat{\boldsymbol{\psi}})$ is the transport mass matrix;
- vector $\mathbf{r}^*$ accounts for source and sink terms, and for the dispersive component of the Neumann and Cauchy boundary conditions.

Flow matrices **H** and **P** are symmetric positive definite matrices (SPD) [10]. Transport matrices **A**, **C** and **M** are also SPD matrices, while advective transport matrix **B** is a unsymmetric matrix.

Model parameters of system (11) that are spatially dependent are considered constant within each tetrahedral element. Parameters that depend on pressure head and/or concentration are evaluated using ψ and/or c values averaged over each element and are also element-wise constant.

The numerical integration of matrix and vector coefficients of system (11) is derived in [15]; Appendix A provides also the expressions of these matrices and arrays.

After the spatial discretization using FE, the system (11) is integrated in time using Finite Differences (FD). Denoting by dy/dt the generic time derivative (with $y = \psi, c$), we have

$$dy/dt = (y^{k+1} - y^k)/\Delta t_k$$

where Δt_k is the time interval between current t_{k+1} and previous t_k time steps. Matrices and vectors, generically indicated here with variable $\mathbf{Y}$, that are function of the unknowns, are evaluated at the time $(t + \omega\Delta t)$ using the *trapezoidal rule*:

$$\mathbf{Y}^{k+\omega} = \omega\mathbf{Y}^k + (1-\omega)\mathbf{Y}^{k+1}$$

[10] A symmetric matrix $(n \times n)$ is positive definite if all its eigenvalues are positive.

with ω the weighting parameter. The *forward Euler* scheme is obtained for $\omega = 0$, the *backward Euler* scheme is obtained for $\omega = 1$ and the Crank-Nicolson scheme for $\omega = 0.5$. For numerical stability of the integration scheme, the weighting parameter ω must satisfy the condition $0.5 \leq \omega \leq 1$. Applying the weighted FD scheme to the equations of system (11), with weighting parameter ω_f for the flow equation and ω_t for the transport equation [11], yields the following **system of nonlinear algebraic equations**:

$$
\begin{aligned}
\left(\frac{\mathbf{P}^{k+\omega_f}}{\Delta t_k} + \omega_f \mathbf{H}^{k+\omega_f}\right)\hat{\boldsymbol{\psi}}^{k+1} &= \left[\frac{\mathbf{P}^{k+\omega_f}}{\Delta t_k} - (1-\omega_f)\mathbf{H}^{k+\omega_f}\right]\hat{\boldsymbol{\psi}}^{k} - \mathbf{q^*}^{k+\omega_f} \\
\left(\frac{\mathbf{M}^{k+1}}{\Delta t_k} + \omega_t \mathbf{D}^{k+\omega_t}\right)\hat{\mathbf{c}}^{k+1} &= \left[\frac{M^{k}}{\Delta t_k} - (1-\omega_t)\mathbf{D}^{k+\omega_t}\right]\hat{\mathbf{c}}^{k} - \mathbf{r^*}^{k+\omega_t}
\end{aligned} \tag{13}
$$

which may be concisely rewritten:

$$
\begin{aligned}
\mathbf{A}_f^{k+1} \cdot \hat{\boldsymbol{\psi}}^{k+1} &= \mathbf{b}_f \\
\mathbf{A}_t^{k+1} \cdot \hat{\mathbf{c}}^{k+1} &= \mathbf{b}_t
\end{aligned} \tag{15}
$$

with $\mathbf{A}_f$ and $\mathbf{A}_t$ the coefficient matrices (also called the left hand side (LHS)), $\hat{\boldsymbol{\psi}}$ and $\hat{\mathbf{c}}$ the vectors of unknowns and $\mathbf{b}_f$ and $\mathbf{b}_t$ the right hand side (RHS) vectors of the flow and transport equations, respectively.

At this point of the discretization process, the first equation (*variably saturated flow*) of system (15) needs special attention for its intrinsic nonlinearity due to the presence of the nonlinear coefficients σ and k_{rw} [3] incorporated in matrices $\mathbf{P}$ and $\mathbf{H}$, respectively. The linearization techniques adopted in CODESA-3D are the *Picard iteration*, also known as the method of successive substitution (SS), and the *Newton's method*, also known as Newton-Raphson [5]. Again all the numerical discretization details are described in [15]. For the purposes of the present report, at the end of the adopted linearization scheme of the flow equation in (15) the following **linearized flow equation** is obtained:

$$(\tilde{\mathbf{A}}_f)_{m+1}^{k+1} \cdot \hat{\mathbf{s}} = (\tilde{\mathbf{b}}_f)_{m+1}^{k+1} \tag{16}$$

where $(m+1)$ is the current nonlinear flow iteration step, $(\tilde{\mathbf{A}}_f)$ is the *linearized* flow matrix and the nodal discretized vector $\hat{s}= (\hat{\psi}_{m+1}\text{-}\hat{\psi}_m)$ is called the *search direction* vector.

A difference between the two cited linearization schemes is that Picard linearization generates a symmetric flow coefficient matrix $\tilde{\mathbf{A}}_f$, thus preserving the symmetry of the original matrix of the discrete flow equation, whereas Newton iteration generates a unsymmetric matrix $\mathbf{A}_f$, which is called the *Jacobian* [12]. This fact has some relevance on the choice of the linear

[11] Weighting parameters ω_f and ω_t are not required to be the same.
[12] The right hand side of the Newton's scheme is called the *residual*.

solver and on its computational cost (memory and CPU requirements) and efficiency [30]. On the other hand Newton iteration is known as more efficient method in presence of more pronounced nonlinearity effects. The peculiarity of the two methods for flow and transport applications are reported in [26, 28].

The system (15), incorporating the linearized flow equation (16) can be rewritten at each time step $k+1$ and for each innermost *nonlinear flow iteration* step $m+1$ as:

$$\begin{aligned} (\tilde{\mathbf{A}}_f)_{m+1} \cdot \hat{\mathbf{s}} &= (\tilde{\mathbf{b}}_f)_{m+1} \\ \mathbf{A}_t \cdot \hat{\mathbf{c}} &= \mathbf{b}_t \end{aligned} \qquad \textit{(nonlinear system 1.2)} \qquad (18)$$

where matrix $\tilde{\mathbf{A}}_f$ can be symmetric or not depending on which linearization method is adopted and matrix $\mathbf{A}_t$ is always unsymmetric due to the contribution of convective transport stiffness matrix **B**. Remember that $\hat{\mathbf{s}} = (\hat{\psi}_{m+1} - \hat{\psi}_m)$, thus the linearized flow equation is solved at each linearization step in terms of equivalent freshwater head increments $\hat{\mathbf{s}}$.

Neumann and Cauchy boundary conditions are incorporated in the matrix and vector coefficients of the FE formulation, as described in [15], while Dirichlet boundary conditions are finally imposed on the left and right hand sides of the assembled discrete equations of system (18).

The resulting nonlinear system (18) is solved using the successive substitution (SS) scheme (Picard linearization) which for a given time step $k+1$ reads:

1. $\hat{\mathbf{c}}^{k+1}_{n=0} = \hat{\mathbf{c}}^k$;
2. *For* $n = 0, \dots, \overline{n}$
3. $\quad \hat{\mathbf{c}}^{k+1}_{n+1} = \hat{\mathbf{c}}^k_n$;
4. solve the flow equation: $(\tilde{\mathbf{A}}_f)_{m+1} \cdot \hat{\mathbf{s}}^{k+1}_{n+1,m+1} = (\tilde{\mathbf{b}}_f)_{m+1}$ with an *innermost flow linearization scheme*;
5. *If* ($n > 0$ .and. $\|\hat{\mathbf{c}}^{k+1}_{n+1} - \hat{\mathbf{c}}^k_n\| < \epsilon_t$) *Then*
   Stop
   Else
   solve the transport equation: $\mathbf{A}_t \cdot \hat{\mathbf{c}}^{k+1}_{n+1} = \mathbf{b}_t$;
   End If
6. *End For*

In the above algorithm ϵ_t is the loop exiting tolerance and n is the loop index with maximum step counter $\overline{n}$. This scheme will be shortly indicated with SS($\overline{n}$,ϵ_t). Note that step 4 in the above algorithm is solved using an innermost linearization scheme with tolerance ϵ_f and maximum step counter $\overline{m}$, either Picard or Newton scheme.

Since the initial solution estimate y^0, with $y = (\psi, c)$ has a big effect on the convergence behavior of the adopted linearization schemes often, a *relaxed iteration*, defined as:

$$y^{m+1} = y^m + \lambda(y^{m+1} - y^m)$$

with λ damping parameter [13], is adopted to accelerate convergence when a poor initial estimate is used.

Linear systems of general type: $Ax = b$, in step 4 and 5 (else branch) are solved using Krylov subspace iterative methods [30] for symmetric and unsymmetric systems. Some of the iterative methods available in CODESA-3D are:

- Symmetric positive definite matrices (SPD):
  Conjugate Gradient Method [17].
- Unsymmetric matrices [31, 32, 8]:
  BiConjugate Gradients STABilized (BCGSTAB);
  Transpose Free Quasi-Mimimal Residuals (TFQMR);
  minimum residuals (GRAMRB);

The exit condition of the linear solver iteration is given by

$$\frac{||r||_2}{||b||_2} < \epsilon_s$$

with $r = (b - Ax)$ the residuals and ϵ_s the prescribed tolerance for solver convergence, usually in the range of $10^{-8} \div 10^{-12}$, for floating point calculation having 8 byte precision. All these solvers are preconditioned, in order to accelerate convergence, using the *incomplete* LU decomposition [14] or the main diagonal of the coefficient matrix A.

[13] Damping parameters λ_f, adopted in the linearization of the flow equation, and λ_t, adopted in the linearization of the coupled system, are obviously not required to be the same.
[14] In the symmetric case the *incomplete* Cholesky LL^T decomposition is used.

3 Implementation issues

The chapter describes the CODESA-3D *implementation model* including: the computer code design and structure, and the data organization as well.

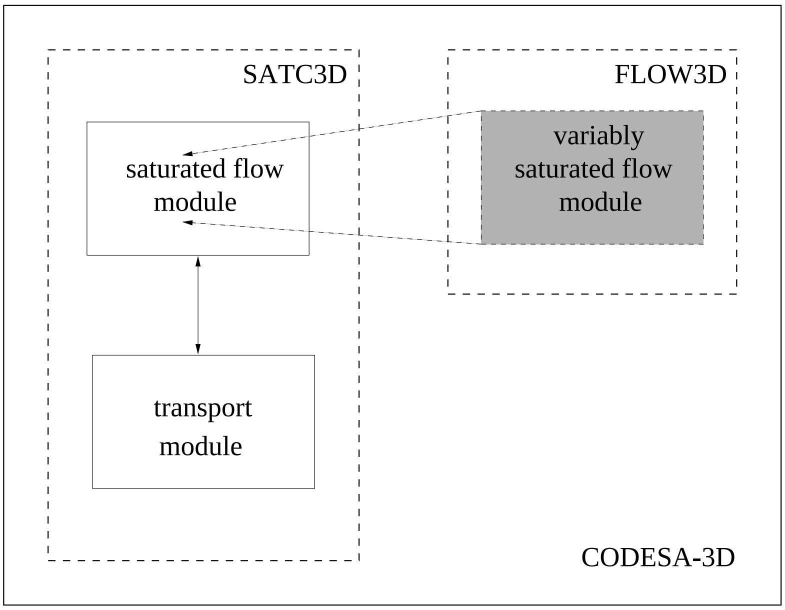

Figure 3: *Integration of FLOW3D* variably saturated *flow module in place of the corresponding SATC3D* saturated *flow module for the creation of the CODESA-3D model.*

3.1 Code design

3.1.1 Integration of existing computer codes

CODESA-3D code skeleton originates from SATC3D [14], a coupled *saturated* flow and miscible transport simulator. Like SATC3D, CODESA-3D is a coupled flow and transport simulator, but in addition it allows for a *variably saturated* flow regime. In doing this extension, CODESA-3D incorporates (Figure 3) the pre-existing *freshwater* flow model FLOW3D [11, 13], integrating it with the presence of the diluted salt, thus giving rise to a flow simulator for a variably dense fluid in a variably saturated porous medium. The integration required model extensions both in the incorporated *flow module*, due to the introduction of the salt concentration terms which affects the water density, and in the *transport module*, which, conversely, deals now with a variably saturated flow field. The principal integration in the CODESA-3D, with reference to the SATC3D code, was the introduction of the innermost linearization scheme required by the solution of the variably saturated (*nonlinear*) flow equation (16). This piece of software was collected from FLOW3D routines. Apart from the linearization of the flow equation, other extensions to the original codes SATC3D and FLOW3D and planned future extensions to reach the final version of the CODESA-3D model [15] are summarized in Appendix A.

Parent codes (FLOW3D, SATC3D) integration in CODESA-3D has been done exploiting a series of features of Fortran 90 programming language: the *dynamic memory allocation* and *derived data structures*, which are described in the following paragraphs.

3.1.2 Dynamic memory allocation

For this software integration project, an important point in favor of dynamic memory allocation was the possibility to allocate exactly the memory size amount required for the given simulation, especially useful for large scale problems.

Parent Fortran 77 codes used to define, *statically*, the dimensioning parameters of the test case in a include file `<codename>.H`. The following lines show the include file `CATHY.H` from FLOW3D source distribution:

```
C this file:  CATHY.H
C-----------------------------------------------------------------------
C Dimensioning Parameters
C NODMAX - maximum # of surface nodes in 3-d mesh
C NTRMAX - maximum # of triangles in 2-d mesh
C MAXSTR - maximum # of vertical layers
C NMAX - NODMAX*(MAXSTR + 1) maximum # of nodes in 3-d mesh
C NTEMAX = 3*NTRMAX*MAXSTR - maximum # of tetrahedra in 3-d mesh
C N1MAX - maximum # of element connections to a node
C MAXTRM = N1MAX*NMAX - maximum # of nonzero elements in system matrices
C MAXBOT - maximum size of NONSYM real working storage
C INTBOT - MAXBOT + 6*NMAX + 1
C
             INTEGER NODMAX, NTRMAX, MAXSTR
             INTEGER NMAX, NTEMAX
             INTEGER N1MAX, MAXTRM
             INTEGER MAXBOT, INTBOT
             PARAMETER (NODMAX = 2116, NTRMAX = 4050, MAXSTR = 42)
             PARAMETER (NMAX = NODMAX*(MAXSTR+1),NTEMAX = 3*NTRMAX*MAXSTR)
             PARAMETER (N1MAX = 20,MAXTRM = N1MAX*NMAX)
             PARAMETER (MAXBOT = 90000,INTBOT = MAXBOT+6*NMAX+1)
C-----------------------------------------------------------------------
```

Arrays were thus declared using the predefined parameters of the include file (`CATHY.H`), which was accessed by each program unit through the keyword `INCLUDE` (see example below):

```
C An example of static allocation of array X
C-----------------------------------------------------------------------
      PROGRAM MAIN
      IMPLICIT NONE
      INCLUDE 'CATHY.H'
      ...
      REAL*8 X(NMAX)
      ...
C-----------------------------------------------------------------------
```

When a dataset did not fit the declared dimensioning parameters, these had to be enlarged, editing the include file and recompiling all the source codes that include it.

Now instead, using dynamic memory allocation the equivalent portion of the code is:

```
C An example of dynamic allocation of array X
C-----------------------------------------------------------------------
            ...
            USE MOD_KIND ! specification of real precision
            ...
            INTEGER ::  N
            REAL (MY_PRECISION), POINTER, DIMENSION (:)  ::  X
C
            READ(*,*) N
            IF(N.GT.0)THEN
               ALLOCATE (X(N), STAT=err)
               IF(err.NE.0)STOP 'WARNING: error during allocation'
            ELSE
               STOP 'WARNING: N.LE.0'
            ENDIF
            ...
C-----------------------------------------------------------------------
```

In the example above a monodimensional (1-D) array is declared using a 1-D pointer [15] (<type>, POINTER, DIMENSION (:)). The actual memory size required by the vector is read from the standard input at run time, when it is exactly known the size of the problem at hand. The array associated memory may be eventually released and re-allocated to another array during program execution.

The present version of CODESA-3D uses only dynamic memory.

Aside from the main point of dynamic allocation, in the code fragment above there is another Fortran 90 interesting feature. The precision of a real variable X is set to the value MY_PRECISION, which in turn is defined in the module program unit [16] MOD_KIND. Such module

[15] A 2-D pointer is declared as: <type>, POINTER, DIMENSION (:,:) and a 3-D one as: <type>, POINTER, DIMENSION (:,:,:) and so on.

[16] Modules are third type of program units besides main (PROGRAM) and external subprograms (SUBROUTINE, FUNCTION) which can contain both data and instructions. A typical Fortran 90 module is a library object.

unit is made visible to this program through the keyword USE. Below is shown the content of MODULE MOD_KIND:

```
C this file:  mod_kind.f
C-----------------------------------------------------------------------
C Set real number precision by uncommenting the proper line
C
             MODULE MOD_KIND
c 4 byte precision
ccccc          INTEGER, PARAMETER ::  MY_PRECISION = 4
c 8 byte precision
                INTEGER, PARAMETER ::  MY_PRECISION = 8
             END MODULE MOD_KIND
C-----------------------------------------------------------------------
```

Changing the floating point precision of computations requires simply editing the content of MODULE MOD_KIND and recompiling all the source files that use that module, without the need of editing each single routine of the entire source distribution.

3.1.3 Data organization using derived data structures

Another important Fortran 90 feature incorporated in CODESA-3D code is the use of derived data types, which are compound variables made by combination of the basic intrinsic data types (INTEGER, READ, LOGICAL, etc.) and shapes (scalar, vector, matrix, pointer etc.).

This extension was introduced to obtain the desired flexibility needed during the integration of the existing routines. The major benefits of the practice are:

- the great compactness in passing the actual argument list to the called routines, moving from a long list of intrinsic variables (typically 20÷30 items) to a shorter list of compound variables (typically up to 10 items);
- the existence of a unique place in the code where the derived variables are defined. All the prototypes are contained in MODULE units;
- the existence of a unique place in the code where the derived variables are instantiated with actual dimensions. All the allocations are in allocate.f file.

These three functionalities greatly minimize coding mistakes and further error checking. The number and composition of these derived data types were carefully based on the code organization. Indeed the list of implemented derived types reflects the main "recipients" of a typical discretized (FE) model and the relationships between them. Eleven (11) major derived classes were defined and 3 minor ones for a total of 14 classes:

1. mod_Dim: collection of *actual* dimension parameters read at run-time form input files;
2. Par_tag: collection of all the other principal parameters;
3. CPU_tag; collection of CPU timing variables;

4. IO_tag: collection of logical units and file names;
5. Grid_tag: collection of variables related to mesh definition;
6. Flow_BC_tag: collection of variables related to flow BC's;
7. Transp_BC_tag: collection of variables related to transport BC's;
8. MBal_tag: collection of variables related to mass balance computations;
9. MS_prop_tag: collection of variables related to material (porous medium) and solute (salt) properties. This major class incorporates also three subclasses related to the moisture-retention soil properties [25]:
   (a) VG_tag: set of Van Genutchen curve parameters.
   (b) HU_tag: set of Huyakorn curve parameters;
   (c) BC_tag: set of Brooks-Corey curve parameters;
10. Sys_tag: collection of variables related to discretized linear systems;
11. Out_tag: collection of variables related to model output.

The Naming convention is such that the derived variable named <var> of type <var>_tag is declared in the module mod_<var> contained in file mod_<var>.f.

As an example, the derived data type Grid_tag is defined in module mod_Grid as follows:

```
C this file: mod_Grid.f
C-----------------------------------------------------------------------
C     derived data type containing all the grid parameters and data
      MODULE MOD_GRID
        USE MOD_KIND    ! specification of real precision
          TYPE GRID_TAG
          INTEGER                                     :: IVERT, ISP
          INTEGER,             POINTER, DIMENSION (:)   :: IVOL,TP
          INTEGER,             POINTER, DIMENSION (:,:) :: TRIANG, TETRA
          REAL (MY_PRECISION)                           :: BASE
          REAL (MY_PRECISION), POINTER, DIMENSION (:)   :: AREANOD, X, Y, Z, ZRATIO,
     >                                                    VOLNOD, VOLU, VOLUR
        END TYPE GRID_TAG
      END MODULE MOD_GRID
C-----------------------------------------------------------------------
```

The compound variable GRID contains variables and arrays related to the definition of the computational grid. See Appendix B for the meaning of a single variable.

Relevant subfields of structure GRID are the integer connectivity matrices Grid%TRIANG ($4 \times$ Dim%NTRI) and Grid%TETRA ($5 \times$ Dim%NT) describing the initial 2-D ground surface and the whole 3-D aquifer meshes, respectively. In particular, as depicted in Figure 4, the e-th column of matrix TRIANG contains the 3 nodes (P,S, and U in the Figure) belonging to e-th triangle while the fourth location contains an integer identifier of the the hydrogeological zone (represented as gray patches in the Figure) of that finite element; analogously the e-th column of matrix TETRA contains the 4 nodes (I,J,L, and M) belonging to e-th tetrahedron while the fifth location identifies the hydrogeological zone. Physical parameters which are assigned according to

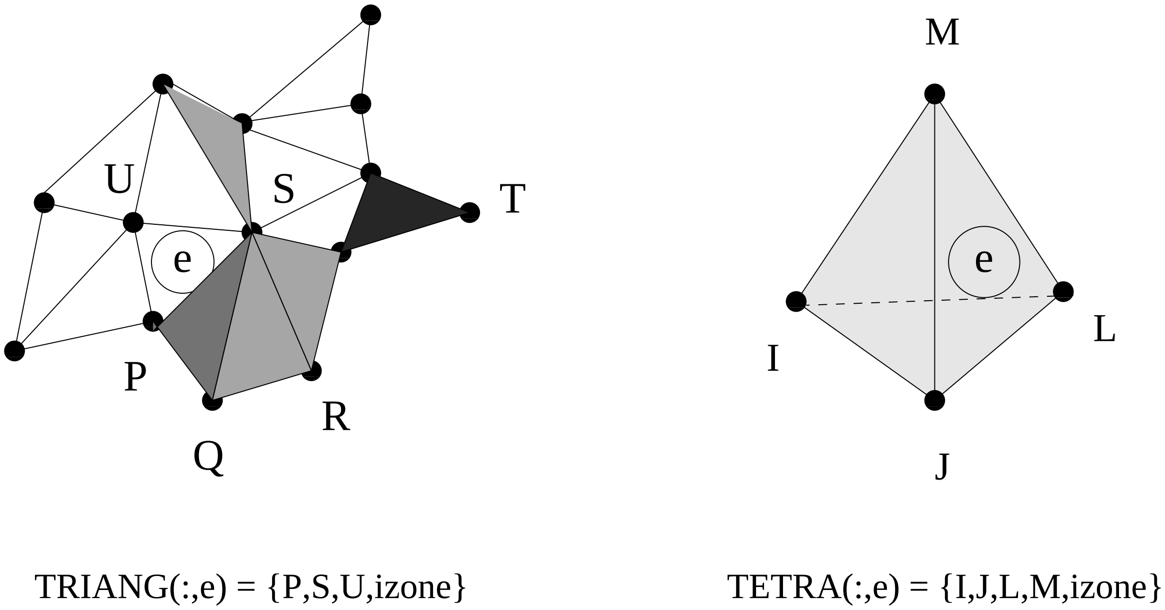

Figure 4: *Connectivity arrays* `TRIANG` *and* `TETRA`*, which define the list of nodes and the hydrogeological zone (depicted as gray patches) of the triangles and tetrahedra belonging to the 2-D ground surface and 3-D aquifer grids, respectively.*

hydrogeological zones are saturated hydraulic conductivity K, porosity ϕ, elastic storage S_s and longitudinal α_L and transversal α_T dispersivities. The maximum number of zones in CODESA-3D is given by the number of 2-D superficial triangles `Dim%NTRI` multiplied by the number of vertical strata `Dim%NST`. [17]

In the code fragment that follows we see that the prototype of the derived variable `GRID` is made visible to all program units which have the instruction `USE MOD_GRID`. The actual compound variable `GRID` of type `GRID_TAG` is declared, in each program unit, with the instruction `TYPE (GRID_TAG) :: GRID`, and it is allocated with actual dimension only once at the beginning of the run, as in the code fragment that follows:

[17]The vertical zoning in CODESA-3D is formulated as a projection of the superficial one, thus one is allowed to modify only the specific hydrogeological value assigned to a vertical zone but not its lateral extension.

```
C An example of allocation of some subfields of compound variable GRID
C-----------------------------------------------------------------------
             ...
             USE MOD_KIND ! specification of real precision
             USE MOD_GRID ! prototype of derived grid data
             ...
             IMPLICIT NONE
             TYPE (GRID_TAG) ::  GRID ! declaration of compound variable GRID
             ...
             IF(Dim%n.le.0)THEN
                  STOP 'error:  n.le.0 '
             ENDIF
             ALLOCATE (Grid%x(Dim%n),stat=err) !  dimension N
                IF(err.ne.0)STOP
             ALLOCATE (Grid%y(Dim%n),stat=err)
                IF(err.ne.0)STOP
             ALLOCATE (Grid%z(Dim%n),stat=err)
                IF(err.ne.0)STOP
             ...
C-----------------------------------------------------------------------
```

The compound variable `GRID` is then used as shown in the following code fragment:

```
C An example of use of some subfields of compound variable GRID
C-----------------------------------------------------------------------
             ...
             TYPE (GRID_TAG) ::  GRID ! declaration of compound variable GRID
             ...
c read X and Y coordinate values
             READ(IO%IN2,*)(Grid%X(I),Grid%Y(I),I=1,Dim%NNOD)
c read only surface elevation values
             READ(IO%IN2,*)(Grid%Z(I),I=1,Dim%NNOD)
             ...
c generate 3D mesh with vertical projection of ground surface 2D triangulation
             CALL GRID3D(Dim,Par,IO,Grid,Flow_BC,Transp_BC)
             ...
C-----------------------------------------------------------------------
```

Note that a single field of a derived data structure is accessed using % operator. Recursively a derived data structure may be a subfield of an higher-level derived data structure, thus defining a hierarchy of data. In CODESA-3D, for instance, the real coefficient `ALFA` of Huyakorn moisture-retention curves [25] is defined as a subfield of structure `HU`, which is in turn a subfield of the material and solute class `MS_prop`; to access variable `ALFA` we write: `MS_prop%HU%ALFA`.

As we can see in the code fragment above, the external routine `GRID3D` is called with all the derived variables to be read/written passed as a very compacted list (6 items) of actual arguments. As shown this procedure greatly enhances code compactness and readability; on the other hand it may eventually introduce some overhead during compilation and execution phases. Efficient Fortran 90 compilers greatly reduce these overheads.

3.2 The code structure

The original 107 FLOW3D and 80 SATC3D routines have been maintained, with minor modifications, in CODESA-3D, which is composed by about 175 routines, for a total of about 20 000 Fortran source lines (including comments).

The naming convention is such that `SUBROUTINE PIPPO` is contained in file named `pippo.f`.

3.2.1 Level 0: `PROGRAM MAIN_CODESA3D`

The CODESA-3D code is organized in a main program `PROGRAM MAIN_CODESA3D` which drives the actual computational unit called `SUBROUTINE CODESA3D`. The main program skeleton, shown in the shaded box below, does the following operations: open input/output files, read from these input files the dimensioning parameters of the problem at hand and then rewind files, allocate arrays using actual memory requirements, run the computations, deallocate associated memory and close input/output files prior to exit.

```
C this file:  main_codesa3d.f
C---------------------------------------------------------------------
C DRIVER of CODESA-3D
       ...
       IMPLICIT NONE
       ...
C open input files
       CALL OPENIO (Par,IO)
C read actual dimensioning parameters from input files
       CALL READ_DIM (Dim,Par,IO)
C allocate arrays dynamically with actual dimensions
       CALL ALLOCATE (Dim,Par,IO,Grid,Flow_BC,Transp_BC,MS_prop,Sys,Out)
C do computations
         CALL CODESA3D (Dim,Par,IO,Grid,Flow_BC,Transp_BC,MS_prop,Sys,Out)

C close all input files
       CALL CLOSIO (Par,IO)
C deallocate dynamic arrays
       CALL DEALLOCATE (Dim,Par,IO,Grid,Flow_BC,Transp_BC,MS_prop,Sys,Out)
       ...
       END
C---------------------------------------------------------------------
```

3.2.2 Level 1: SUBROUTINE CODESA3D

It is the called SUBROUTINE CODESA3D that manages the computations allowing two types of simulations:

- freshwater flow alone (Par%ITRANS = 0);
- *density dependent* coupled flow and transport (Par%ITRANS $\neq$ 0).

The actual arguments of SUBROUTINE CODESA3D are simply 9 of the 11 principal derived data types, which are easily passed in cascade to the called routines. The code structure of SUBROUTINE CODESA3D is shown below:

```
C this file:  codesa3d.f
C-----------------------------------------------------------------------
      SUBROUTINE CODESA3D (Dim,Par,IO,Grid,Flow_BC,Transp_BC,MS_prop,Sys,Out)
      ...
C initialization
      ...
C start new time step
C---------------------------------------------------NEW TIME STEP
      DO WHILE ((.NOT. Par%flow_exit_flag) .AND. (.NOT. Par%coupled_exit_flag))
C
C case (1) ---> ITRANS=0:  variably saturated freshwater flow only
        IF (Par%ITRANS .EQ. 0) THEN
C Assemble and solve for unknown pressure heads
          CALL FLOWSOLVE (Dim,Par,IO,CPU,Grid,Flow_BC, MS_prop,MBal,Sys,Out)
C
C case (2) ---> ITRANS .ne.  0:  coupled flow & transport
        ELSE
C Assemble and solve for unknown pressure heads and concentrations
          CALL FLOWTRANSOLVE (Dim,Par,IO,CPU,Grid, Flow_BC,Transp_BC,
     >              MS_prop,MBal,Sys,Out)
         END IF
      END DO
C---------------------------------------------GO TO THE NEXT STEP
C ending
      ...
      RETURN
      END
```

The **initialization phase** for SUBROUTINE CODESA3D includes: complete reading of input files, generation of the 3-D mesh, allocation and set up of arrays for storage of the assembled sparse matrices [18] in the Compressed Sparse Row (CSR) format [2], initialization and set up of simulation parameters, variables and arrays. The **ending phase** includes: output of final

[18] The global matrices arising from FE formulations are very sparse since each row, which corresponds to a mesh node, collects contributions only from the mesh near neighbors of the node. Only the non-zero coefficients of these

results, cumulative mass balance errors, flag and failure summaries, CPU times both for code sections and total execution, and deallocation of assembled matrices of actual dimension `Dim%NTERM` and `Dim%NTERMC`. All these phases are common to the simply flow (1) and the flow & transport simulations (2). Also the `DO WHILE` loop of time stepping is common to the (1)/(2) branches of the IF-ENDIF construct, while the set up of the time marching marching parameters (`Par%DELTAT`, `Par%TIME` etc.) is done at the lower code structure level (`SUBROUTINE FLOWSOLVE`, `FLOWTRANSOLVE`).

The 3-D Mesh generation is accomplished by replication, for a given number of vertical layers (`Dim%NTRI+1`), of the given 2-D ground surface triangulation, composed by `Dim%NNOD` nodes and `Dim%NTRI` triangles. Since the vertical projection of each 2-D triangle generates a 3-D prism (with a triangular basis) and in turn each prism generates 3 tetrahedra, the final 3-D mesh has `Dim%NNOD*(Dim%NSTR +1)` nodes and `Dim%NTRI*3*Dim%NSTR` tetrahedra. Different mesh generation options with reference to layer thicknesses are available according to integer parameters `Par%ISP`, `Par%IVERT` (see Appendix B).

The matrix memorization is row-wise compressed according to the CSR standard [2]. Figure 5 shows the contents of arrays (`COEF`, `JA`, `IA`) involved in symmetric (left) and unsymmetric (right) compressed storage [19].

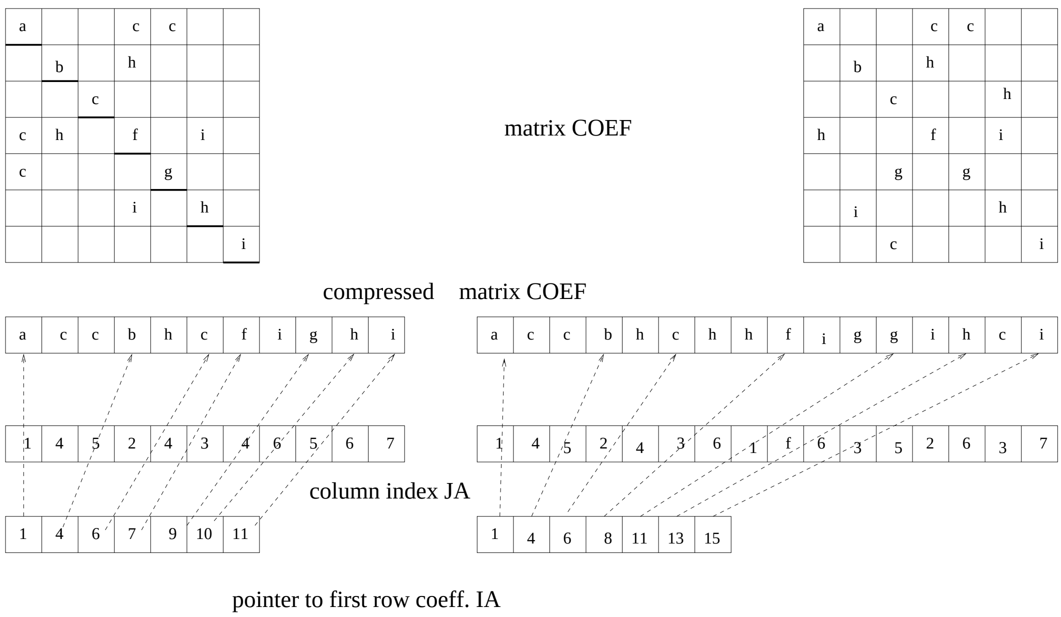

Figure 5: *Compacted storage of symmetric (left)/unsymmetric (right) matrices using the CSR format. Observe that only the upper part of a symmetric matrix is stored.*

The allocation of flow assembled matrices is done taking into account whether they are symmetric or not [20] for memory saving purposes. If both flow and transport coefficient matrices are unsymmetric [21] only one copy (that of flow) of CSR pointer arrays (column index `Sys%JA`, row pointer `Sys%TOPOL`, etc.) is computed and maintained. The corresponding transport CSR arrays are simply "pointers", with no associated memory, that address the flow counterparts,

matrices are stored in a row-wise fashion in a real vector and then integer vectors of indices are used to relate these vector locations to the original matrix coefficients.

[19] While `COEF` and `JA` in CODESA-3D maintain the same name, pointer array IA is called `TOPOL`.

[20] Only Picard iteration for the linearization of the flow equation gives rise to a symmetric system.

[21] Since transport matrix is always unsymmetric, this case occurs when Newton linearization is applied to the flow equation.

as it shown in the following code fragment taken from the SUBROUTINE CSR_PTR:

```
C An example of pointer association without new memory request (else branch)
C-----------------------------------------------------------------
      ...
            IF(Par%ITRANS.ne.0)THEN ! coupled flow & transport
                  IF(Par%IFLOW.eq.1)THEN ! flow symmetric storage
C compute dimension NTERMC for integer pointers and allocate them
                        CALL TRANSPCSR_PTR (Dim,Par,IO,Grid,Sys)
                  ELSE ! flow non-symmetric storage
C simply refer to previously allocated integer FLOW pointers
                        Dim%NTERMC = Dim%NTERM
                        Sys%JAC => Sys%JA
                        Sys%TOPOLC => Sys%TOPOL
                        Sys%TETJAC => Sys%TETJA
                  ENDIF
            ELSE ! only flow
      ...
            ENDIF
      ...
C-----------------------------------------------------------------
```

The Fortran 90 symbol => has the special meaning of pointer association.

3.2.3 Level 2: SUBROUTINE FLOWTRANSOLVE

At the secondary code structure level SUBROUTINE FLOWTRANSOLVE manages the solution of the coupled system (11) at each time step for the unknown pressure heads and concentrations [22]. The coupled system is linearized using a *successive substitution* (SS) scheme (see pseudocode in the box below the system (18)) allowing two types of linearization method (Picard and partial Newton), both for flow and transport equations. The choice of the linearization scheme is made upon the choices:

- flow equation
  - Par%IFLOW = 1, Picard scheme (SUBROUTINE PICFLW);
  - Par%IFLOW = 2, partial Newton scheme (SUBROUTINE NEWTFLW).
- transport equation
  - Par%ITRANS = 1, Picard scheme (SUBROUTINE PICCPL);
  - Par%ITRANS = 2, partial Newton scheme (SUBROUTINE NEWTCPL).

The shaded boxes below show the content of source file flowtransolve.f:

[22] Analogously SUBROUTINE FLOWSOLVE, which is not discussed here, solves the equation for unknown freshwater pressure heads, when a freshwater flow alone is simulated.

```
C this file:  flowtransolve.f (1rst part)
C-----------------------------------------------------------------------
      SUBROUTINE FLOWTRANSOLVE (Dim,Par,IO,CPU,Grid,Flow_BC,Transp_BC,
     >                    MS_prop,MBal,Sys,Out)
         ...
C Successive substitution (SS) scheme to solve the coupled system
C
C-------------------------CASE Par%ITRANS=1:  PICARD SCHEME
      IF (Par%ITRANS .EQ. 1) THEN
C
C Picard linearization of the transport equation
C
```

```
         CALL PICCPL (Dim,Par,IO,CPU,Grid, Flow_BC,Transp_BC,
        >                 MS_prop,MBal,Sys,Out)
```

```
C
C-------------------------CASE Par%ITRANS=2:  NEWTON SCHEME
C
      ELSE IF (Par%ITRANS .EQ. 2) THEN
C
C partial Newton linearization of the transport equation
C
         CALL NEWTCPL (Dim,Par,IO,CPU,Grid,Flow_BC,Transp_BC,
        >                 MS_prop,MBal,Sys,Out)
C
      END IF
C-----------------------------------------------------------------------
(.   .   .  to be continued on the next pages .   .   .)
```

After the solution of the system (SUBROUTINE PICCPL or NEWTCPL) some auxiliary calculations are made, as can be seen from the shaded box below, namely: check for time backstepping or stop conditions; output of the results; hydrograph outputs; update of flow and transport variables, processing of time step variables, interpolation of time-varying BC, prior to return the control to the callee routine CODESA-3D which increments the DO WHILE in the time step counter.

Observe that from this level below in the code structure coexist routines with both compound and intrinsic arguments.

```
C this file:  flowtransolve.f (2nd part)
(.   .   .  continuing from the previous pages .   .   .)
C
C handle the back-stepping case for the flow eqn.
C
    IF (Par%ICNVRF.EQ.-2) RETURN
C no back-stepping possible, simulation is terminated
C
    IF ((Par%ICNVRF .EQ. -3).OR.(Par%ICNVRC .EQ. -3)) THEN
     CALL CLOSIO (IO)
     STOP
    ENDIF
C back-calculate fluxes at Dirichlet and Cauchy nodes, and perform mass balance
C and hydrograph calculations; calculate soil moisture characteristics needed
C for velocity calculations and for output; and output mass balance errors
       ...
C flow eq.:  update a first block of variables and arrays
C before going to the next time step
C
    MBal%ETOT = MBal%ETOT + ABS(MBal%ERRAS)
    MBal%VTOT = MBal%VTOT + (MBal%VIN + MBal%VOUT)
    MBal%VTOTI = MBal%VTOTI + MBal%VIN
    MBal%VTOTO = MBal%VTOTO + MBal%VOUT
    MBal%ADINP = MBal%ADIN
    MBal%ADOUTP = MBal%ADOUT
    MBal%NDINP = MBal%NDIN
    MBal%NDOUTP = MBal%NDOUT
    MBal%ANINP = MBal%ANIN
    MBal%ANOUTP = MBal%ANOUT
    MBal%NNINP = MBal%NNIN
    MBal%NNOUTP = MBal%NNOUT
    MBal%SFFLWP = MBal%SFFLW
    MBal%SFFLWP = MBal%SFFLW
C
    Flow_BC%QPOLD = Flow_BC%QPNEW
    Flow_BC%SFEXP = Flow_BC%SFEX
    Flow_BC%SFEXIT = Flow_BC%SFEX
    Flow_BC%SFQP = Flow_BC%SFQ
    Flow_BC%IFATMP = Flow_BC%IFATM
    Flow_BC%ATMOLD = Flow_BC%ATMACT
C
    Out%PSITIMEP = Out%PSINEW
    Out%PTIMEP = Out%PNEW
C
C hydrograph calculations
C and output of solution
       ...
C transport eqn.:  update arrays before going to the next time step
C
    Out%CTIMEP = Out%CNEW
    Out%COLD = Out%CNEW
(.   .   .  to be continued on the next pages .   .   .)
```

```
C this file:  flowtransolve.f (3rd part)
(.   .   .  continuing from the previous pages .   .   .)
C
C time step processing, and preparation for the next time step
C
      CALL DTSTAT(Par%TIME,Par%DELTAT,Par%DTBIG,Par%TBIG,Par%DTSMAL,
     >                     Par%TSMAL,Par%DTAVG)
C
C If we have achieved convergence we prepare for the next time step
C unless we've reached TMAX or we're running a steady state problem
C
      IF((ABS(Par%TIME-Par%TMAX).LE.0.001*Par%DELTAT).OR.(Par%DELTAT.GE.1.0D+10))THEN
         Par%coupled_exit_flag = .TRUE.
         RETURN
      ENDIF
      CALL TIMCPL(Par%NSTEP,Par%NITERT,Par%NITERTC,Par%ITER,Par%ITERC,Par%ICNVRC
     >      Par%TIME,Par%TIMEP,Par%DELTAT,Par%DELTATP,Par%DTMIN,Par%DTMAX,Par%TMAX,
     >      Par%DTMAGM,Par%DTMAGA,Par%DTREDM,Par%DTREDS,Dim%NPRT,IO%KPRT,IO%TIMPRT)
C
C input and interpolate boundary conditions for the next time level
C
         CALL BOUNDNEXT (Dim,Par,IO,Grid,Flow_BC,MS_prop,Out)
C
C flow eqn.: update a second block of arrays before going to the next time step
C
         Out Out%POLD = Out%PNEW ! potential head
         Out%PTNEW = Par%TETAF*Out%PNEW + (1.0 - Par%TETAF)*Out%PTIMEP
         Out%PTOLD = Out%PTNEW
C
         RETURN
         END
```

3.2.4 Level 3: SUBROUTINE PICCPL

Following the path of Picard linearization of the transport equation we found SUBROUTINE PICCPL [23] which is shown below:

[23] SUBROUTINE NEWTCPL, which is not discussed here, uses partial Newton linearization for the transport equation and has the same structure of SUBROUTINE PICCPL.

```
C this file:  piccpl.f (1rst part)
C---------------------------------------------------------------
      SUBROUTINE PICCPL (Dim,Par,IO,CPU,Grid, Flow_BC,Transp_BC,MS_prop,
     >                  MBal,Sys,Out)
      ...
C start the timer
      CALL TIM(TIMNL,1)
C
C------------------------- START COUPLED NONLINEAR ITERATION -----
      200 CONTINUE
C
C assemble and solve the nonlinear system of flow for
C pressure heads (psi) and then compute potential heads (h = z + psi)
C

      CALL UNSATFLW (Dim,Par,IO,CPU,Grid,Flow_BC,MS_prop,MBal,Sys,Out)

C
C flow eqn.:  handle the back-stepping and no back-stepping case
C
      IF (Par%ICNVRF.LE.-2) RETURN
C
C calculate the weighted potential head and concentration values
C
      Out%PTNEW = Par%TETAC*Out%PNEW + (1.0-Par%TETAC)*Out%PTIMEP
      Out%CTNEW = Par%TETAC*Out%CNEW + (1.0-Par%TETAC)*Out%CTIMEP
C
C calculate velocities on the tetrahedra
C
      CALL NODELT (Dim%NT,Grid%TETRA,MS_prop%CKRW,MS_prop%CKRWE)
      CALL VELUNSAT_H (Dim,Par,Grid,MS_prop,Sys,Out)
C
C assemble and solve the unsymmetric linear system of the transport equation.
C

      CALL UNSATTRN (Dim,Par,IO,CPU,Grid,Flow_BC,Transp_BC,MS_prop,
     >                  MBal,Sys,Out)

C
(.   .   .  to be continued on the next pages .   .   .)
```

The PICCPL routine manages the solution of the flow (SUBROUTINE UNSATFLW) and transport (SUBROUTINE UNSATTRN) variably saturated nonlinear equations and then executes a convergence check (SUBROUTINE CNVRGC) of the successive substitution (SS) scheme to solve the coupled system (equation (18)) of flow and transport in porous media.

```
C this file:  piccpl.f (2nd part)
(.   .   .  continuing from the previous pages .   .   .)
C check for convergence of the nonlinear iteration
C
       CALL CNVRGC(Dim%N,Par%ICNVRC,Par%ITERC,Par%ITMXC1,Par%ITMXC2,Dim%ITMXC,
     >      Out%IKMAXVC,Par%TOLNLC,Par%DELTAT,Par%DTMIN,Out%PNEW,Out%POLD,
     >      Out%CNEW,Out%COLD,Out%PMAXNV,Out%PMAXOV,Out%CMAXNV,Out%CMAXOV,
     >      Out%DIFFPV,Out%DIFFCV,Out%DIFPMX,Out%DIFCMX,Out%DIFMAX)
C
C update variables of nonlinear iterative scheme and output convergence results
C
       CALL NLUPD(IO,Dim%N,Par%ITER,Par%ITERC,Par%ICNVRC,Par%IRELAXC,Par%NITER,
     >      Par%NITERT,Par%NITERC,Par%NITERTC,Par%ITTOTC,Par%KBACK,Par%NSTEP,
     >      Par%OMEGAP,Par%OMEGAPC,Par%TIME,Par%TIMEP,Par%DELTAT,Par%DTMIN,
     >      Par%DTREDM,Par%DTREDS,Out%IKMAXVC,Out%PMAXNV,Out%PMAXOV,Out%DIFFPV,
     >      Out%CMAXNV,Out%CMAXOV,Out%DIFFCV,Out%DIFPMX,Out%DIFCMX,Out%CNEW,
     >      Out%COLD,Out%CTIMEP,Out%PNEW,Out%POLD,Out%PTIMEP)
C
       IF (Par%ICNVRC .EQ. -1 .OR. Par%ICNVRC .EQ. -2) GO TO 200
C-------------------------------- END NONLINEAR ITERATION -------------
C stop the timer
       CALL TIM(TIMNL,2)
       CPU%NLC=CPU%NLC+TIMNL
C
       RETURN
       END
```

The convergence of the SS scheme is controlled by the the following code fragment taken from file `cnvrgc.f`:

```
       ...
       IF (DIFMAX .LE. TOLCPL) THEN ! << convergence achieved >>
          IF (ITERC .LT. ITMXC1) THEN
             ICNVRC=1 !  increase DELTAT
          ELSE IF (ITERC .LT. ITMXC2) THEN
             ICNVRC=2 !  same DELTAT
          ELSE
             ICNVRC=3 !  decrease DELTAT
          END IF
       ELSE                    !  << convergence failed>>
          IF (ITERC .LT. ITMXC) THEN
             ICNVRC=-1 !  iterate again
          ELSE IF (DELTAT .GT. DTMIN) THEN
             ICNVRC=-2 !  back-stepping
          ELSE
             ICNVRC=-3 !  back-stepping not possible anymore
          END IF
       END IF
```

To achieve coupled system convergence DIFMAX $\leq$ TOLCPL, with TOLCPL the prescribed exit tolerance, usually in the range: $0.01 \div 0.025$. The real parameter DIFMAX, which controls convergence, is essentially given by the maximum value among current and previous nonlinear step differences of normalized pressure heads and concentrations, i.e.:

$$\mathtt{DIFMAX} = \mathtt{MAX}(\mathtt{DIFPSI}; \mathtt{DIFC})$$

where DIFPSI = |(PNEW(:)-POLD(:))/PMAX|, with PMAX = MAX (|PNEW(:)|; |POLD(:)|) and DIFC = |(CNEW(:)-COLD(:)|.

A heuristic method is adopted to determine the length Par%DELTAT of the next time step according to the convergence behavior of the current linearization step of the coupled system (18). The method increases DELTAT when the number of iterations Par%ITERC is smaller of the integer input parameter Par%ITMXC1 and decreases it when ITERC is greater than input integer parameter Par%ITMXC2. The DELTAT is maintained the same when Par%ITMXC1 < Par%ITERC < Par%ITMXC2. The DELTAT is increased according to the formula: DELTAT = DELTAT*DTMAGM + DTMAGA and decreased: DELTAT = DELTAT*DTREDM - DTREDS where the input real constants DTMAGM, DTREDM, DTMAGA and DTREDS represent the amplification and reduction factors and the additive and subtractive terms, respectively. Observe also that at convergence failure it is still possible to continue the run, decreasing current DELTAT and moving back to the previous time (*backstepping*), if the reduced DELTAT is still greater than the input constant Par%DTMIN (minimum allowable time step).

In CODESA-3D, as can be seen above from the source code of SUBROUTINE PICCPL, CPU timings are registered with the SUBROUTINE TIM, whose source code is shown below:

```
C this file:  tim.f
C---------------------------------------------------------------
       SUBROUTINE TIM(TIME,ICOD)
C
C returns elapsed time in seconds from a real time clock
C count_rate is the # of processor clock counts per seconds
C
       INTEGER ::  ICOD,ITIME
       INTEGER ::  COUNT, COUNT_RATE, COUNT_MAX
       REAL ::  TIME
C
       IF (ICOD.EQ.1) THEN ! initial setting
                   CALL SYSTEM_CLOCK(COUNT,COUNT_RATE,COUNT_MAX)
                   TIME = FLOAT(COUNT/COUNT_RATE)
       ELSE              !  periodic setting
                   CALL SYSTEM_CLOCK(COUNT,COUNT_RATE,COUNT_MAX)
                   TIME = FLOAT(COUNT/COUNT_RATE) - TIME
       ENDIF
C
       RETURN
       END
```

The SUBROUTINE SYSTEM_CLOCK is a Fortran 90 intrinsic (primitive) function.

3.2.5 Level 4: SUBROUTINES UNSATFLW, UNSATTRN and further levels

At the fourth code structure level we found two principal routines: `SUBROUTINE UNSATFLW` and `SUBROUTINE UNSATTRN`, which assemble and solve the nonlinear flow and transport equations, respectively.

The solution of flow equation.

Source code for `SUBROUTINE UNSATFLW` is shown below:

```
C this file:  unsatflw.f
C---------------------------------------------------------------
       SUBROUTINE UNSATFLW (Dim,Par,IO,CPU,Grid,Flow_BC,Transp_BC,MS_prop,
      >                  MBal,Sys,Out)
       ...
C--------------------------------START NONLINEAR ITERATION--------------
       200 CONTINUE
C
       IF (Par%IFLOW .EQ. 1) THEN
            CALL PICFLW (Dim,Par,IO,CPU,Grid,Flow_BC,MS_prop,Sys,Out)
       ELSE
          CALL NEWTFLW (Dim,Par,IO,CPU,Grid,Flow_BC,MS_prop,Sys,Out)
       END IF
C
C before checking for convergence of the nonlinear scheme, we:
C (i) back-calculate fluxes at Dirichlet nodes
C (ii) compute (but don't output) mass balance errors
C (iii) apply nonlinear relaxation (if required)
C (iv) calculate the nonlinear convergence and residual error norms
C (v) check for switching of atmospheric boundary conditions
C (vi) calculate new position of the exit point along each seepage
C face and check for seepage face exit point convergence
       ...
C check for convergence of the flow nonlinear scheme
C
       CALL CNVRGF (Dim,Par,IO,CPU,Grid,Flow_BC,MS_prop,Out)
C
       IF(Par%ICNVRF.EQ.-1) GO TO 200
C----------------------------------------------GO TO THE NEXT NONLIN STEP
       RETURN
       END
```

The convergence check for the nonlinear flow equation is done in `SUBROUTINE CNVRGF`. The principal criterium for convergence is $||\cdot|| \leq$ `TOLNL`, where `TOLNL` is the nonlinear flow iteration scheme exit tolerance and $||\cdot||$ is a vector norm (real variables `PL2` and `PINF`) of the pressure head differences between current and previous nonlinear steps. The norm can be Euclidean: `PL2=SQRT(SUM(PNEW(:)-POLD(:)))` or infinite: `PINF=MAX(PNEW(:)-POLD(:))`, according to the value (1;0) of integer input parameter `Par%L2NORM`.

At a lower code structure level there is `SUBROUTINE PICFLW` [24] that follows:

```
C this file:  picflw.f (1rst part)
C------------------------------------------------------------
       SUBROUTINE PICFLW (Dim,Par,IO,CPU,Grid,Flow_BC,Transp_BC,MS_prop,
      >              MBal,Sys,Out)
       ...
C initializations
       ...
C assemble global stiffness and mass matrices
C
       IF(Par%OSC_FLAG)THEN
C Galerkin scheme modified by Orthogonal Subdomain Collocation method
          CALL ASSPIC_OSC(Grid%x,Grid%y,Grid%z,Dim%NT,Dim%NTRI,
         >      Dim%NSTR,Grid%TETRA,Sys%TETJA,Sys%LMASS,Sys%COEF1,Sys%COEF2,
         >      MS_prop%PERMX,MS_prop%PERMY,MS_prop%PERMZ,MS_prop%CKRWE,
         >      MS_prop%ETAE,Grid%VOLU,Grid%VOLUR,Sys%BI,Sys%CI,Sys%DI)
       ELSE
C standard Galerkin scheme
          CALL ASSPIC(Dim%NT,Dim%NTRI,Dim%NSTR,Grid%TETRA,Sys%TETJA,
         >      Sys%LMASS,Sys%COEF1,Sys%COEF2,
         >      MS_prop%PERMX,MS_prop%PERMY,MS_prop%PERMZ,MS_prop%CKRWE,
         >      MS_prop%ETAE,Grid%VOLU,Grid%VOLUR,Sys%BI,Sys%CI,Sys%DI)

       ENDIF
C assemble the RHS vector, without contribution of the unsaturated zone
C gravitational term, the concentration term, and the boundary conditions.
C
       CALL RHSPIC(Dim%N,Sys%JA,Sys%TOPOL,Par%DELTAT,Out%PSITNEW,Out%PSINEW,
      >      Out%PSITIMEP,Sys%COEF1,Sys%COEF2,Sys%TNOTI)
C assemble the global LHS system matrix from the stiffness and mass matrices
C
       CALL CFMATP(Dim%NTERM,Par%TETAF,Par%DELTAT,Sys%COEF1,Sys%COEF2)
C add to the RHS contributions of:  the gravitational and concentration term
C
       IF(Par%ITRANS.ne.0)THEN
          CALL RHSGRVCON (Par,Dim%NT,Dim%NTRI3,Grid%TETRA,Grid%IVOL,
         >         MS_prop%EPSLON, Par%DELTAT,Par%TETAF,
         >         Out%CNEW,Out%CTIMEP,Dim%NSTR,
         >         MS_prop%PERMZ,MS_prop%POROS,Grid%VOLU,
         >         Sys%DI,MS_prop%CKRWE,MS_prop%SWE,Sys%TNOTI)
       ELSE
          CALL RHSGRV(Dim%NT,Dim%NTRI3,Dim%NSTR,Grid%TETRA,Sys%TNOTI,
         >         Sys%DI,MS_prop%PERMZ,Grid%IVOL,MS_prop%CKRWE)
       ENDIF
(.  .  .  to be continued on the next pages .  .  .)
```

[24] `SUBROUTINE NEWTFLW`, which is not discussed here, has the same structure.

The **initialization phase** for `SUBROUTINE PICFLW` includes: calculation of soil moisture characteristics needed for Picard scheme and setting to zero of the global stiffness (`Sys%COEF1`) and mass (`Sys%COEF2`) matrices.

```
C this file:  picflw.f (2nd part)
(.  .  .  continuing from the previous pages .  .  .)
C save a copy of RHS vector before imposing Dirichlet BC's
C
      Sys%XT5 = Sys%TNOTI
C
C impose BC's and save diagonal original coeff.  of LHS system matrix
C corresponding to Dirichlet nodes
C
      CALL BCPIC(Dim%NP,Dim%NQ,Flow_BC%CONTP,Flow_BC%CONTQ,Sys%LHSP,Flow_BC%Q,
     >      Dim%NNOD,Sys%LHSATM,Flow_BC%IFATM,Flow_BC%ATMACT,Flow_BC%ATMOLD,
     >      Sys%TOPOL,Sys%COEF1,Sys%TNOTI,Par%TETAF,Par%RMAX,Flow_BC%NUMDIR,
     >      Flow_BC%NODDIR,Dim%NSF,Flow_BC%NSFNUM,Flow_BC%NSFNOD,Flow_BC%SFEX,Sys%LHSSF)
C
C solve the linear system of equations and calculate the residual
C
      CALL SYMSLV(IO%OUT1,Dim%N,Dim%NTERM,Flow_BC%NUMDIR,Par%NITER,Par%ITMXCGSY,
     >      Par%TOLCGSY,Par%RMIN,Flow_BC%NODDIR,Sys%JA,Sys%TOPOL,
     >      Out%PSIDIFF,Sys%TNOTI,Sys%COEF1,Sys%COEF2,Sys%SCR1)
      Par%ITLIN = Par%ITLIN + Par%NITER
      Par%NITERT = Par%NITERT + Par%NITER
C
C set flag if the linear solver failed
      ...
C extract pressure heads PSINEW from the difference solution PDIFF
C
      Out%PSINEW = Out%PSINEW + Out%PSIDIFF
C
C restore diagonal elements of LHS system matrix corresponding to
C Dirichlet nodes.
C
      CALL SHLPIC(Dim%NP,Sys%TOPOL,Flow_BC%CONTP,Flow_BC%PRESC,Sys%LHSP,
     >     Sys%COEF1,Out%PSINEW,Out%PSIOLD,Dim%NSF,Flow_BC%NSFNUM,Flow_BC%NSFNOD,
     >      Flow_BC%SFEX,Sys%LHSSF,Dim%NNOD,Flow_BC%IFATM,Sys%LHSATM)
C
C calculate the maximum norm of the pressure head difference
C between the current and previous nonlinear iterations.
C
      CALL MAXNORM(Dim,Par,Out)
C
      RETURN
      END
```

Boundary conditions (BC's) are incorporated into the global discretized linear system. Neumann (or Cauchy) fluxes are added to the RHS of the linear system, while the Dirichlet boundary conditions are applied both to the LHS and the RHS of the same algebraic system. Setting of a Dirichlet boundary condition for mesh node I (`SUBROUTINE BCPIC`) is done multiplying the diagonal I-th element of LHS matrix by a very large number (usually a function of the largest real number representable by the computer $\equiv$ `HUGE(1.0)`) and setting the corresponding I-th coefficient of RHS vector to the prescribed value of the unknown. In this way we practically transform the I-th equation in the identity: `unknown = assigned value`. Before doing this we must store the diagonal coefficient of LHS matrix (`SUBROUTINE BCPIC`) in order to restore them after the solution of the system (`SUBROUTINE SHKPIC`). The same procedure is adopted for the solution of the transport equation (see `SUBROUTINE UNSATTRN` on next pages).

The solution of the linear SPD system resulting from the Picard linearization of the flow equation is solved with the CG preconditioned with IC decomposition (`SUBROUTINE SYMSLV`).

The assembling of global stiffness and mass matrices that constitute the LHS of the flow linearized system is done in `SUBROUTINE ASSPIC` [25], whose source code is shown below. The only difference between the *if branch* (flow & transport) and *else branch* (freshwater flow), which is not included for simplicity, is the presence in the *if branch* of the hydraulic permeability multiplier: `mult1=` $(1+\epsilon\overline{c})/(1+\epsilon'\overline{c})$ and the overall storage multiplier: `mult2=` $(1+\epsilon\overline{c})$, which are both set to 1 in the *else branch*.

[25] `SUBROUTINE ASSPIC_OSC`, which implements a Galerkin FE scheme modified by the Orthogonal Subdomain Collocation (OSC) method, has the same structure.

```
C this file:  asspic.f
C---------------------------------------------------------------
      SUBROUTINE ASSPIC(...)
      ...
      IF(ITRANS.NE.0)THEN ! flow & transport
         DO IEL=1,NT           !  loop on finite elements
            ISTR=1+IEL/(NTRI*3)
            IR=MOD(IEL,NTRI*3)
            IF (IR .EQ. 0) ISTR=ISTR-1
            MTYPE=TETRA(5,IEL) ! hydrogeological zone
C
            CAVG = 0.0
            CMED = 0.0
            CMEDP = 0.0
            DO K = 1,4
               I = TETRA(K,IEL)
               CMED = CMED+CNEW(I)
               CMEDP = CMEDP+CTIMEP(I)
            END DO
            CMED = 0.25*CMED
            CMEDP = 0.25*CMEDP
            CAVG = TETAF*CMED + (1-TETAF)*CMEDP ! averaged c
C
            mult1 = (1+EPSLON*CAVG)/(1+EPSLON1*CAVG) ! permeability multiplier
            mult2 = (1+EPSLON*CAVG)       !  storage multiplier
C
            VK=VOLUR(IEL)*CKRWE(IEL)
            VE=VOLU(IEL)*mult2*ETAE(IEL)
            PVK1=mult1*PERMX(ISTR,MTYPE)*VK
            PVK2=mult1*PERMY(ISTR,MTYPE)*VK
            PVK3=mult1*PERMZ(ISTR,MTYPE)*VK
            DO K=1,4
               PVKB=PVK1*BI(K,IEL)
               PVKC=PVK2*CI(K,IEL)
               PVKD=PVK3*DI(K,IEL)
               DO L=K,4
                  IND=TETJA(K,L,IEL)
                  COEF1(IND)=COEF1(IND) + PVKB*BI(L,IEL) + stiffness matrix
     >                            PVKC*CI(L,IEL) + PVKD*DI(L,IEL)
                  COEF2(IND)=COEF2(IND) + VE*LMASS(K,L) ! mass matrix
               END DO
            END DO
         END DO           !  end of loop on finite elements
C
      ELSE                     !  freshwater flow alone
         ...
      ENDIF
C
      RETURN
      END
```

The solution of transport equation.

Source code for `SUBROUTINE UNSATTRN` is shown below:

```
C this file:  unsattrn.f
C-------------------------------------------------------------
      SUBROUTINE UNSATTRN (Dim,Par,IO,CPU,Grid,Flow_BC,Transp_BC,MS_prop,
     >                  MBal,Sys,Out)
      ...
C initializations
      ...
C assemble global stiffness and mass matrices
C
        CALL ASSTRN(Dim%NT,Dim%NTRI3,Dim%NSTR,Grid%TETRA,Par%IP4,Grid%IVOL,
       >           Sys%TETJAC,MS_prop%DIFFUS,Grid%X,Grid%Y,Grid%Z,
       >           Out%UU,Out%VV,Out%WW,Grid%VOLU,Grid%VOLUR,Sys%BI,
       >           Sys%CI,Sys%DI,Sys%COEF1C,Sys%COEF2C,MS_prop%POROS,
       >           MS_prop%SWE,MS_prop%RETARD,MS_prop%ALFAL,
       >           MS_prop%ALFAT,Sys%LMASSC)

C
C assemble RHS vector, without the contribution of the BC's
C
      CALL RHSTRN(Dim%N,Sys%TOPOLC,Sys%JAC,Par%DELTAT,Par%TETAC,
     >           Sys%TNOTIC,Out%CTIMEP,Sys%COEF1C,Sys%COEF2C)
C
C assemble the global LHS system matrix from the stiffness and mass matrices
C
      CALL CFMAT(Dim%NTERMC,Par%TETAC,Par%DELTAT,Sys%COEF1C,Sys%COEF2C)
C
C get an estimate of the boundary fluxes for Dirichlet BC's at time level zero
      ...
C save diagonal elements of LHS system matrix corresponding to Dirichlet nodes
C
      CALL LHSTRN(Dim%NPC,Transp_BC%NNPC,Sys%TOPOLC,Sys%JAC,Sys%LHSC,Sys%COEF1C)
C
C impose boundary conditions
C
      CALL BCTRN(Dim%N,Dim%NPC,Par%NSTEP,Dim%NMC,Transp_BC%NNPC,
     >           Transp_BC%NNMC,Sys%TOPOLC,Sys%JAC,Par%RMAX,Sys%XT5C,
     >           Sys%TNOTIC,Transp_BC%PC,Transp_BC%MC1,
     >           Transp_BC%MC2,Sys%COEF1C)
(.  .  .  to be continued on the next pages .  .  .)
```

```
C this file:  unsattrn.f (2nd part)
(.   .   .  continuing from the previous pages .   .   .)
C
C solve the linear system of equations and calculate the residual
C
      CALL NSYSLV(Par%ISOLVC,IO%OUT1,Dim%N,Dim%NTERMC,Dim%NPC,
     >      Transp_BC%NNPC,Par%ITMXCGNS,Dim%IBOT,Dim%MINBOT,
     >      Dim%MAXBOT,IERSYM,Par%NITERC,Sys%IAC,Sys%JAC,
     >      Sys%TOPOLC,Sys%INSYM,Par%TOLCGNS,Par%RMIN,Sys%COEF1C,
     >      Sys%COEF2C,Sys%SCR1,Sys%SCR2,Sys%RNSYM,Out%CNEW,Sys%TNOTIC)
      IF(Par%ISOLV .EQ. 3 .AND. IERSYM .NE. 0 ) THEN
         CALL CLOSIO (IO)
         STOP
      END IF
      Par%ITLINC =Par%ITLINC+Par%NITERC
      Par%NITERTC=Par%NITERTC+Par%NITERC
C
C restore diagonal elements of LHS system matrix corresponding to
C Dirichlet nodes
C
      CALL SHLNSY(Dim%NPC,Transp_BC%NNPC,Sys%TOPOLC,Sys%JAC,
     >      Out%CNEW,Transp_BC%PC,SysC
      RETURN
      END
```

The solution of the linear unsymmetric system resulting from the Picard (or partial Newton) linearization of the transport equation is solved with Krylov subspace solvers of the CG-family in the `SUBROUTINE NSSLV`. The choice of the solver is made upon the value of the integer input variable `Par%ISOLVC`:

- `ISOLV=-5 BCGSTAB (w diagonal preconditioner);`
- `=-4 BCGSTAB w/o preconditioner;`
- `=-3 TFQMR (w diagonal preconditioner);`
- `=-2 TFQMR (w/o preconditioner);`
- `=-1 TFQMR (incomplete LU preconditioner);`
- `=0 BCGSTAB (incomplete LU preconditioner);`
- `=1 minimum residuals (GRAMRB);`
- `=2 GCRK(5);`
- `=3 direct Gauss solver.`

The assembling of global stiffness and mass matrices that constitute the LHS of the transport linearized system is done in `SUBROUTINE ASSTRN`, whose source code is shown below:

```
C this file:  asstrn.f
C-----------------------------------------------------------------
       SUBROUTINE ASSTRN(...)
       ...
       DO IEL=1,NT              !  loop on finite element
          ISTR=1 + IEL/NTRI3
          IR=MOD(IEL,NTRI3)
          IF (IR .EQ. 0) ISTR=ISTR-1
          MTYPE=TETRA(5,IEL) ! hydrogeological zone
          DO I=1,4
             XX(I)=X(TETRA(I,IEL))
             YY(I)=Y(TETRA(I,IEL))
             ZZ(I)=Z(TETRA(I,IEL))
             TETRAL(I)=I
          END DO
C rotate the current element so that the local x-axis is aligned with
C the velocity vector, and re-compute the basis functions
C
          CALL ANIS3D(UU(IEL),VV(IEL),WW(IEL),XX,YY,ZZ,VEL)
          CALL BASIS(IP4,TETRAL,XX,YY,ZZ,BIL,CIL,DIL)
C
          VPR=VOLU(IEL)*POROS(ISTR,MTYPE)*SWE(IEL)*RETARD(ISTR,MTYPE) ! nS_wR_dVe
C
          DXXV=(VEL*ALFAL(ISTR,MTYPE) +
         >      DIFFUS*POROS(ISTR,MTYPE)*SWE(IEL))*VOLUR(IEL) ! D'_xx = α_L|v|+D0nS_w
          DYYV=(VEL*ALFAT(ISTR,MTYPE) +
         >      DIFFUS*POROS(ISTR,MTYPE)*SWE(IEL))*VOLUR(IEL) ! D'_yy = α_T|v|+D0nS_w
          DZZV=DYYV ! D'_yy=D'_zz
C
          DO K=1,4
             KNOD=TETRA(K,IEL)
             DXXVB=DXXV*BIL(K)
             DYYVC=DYYV*CIL(K)
             DZZVD=DZZV*DIL(K)
             DO L=1,4
                LNOD=TETRA(L,IEL)
                IND=TETJAC(K,L,IEL)
                IF (LNOD .GE. KNOD) THEN
                      COEF1C(IND)=COEF1C(IND) + DXXVB*BIL(L) +
                     >      DYYVC*CIL(L) + DZZVD*DIL(L)     !  stiffness matrix
                      COEF2C(IND)=COEF2C(IND) + VPR*LMASSC(K,L) ! mass matrix
                END IF
             END DO
          END DO
C
(.  .  .  to be continued on the next pages .  .  .)
```

In the code fragment above the input real variable MS_prop%Rd is the *retardation factor* which is set to one.

```
C this file:  asstrn.f (2nd part)
(.  .  .  continuing from the previous pages .  .  .)
C symmetrize matrix
C
          DO K=1,4
             KNOD=TETRA(K,IEL)
             DO L=1,4
                LNOD=TETRA(L,IEL)
                IF (LNOD .LT. KNOD) THEN
                   IND=TETJAC(K,L,IEL)
                   INDS=TETJAC(L,K,IEL)
                   COEF1C(IND)=COEF1C(INDS)
                   COEF2C(IND)=COEF2C(INDS)
                END IF
             END DO
          END DO
       END DO                 !  end of loop on finite elements
C
C assemble unsymmetric advection term (matrix B) contribution to the stiffness matrix
C
       DO IEL=1,NT              !  loop on finite element
          VSIGN4=IVOL(IEL)*0.25
          DO L=1,4
             ADVCTN(L)=VSIGN4*(UU(IEL)*BI(L,IEL) +
     >             VV(IEL)*CI(L,IEL) + WW(IEL)*DI(L,IEL))
          END DO
          DO K=1,4
             DO L=1,4
                IND=TETJAC(K,L,IEL)
                COEF1C(IND)=COEF1C(IND) + ADVCTN(L)
             END DO
          END DO
       END DO                 !  end of loop on finite elements
C
       RETURN
       END
```

At lower code structure levels we found routines that were already in the FLOW3D and SATC3D source distributions, so to describe these source file we simply refer to related manuals and publications [11, 13] and [14].

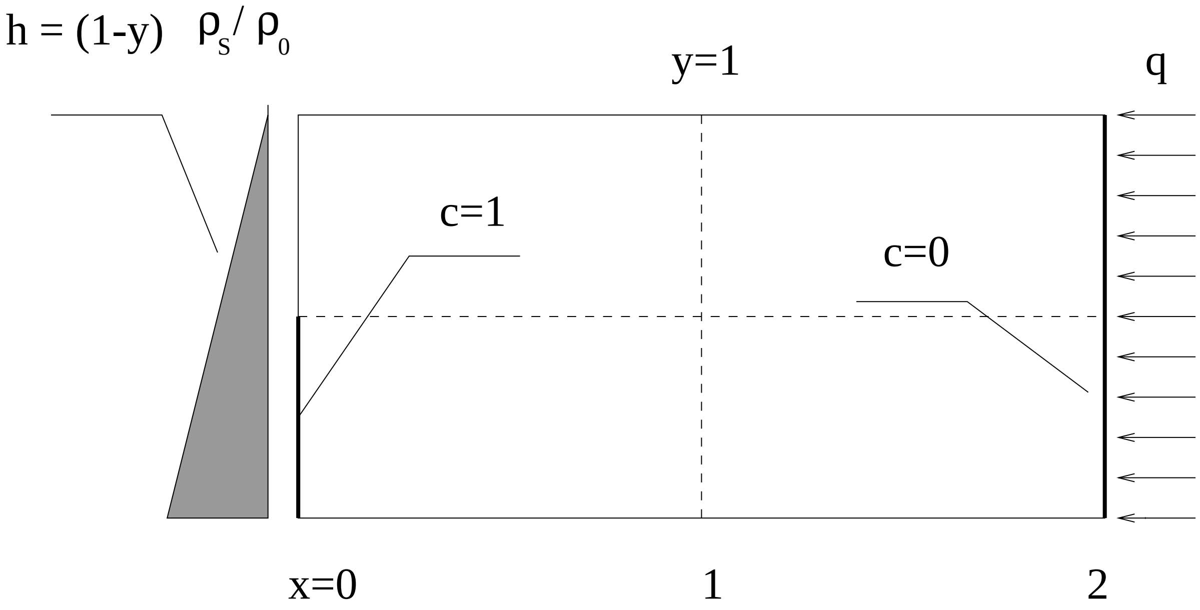

Figure 6: *Definition of the Henry problem and boundary conditions.*

4 Benchmark and applications

In general exact solutions of density-dependent coupled flow and transport problems are not available due to the nonlinear nature of these processes, excepting a semi-analytical solution by Henry of a steady state case [16]. In this chapter the Henry problem is developed as a benchmark to test reliability of the CODESA-3D code. Other tests have been undertaken in order to verify model response in presence of strongly coupled flow and transport problems [4] Two other examples are added to demonstrate different applications of the model; the first example shows the capability to mimic a seepage face in a partially saturated aquifer [15] and the second one simulates the seawater intrusion in a coastal phreatic aquifer (Korba region, Tunisia) [21, 22]. Other CODESA-3D applications are described [24, 27].

4.1 Benchmark problem: the Henry problem

The Henry problem describes the advance of a saltwater front in a confined aquifer which was initially saturated with uncontaminated freshwater. The geometry and the boundary conditions of the Henry problem are shown in Figure 6.

Problem definition. The computational domain, homogeneous and isotropic, is represented by a 3-D parallelepiped box of 0.1 m thickness, 1 m depth and 2 m length. The flow boundary conditions consist of impermeable borders along the top and the bottom of the computational box. These side walls are also impervious for diffusive solute fluxes (i.e. $\partial c/\partial n = 0$). Hydrostatic pressure is assumed along the vertical boundary of the right side corresponding to the sea side. The aquifer is charged with freshwater at a constant flux from the left side. At the inland side, the concentration is zero (freshwater condition), while at the coastal side the relative concentration of seawater is imposed for an height of 0.5 m from the aquifer bottom. Initial conditions are zero hydraulic heads and zero concentrations throughout the domain. The simulation parameters for the Henry problem are given in Table 4.1.

Quantity	Value	Unit
D_o	6.6×10^{-6}	m^2 s^{-1}
g	9.81	ms^{-2}
k	1.019368×10^{-9}	m^2
q	6.6×10^{-2}	kg m^{-1} s^{-1}
ϕ	0.35	/
μ	10^{-3}	$kgm^{-1}s^{-1}$
ρ_0, ρ_s	$(1.,1.025) \times 10^3$	kg m^{-3}

Table 1: Simulation parameters of the Henry problem.

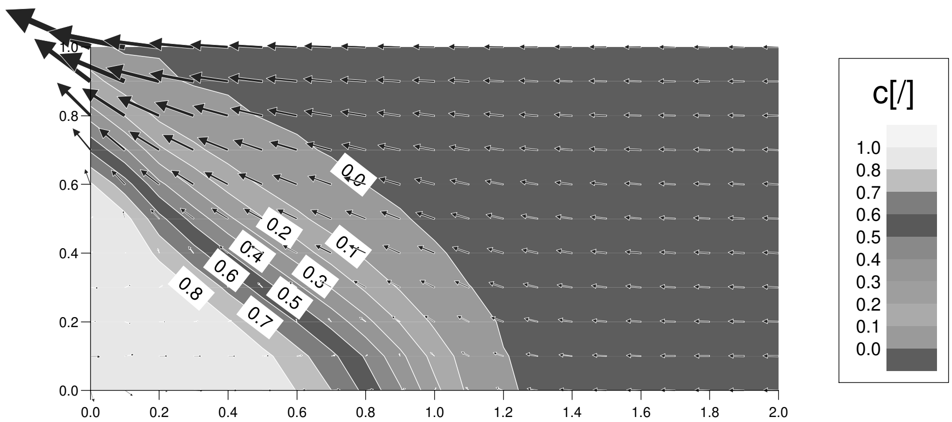

Figure 7: *Relative concentration isocontours and velocity field at steady state for the Henry problem.*

Results and discussion. Figure 7 shows CODESA-3D computed relative concentration isolines at steady state. Seawater intrudes into the model domain through the lower left boundary. In this area, where the density is highest, pressure gradients are oriented almost vertically upward. In contrast the gravitational force is directed vertically downward. These two driving forces causes the observed concentration pattern into the aquifer domain. As density differences decrease along the aquifer bottom from theleft to the right hand side, the potential-driven flow forced by the freshwater influx becomes more important and allows less dense fluid leaving the domain through the upper left boundary section. Figure 7 compare very well with the analogous numerical results of numerical models [9, 20] of the Henry problem. Appendix C report the input dataset for CODESA-3D of the Henry problem.

Computational issues. The computational grid is a regularly spaced (with Δ = 0.10 m) parallelepiped box of dimensions $2 \times 1 \times 1$ m. It is composed by 2 541 nodes and 12 000 tetrahedra. The steady state was reached within 1 time step only, using a very large `DELTAT` (10^{20} seconds $\to \infty$) such that time derivatives of system (4) practically tend to zero. The *SS* linearization procedure of the coupled system (with Picard linearization of the transport equation) required 31 iterations with a exit tolerance of 1E-2. The flow equation, linearized also with the Picard scheme in 62 iterations (exit tolerance of 1.E-4), was solved using the

CG scheme with diagonal preconditioning. The average number of linear solver iterations per flow linearization step was 107.42 for a total of 6660 preconditioned CG iterations. The transport equation was solved using the TFQMR scheme with **LU** preconditioning. On the average the solution of the transport equation needed 14 iterations per linearization step of the coupled system for a total of 434 iterations. The CPU time for the simulation on a Silicon Graphics (SGI) computer equipped with RISC10000 processor, 512 Mbyte of RAM and running IRIX/6.4 operating system (*filuferru.crs4.it*) was of 45 seconds of which 43 seconds for the solution of the coupled system. Of this time about the 50% was taken in the assembling and solution of the flow equation and about 30% in the assembling and solution of the transport equation.

4.2 Applications

4.2.1 Contamination of a ditch-drained aquifer by trickle infiltration from a salt dome

The example is a three-dimensional problem of variably saturated flow and transport in a ditch-drained aquifer with incident steady rainfall and trickle infiltration of a salt contaminant.

Problem definition. The computational domain, homogeneous and isotropic, is represented by a 3-D parallelepiped box of x m thickness, y m depth and z m length. A Darcy flux of 0.15 cm/d is applied over the upper square central area, while the rest of the surface is subjected to a Darcy flux of 0.1 cm/d. A fixed seepage face, of 10 cm height from the bottom of the box, boundary condition is applied along the front vertical face. Boundary conditions of zero Darcy flux are imposed along the other three vertical faces, and at the base of the domain. The aquifer system is isotropic and homogeneous, with a saturated hydraulic conductivity of 40 cm/d and a porosity of 0.3. We assume that the aquifer is initially free of contaminant (zero concentration), and that the contaminant enters the aquifer from the salt dome represented by the trickle infiltration area. Along all the other boundaries of the aquifer conditions of zero dispersive flux are imposed. The density ratio is $\epsilon = 0.03$ while the viscosity ratio and the diffusion coefficient D_o are set to zero; the values of the dispersivity coefficients are $\alpha_L = 2$ cm, and $\alpha_T = 0.4$ cm. Huyakorn equations [7] describe the soil hydraulic properties. Water saturation is:

$$S_w(\psi) = (1 - S_{wr})S_e + S_{wr} \tag{19}$$

where $S_{wr} = 0.001$ is the residual water saturation and S_e is the effective water saturation:

$$\begin{array}{lll} S_e(\psi) & = [1 + k^\beta(\psi_a - \psi)^\beta]^{-\gamma}; & \psi < \psi_a \\ S_e(\psi) & = 1 & \psi \geq \psi_a \end{array} \tag{20}$$

with $\psi_a = -10.0$ cm the air entry pressure, $k = 0.015$, $\beta = 2.0$ and $\gamma = 3.0$. Relative permeability is:

$$k_r(\psi) = 10^G \tag{21}$$

where $G = aS_e^2 + (b - 2a)S_e + a - b$, $a = 2.0$ and $b = 3.5$.

Results and discussion. The pressure head contours and the velocity and concentration fields along cross section AA′ after 60 days are shown in Figures 8 and 9. The results show how the aquifer drains at a rate faster than the recharge form the surface, generating unsaturated conditions in the upper portion of the aquifer. The water table surface is represented by the

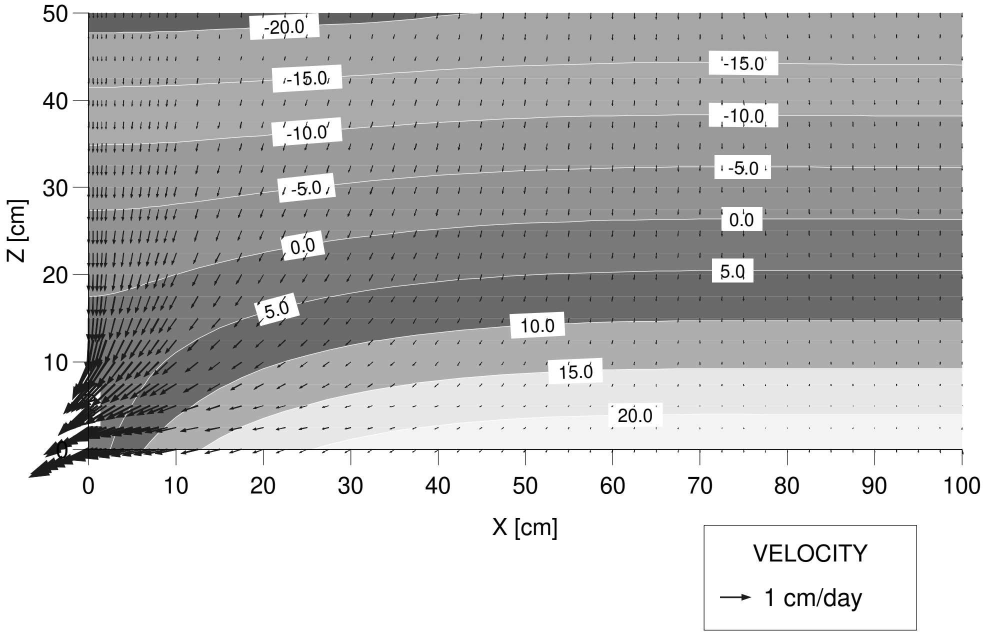

Figure 8: *Pressure head and velocity field after 60 days along cross section A′-A′ for Example 3.*

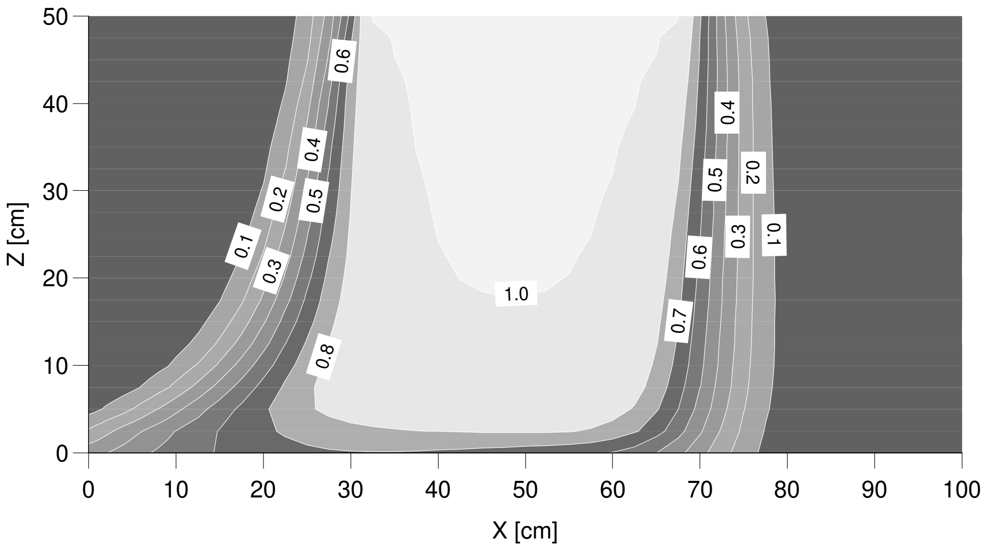

Figure 9: *Salt concentration fields after 60 days along cross section A′-A′ for Example 3.*

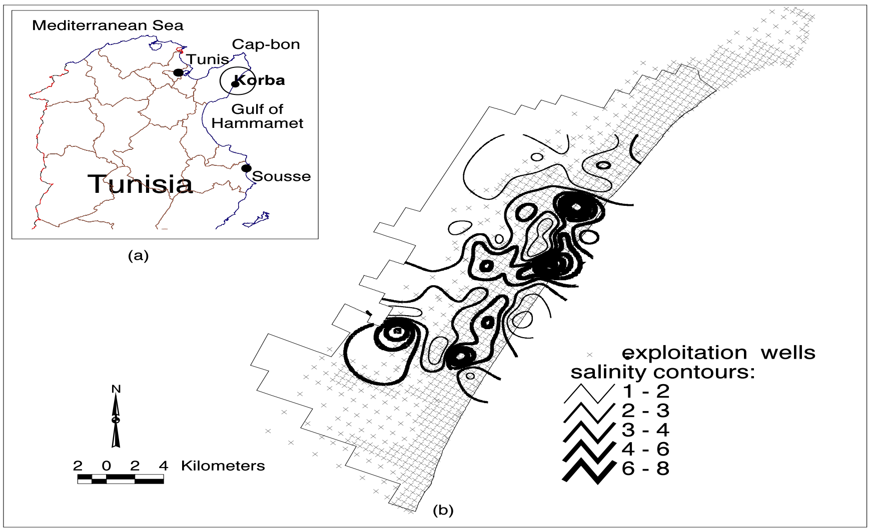

Figure 10: *a) Geographic location of the Korba coastal aquifer; b) exploitation map and salinity contours [g/l] from 1996 measurements.*

contour line at zero pressure, which intersect the front vertical face on the left at about 20 cm height from the aquifer bottom. The salt plume is roughly symmetric near the surface but turn towards the seepage face in the lower aquifer where the velocities are higher.

Computational issues. The domain contains 2 009 nodes and 3 840 triangles at the surface, and is discretized into 20 vertical layers, to yield 42 189 nodes and 230 400 tetrahedra for the 3-D grid.

4.2.2 The saltwater intrusion problem of the Korba coastal aquifer

The example is a three-dimensional problem of seawater intrusion in a coastal phreatic aquifer. The Korba aquifer has been studied within the Avicenne project [33], in collaboration with the Institute National Agronomique de Tunisie (INAT).

Problem definition. The 438 km^2 Korba aquifer is a part of the western coastal aquifer of the Cap Bon area, which extends from the city of Ras Maamoura in the south to the city of Kelibia in the North, and is bounded by the Mediterranean Sea in the east and the Djebel Abderrahman anticline in the west (Figure 10). The phreatic alluvial aquifer consists of two important formations: a Pliocene sandstone whose stratigraphic series correspond to an alternation of sandstone and marl, and a Quaternary alluvium containing detrital sediment (sand, gravel, silt) with thin clay lenses. The aquifer depth is in reality highly variable, ranging from 150 m in the south to 30 m in the north, and decreasing from east to west to nearly vanish at the Djebel Abderrahman anticline. Recharge of this aquifer is mainly from infiltration from natural replenishment at an average rate of 32 mm/year [6], or about 7% of the mean annual rainfall (460 mm).

Model setup. The finite element surface mesh used for the numerical simulation of the Korba aquifer contains 1 643 nodes and 2 917 Delauney triangles and follows the digital elevation model of the Korba plain with elevations above see level (a.s.l.) in the range 0 to 160 m [19, 21]. The same 2-D grid was adopted to cover the aquifer bottom having depths, with reference to the topographic surface, in the range 24 to 150 m. The 3-D grid contains 7 layers of varying depth for a total of 11 501 nodes and 52 506 tetrahedra. In accordance with the geology of the Korba plain, 2 material subregions have been defined with the corresponding saturated conductivities given in Table 4.2.2. Of the two formations, the Pliocene sandstone constitutes the main part of the aquifer domain, while the Quaternary alluvium is sparser and tinner, thus only the upper two layers of the 3-D grid were set to the corresponding hydraulic permeability. The aquifer is assumed isotropic and homogeneous within each subregion. For

Hydrological zone	Variable	Value
Pliocene sandstone	$K_x = K_y = K_z$	3.4×10^{-6}
Quaternary alluvium	$K_x = K_y = K_z$	3.4×10^{-5}

Table 2: Isotropic saturated hydraulic condictivities K_s [m/s] of the Korba case study.

the entire aquifer the other material and solute properties for the simulations are set to the following values: n = 0.25, S_s = 0.0012 m^{-1}, ϵ= 0.025, α_L = 170 m, α_T = 7 m, D_o = 0 m^2/s. *Boundary conditions*. The southern, western, and northern boundaries and the aquifer bottom are treated as impermeable to flow and not allowing mass dispersive flux. On the eastern (coastal) boundary, the flow equation is fixed real head $h^r = C$, which, expressed in terms of equivalent freshwater head gives a linear distribution with z: $h = C + (C - z)\epsilon$. For the transport equation a zero dispersive flux is imposed over a window of variable depth w ($w <$ 30 m) and a prescribed seawater concentration (c=1) is imposed below w. Along the rivers Dirichlet boundary conditions are imposed as constant in time freshwater heads equal to the corresponding river bed elevation. The effect of this condition, which implies that the water table lies on the topographic surface, needs to be evaluated. The Neumann condition of zero dispersive flux along the upper portion of the coastal boundary allows the lighter freshwater to discharge into the sea through the seepage window. A constant in time infiltration rate of 32 mm/year was applied over the entire surface boundary, except for few areas (dunes near the coastal boundary), which were subjected to a doubled infiltration rate of 64 mm/year. A leakage from the lower confined aquifer was considered, imposing 121 point sources located in the western boundary of the aquifer bottom, each of 2.0$\times$ 10^{-3} m^3/s of freshwater flux, for a total recharge of 241 $\times$ 10^{-3} m^3/s (7.6 $\times$ 10^6 m3/year). Variable pumping rates at 953 clustered wells, with penetrations at two different depths of 30 and 45 m, were imposed with a maximum total extraction of 50 $\times$ 10^6 m^3/year.

Results and discussion. Two set of simulations (freshwater flow alone and coupled variably dense water flow and salt transport) were run in order to reproduce current situation for the threatened aquifer.
First set of simulations. A steady state simulation of the flow model was run, with the aquifer initially completely saturated at a uniform equivalent freshwater pressure head of zero throughout. The simulation takes into account leakage (7.6 $\times$ 10^6 m^3/year), natural recharge (6 $\times$ 10^6 m^3/year) and pumping at a rate of 37the maximum total extraction (18.50 $\times$ 10^6 m^3/year). The hydraulic conductivity values were calibrated to best reproduce the reference piezometric field (1962). Figure 11 compares the steady state simulated water table elevation above sea level (a.s.l.) with the values recorded at the observation wells in 1962, which was used as the starting point for the transient simulations described below. Although differences exist with field data, the correspondence is relatively good. In the calibration process the parameters which most heavily influenced the behavior of the system were the imposed boundary conditions, namely the prescribed head along the river beds, the infiltration rate, and the saturated hydraulic conductivity. The adopted parameters were utilized for subsequent transient flow and mass transport simulations.
Second set of simulations. A second set of simulations was run using the steady state con-

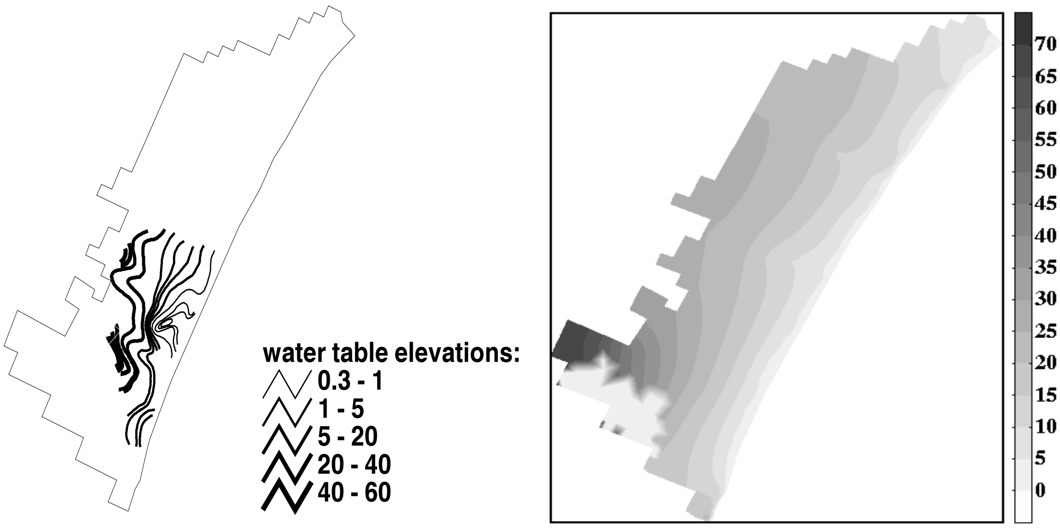

Figure 11: *Comparison between field measured and calculated water table elevations (m.a.s.l) taking into account natural recharge for the unstressed (no pumping) steady state flow simulation.*

ditions from the first set as initial conditions for the flow field. Initial conditions for the relative salt concentration were set assuming an ideal interface (not miscible) between salt- and freshwater according to the Ghyben-Herzberg approximation [3]: $h_s = h_0/\epsilon$, where h_s is the depth of a stationary interface below see level and h_0 is the phreatic level above sea level. That is, at any distance from the sea, the depth of the interface below the sea is 40 (=1/0.025) times the height of the water table above it. A period of 35 years was simulated. In this second set, pumping at variable rates was applied to the aquifer. Applying the maximum total exploitation rate for 35 years would have resulted in a severe water table lowering and at the end an almost complete drainage of the aquifer.
Scenario (a). Pumping rates linearly increasing from 18.50 to 35 $\times$ 10^6 m^3/year during the entire simulation time. Figure 12 shows the water table drawdown at the end of the period and the saltwater concentration field along the water table surface for the same period. The pumping area is evident in the figure from the flow field. The depression cones, artificially separated by the river network, have a maximum depth of -6 m, which is reasonably near to the observed field values. The results of the simulations show that water withdrawal from pumping has caused significant saltwater encroachment.
Scenario (b). Pumping rates linearly increasing from 18.50 to 50 $\times$ 10^6 m^3/year during the entire simulation time. The pumping area is evident in Figure 13 both from the flow field and from the deflection of the saltwater front towards the two major upconing zones. The depression cones have a maximum depth of -14 m, which has been observed in the Korba plain during 1996.
Scenario (c): To analyze the impact of aquifer artificial replenishment, the same period of 35 years was also simulated considering an additional hypothetical recharge of 1.3 $\times$ 10^6 m^3/year distributed in a few points belonging to the dune regions near the coast. Figure 13 shows that the saltwater concentration isolines for the run are significantly closer to the coastal boundary than in the previous cases, showing the induced effect of the freshwater discharge into the sea. With regards to the coupled flow and transport simulation in both pumping and recharge conditions, the calculated equiconcentration lines did not agree well with the measured salinity contour lines shown in Figure 10, probably due to aforementioned lack of information about soil and aquifer physical parameters and to inadequate field data representation in space and time. Two other simulations were run in order to investigate the

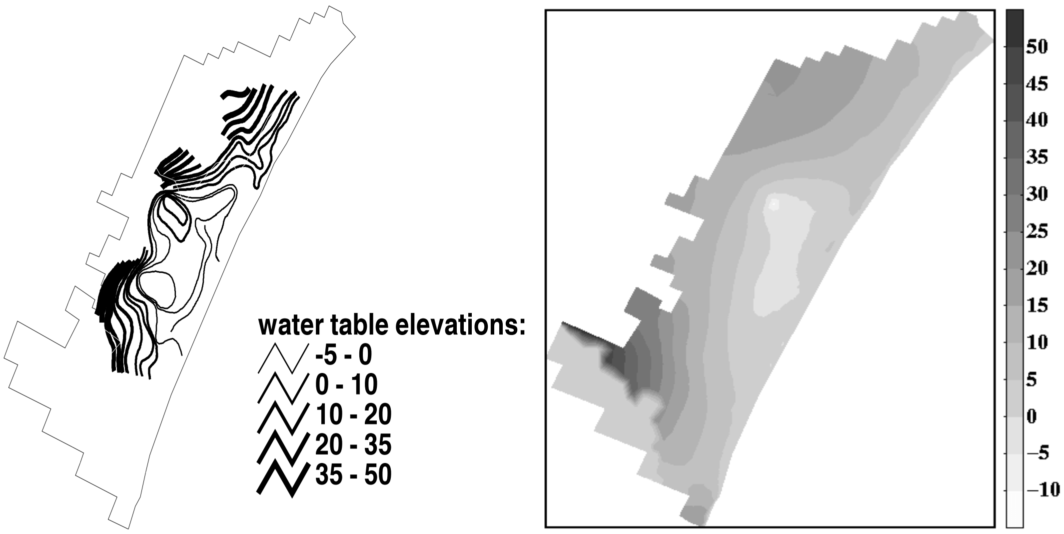

Figure 12: *Comparison between field measured and calculated water table elevations (m.a.s.l) at 24 years (1996) for the transient coupled flow and transport simulation taking into account both natural recharge and heavy pumping.*

influence of the seepage face window to sea. A window depth of 30 m was used in the first run and a depth of 5 m in the second. The relative salinity field difference between the two cases for a transient simulation of 10 years with both infiltration and heavy pumping has a mean value of 0.0015 and a standard deviation of 0.011.

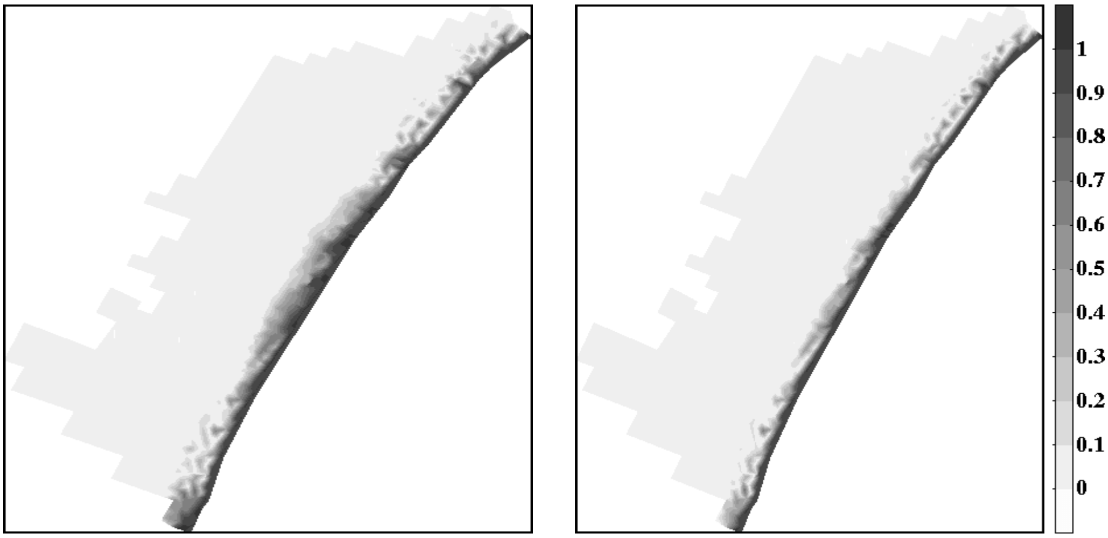

Figure 13: *Comparison between calculated relative concentration fields along the water table surface at 24 years (1996) for the transient coupled flow and transport simulation taking into account both natural recharge and heavy pumping for the transient coupled flow and transport simulation taking into account both natural recharge and heavy pumping (on the left) or half-rate pumping (on the right). Note the retreating saltwater front in the right figure in comparison to the left.*

Acknowledgement

The financial support of this work was provided by the Italian Ministry of the University, Project ISR8, C11-B and by the Sardinian Regional Authorities.

A Appendix A

The Appendix A defines the physical parameters of CODESA-3D mathematical model, pointing out which physical parameters are read in input files and written in the output files. It also gives the expression of matrices and vectors of the numerical model, as in [15]. By doing this, the CODESA-3D implemented extensions to the parent codes (SATC3D, FLOW3D) are listed and some planned extensions to build up the final model reported in [15] are also indicated.

A.1 Physical parameters

In the following are described the physical parameters of the coupled flow and transport equations of system (4). Refer to the Chapter § 2 for symbol meanings not specified here.

The parameters of the *flow equation* are

- $\mathbf{K} = k_{rw}\mathbf{K_{sat}}$: the *variably saturated* conductivity tensor $[L/T]$, with
  - $k_{rw}(\psi)$ [/] the relative permeability, that is a nonlinear function of ψ [3, 7];
  - $\mathbf{K_{sat}} = \frac{\rho g}{\mu}\mathbf{k}$ the saturated conductivity tensor $[L/T]$, with with $\mathbf{k}$ the permeability tensor $[L^2]$. Using *principal directions of the medium permeability* as the global coordinate system, we obtain a *diagonal* permeability tensor $\mathbf{k}$:

$$\mathbf{k} = \begin{bmatrix} k_{xx} & 0 & 0 \\ 0 & k_{yy} & 0 \\ 0 & 0 & k_{zz} \end{bmatrix}$$

    Incorporating constitutive equations for variable density ρ and viscosity μ, the x-coefficient of the saturated conductivity tensor $\mathbf{K_{sat}}$ becomes:

$$K_{xx} = k_{rw}\frac{(1+\epsilon c)}{(1+\epsilon' c)}\frac{\rho_o g}{\mu_o}k_{xx} = k_{rw}\frac{(1+\epsilon c)}{1+\epsilon' c}K_{xx,o}$$

    $K_{xx,o}$ the xx-coefficient of the saturated conductivity tensor at reference conditions (ρ_o, μ_o).

- $\sigma = (1+\epsilon c)\sigma^\star$ is the *overall storage* coefficient, with:
  - $\sigma^\star = (S_w S_{sat} + \phi\rho_o g\frac{\partial S_w}{\partial p})$ the overall storage coefficient for *freshwater* flow, being
    * $S_w = \phi\theta$ the water saturation [/], with ϕ the porosity and θ the moisture content, i.e., considering a representative elementary volume (REV) of porous medium θ is the ratio between the volume of water and the volume of porous medium in the REV;
    * $S_{sat} = \rho g(\alpha + \phi\beta)$ the elastic specific storage $[1/L]$, with α [26] and β soil and fluid compressibility $[T^2L/M]$, respectively. The specific storage S_{sat} is the amount

[26] Soil compressibility can be expressed in terms of mechanical parameters: $\alpha = \frac{(1+\nu)(1-2\nu)}{E(1-\nu)}$, with E Young and ν Poisson modulus.

of water per unit volume of a saturated formation stored/released owing to expansion/compression of mineral skeleton and pore fluid per unit change of hydraulic head h.

In the *transport equation* remains to be defined the dispersion tensor $\mathbf{D}$, $[L^2/T]$ whose coefficients [3, 7] $D_{ij}, i, j = x, y, z$ are given by:

- $D_{ij} = \alpha_T \|\mathbf{v}\| \delta_{ij} + (\alpha_L - \alpha_T) v_i v_j / \|\mathbf{v}\| + \phi S_w D_o \tau \delta_{ij}$: is the dispersion tensor $[L^2/T]$, with α_L and α_T the longitudinal and transverse dispersivity $[L]$, respectively, $\|\mathbf{v}\| = \sqrt{v_x^2 + v_y^2 + v_z^2}$, δ_{ij} is the Kronecher symbol, D_o is the molecular diffusion coefficient and τ is the tortuosity (it is usually set to unity). Rotating the Cartesian coordinate system at a nodal point, such that one of its axes, say x, coincides with the direction of the average uniform velocity $\mathbf{v}$ (thus defining a local coordinate system of *principal axes of dispersion* [27]) the dispersion tensor becomes diagonal:

$$\mathbf{D} = \begin{bmatrix} \alpha_L \|\mathbf{v}\| + \phi S_w D_o \tau & 0 & 0 \\ 0 & \alpha_T \|\mathbf{v}\| + \phi S_w D_o \tau & 0 \\ 0 & 0 & \alpha_T \|\mathbf{v}\| + \phi S_w D_o \tau \end{bmatrix}$$

A.1.1 Input and Output

With reference to the *flow* and *transport* equations (4), CODESA-3D input data are:

- input file `soil`
  - specification of capillarity and retention curve characteristics (e.g.: $k_{rw}, \partial S_w/\partial p, \theta$, etc.), which are highly nonlinear functions of ψ, for the given soil [28]. The choice is done using four predefined moisture-retention characteristics [7]: Van Genutchen ordinary (1) and extended (2), Huyakorn (3), and Brooks-Corey (4). The analytical forms of the capillarity and retention curves are reported in [25]. These characteristics curves are selected according to the value (1,2,3,4) of parameter `Par%IVGHU` of the same input file `soil` and the values of the corresponding curve coefficients (Appendix B);
  - hydrogeological parameters:
    - diagonal coefficients K_{xx}, K_{yy}, K_{zz} of the saturated hydraulic conductivity tensor at reference conditions (ρ_0, μ_0), i.e. variables `MS_prop%PERMX`, `MS_prop%PERMY` and `MS_prop%PERMZ`, respectively.
    - the elastic storage coefficient S_{sat} which is the variable `MS_prop%elstor`;
    - the porosity ϕ which is the variable `MS_prop%poros`.

  Typical values of these hydrogeological parameters for different soils/rocks are reported in [3, 23]; These parameters are set for each superficial hydrogeological zone `Dim%NZONE` and each vertical stratum `Dim%NSTR` according to the input scheme of the file:

  $K_{xx}, K_{yy}, K_{zz}\phi, S_{sat}$ `(superf. zone 1 of vertical stratum 1)`
  $K_{xx}, K_{yy}, K_{zz}\phi, S_{sat}$ `(superf. zone 2 " ")`

[27] This coordinate frame rotation is done in `SUBROUTINE ANIS3D`.
[28] Current implementation of CODESA-3D allows for the definition of a single soil characterization for the whole domain.

...

$K_{xx}, K_{yy}, K_{zz}\phi, S_{sat}$ (superf. zone NZONE " ")

...

...

$K_{xx}, K_{yy}, K_{zz}\phi, S_{sat}$ (superf. zone 1 of vertical stratum NSTR)

$K_{xx}, K_{yy}, K_{zz}\phi, S_{sat}$ (superf. zone 2 " ")

...

$K_{xx}, K_{yy}, K_{zz}\phi, S_{sat}$ (superf. zone n " ")

- input file `solute`
  * reference values of density ρ_o (`MS_prop%rho0`) and viscosity μ_o (`MS_prop%mu0`);
  * density difference ratio $\epsilon = (\rho_s - \rho_o)/\rho_o \approx 0.025 \div 0.030$ (`MS_prop%epsilon`) and viscosity difference ratio $\epsilon' = (\mu_s - \mu_o)/\mu_o \approx 1$ (`MS_prop%epsilon1`);
  * longitudinal α_L and transversal α_T dispersivity, namely variables `MS_prop%ALFAL` and `MS_prop%ALFAT`;
  * molecular diffusion coefficient D_o, which is the variable `MS_prop%DIFFUS`.

CODESA-3D input data, with reference to the initial and boundary conditions are

- the initial conditions (IC's) at the simulation starting time which are supplied for all nodes of the computational mesh:
  - flow equation – file `flowic`. IC's can be specified both in terms of *freshwater* pressure heads (ψ) or *freshwater* total heads ($h = z + \psi$) according to the value (`.FALSE.`, `.TRUE.`) of logical flag `Par%POT_IC_FLAG`; only one value to be replicated to all nodes (homogeneous IC's) or `Dim%N` nodal values, referring to the 3-D mesh, can be read according to the value (0;1) of integer parameter `Par%INDP`;
  - transport equation – file `tranic`. IC's are specified in terms prescribed relative concentrations (c); only one value to be replicated to all nodes or `Dim%N` nodal values, referring to the 3-D mesh, can be read according to the value (0;1) of integer parameter `Par%INDPC`.

- the boundary conditions (BC's) must be supplied for selected nodes of the computational mesh; In FE formulations when a domain boundary has no associated field value or prescribed fluxes, it is implicitly considered as an *impervious boundary* to flow. This is true for the coupled system (11) thus if we do not specify nothing for a domain boundary, that is assumed impervious to water and salt flow.
  - flow equation. BC's can be specified as Dirichlet and Neumann types in the input files `nansfnodbc` [29], which contains the list of boundary nodes, `nansfdirbc` and `nansneubc`, which contain the list of boundary Dirichlet and Neumann values, respectively. Another "switching"-type of BC's is set in file `sfbc`, which contains only the list of those boundary nodes since the corresponding boundary values are computed by the model at each time step.
    * **Dirichlet** BC's: The assigned boundary nodes, contained in file `nansfnodbc` are possibly subdivided in two class of nodes: ground surface nodes (`Dim%NDIR`) on which the BC's are to be replicated along depth or 3-D nodes (`Dim%NDIRC`) in the real 3-D space (not to be replicated). Boundary values, contained in the file `nansfdirbc`, can be specified both in terms of *freshwater* pressure heads (ψ)

[29] The acronym NANSFBC stands for Non-Atmospheric Non-Seepage-Face BC.

or *freshwater* total heads ($h = z + \psi$) according to the value (`.FALSE.`, `.TRUE.`) of the logical flag `Par%POT_BC_FLAG` set in the `nansfnodbc` input file.

  * **Neumann** BC's:
    - **ground surface distributed fluxes** $[MT^{-1}]$ (*infiltration rate*) on the 2-D ground surface nodes. These infiltration rates are specified in the `atmbc` input file and can be read as a single value to be replicated for each node or as `Dim%NNOD` values, referring to the 2-D mesh, according to the value (0,1) of the integer parameter `Flow_BC%HSPATM`. It is the model that integrates the infiltration rate on the area belonging to each 2-D node to compute the inlet fluxes;
    - **nodal fluxes** $[M^3/S]$ on 3-D selected nodes, for instance given withdrawal/injection rates at the well screens. These boundary fluxes are contained in the `nansfneubc` input file;
  * **switching** type: a list of nodes can be specified in file `sfbc`, whose BC's can be switched off from Dirichlet to Neumann type and vice versa according to current flow regime. These boundary conditions allow to simulate atmospheric outlet faces (*seepage faces*) of the given flow domain, where according to the flow directions we can have prescribed heads or assigned fluxes. The corresponding boundary values are re-assigned by the model at each time step.

  Boundary values can vary in space and/or time; Time variation of boundary values is set according to the following input scheme:

```
time 1
list of BC values at time 1
time 2
list of BC values at time 2
...
time n
list of BC values at time n
```

  Linear interpolation of the given boundary values is used at intermediate time steps.

- transport equation. BC's are be specified in the input file `tranbc` as Dirichlet and Cauchy (total flux, i.e. advective + dispersive fluxes) node lists interlaced by the corresponding Dirichlet and Cauchy prescribed relative concentrations [/] and *total* fluxes $[M^3/T]$, respectively.

  Current implementation of CODESA-3D does not allow time-varying transport BC's, also it does not allow to prescribe directly only the advective nor the dispersive components of the flux.

With reference to the *flow* and *transport* equations (4), CODESA-3D output data includes:

- mirroring of input data and terminal output (`result.OUT` file);
- print of 3-D automatically generated grid (`xyz.out` file);
- at each time step:
  - information about convergence behavior of the nonlinear iterative scheme and linear solver both for the flow and transport equation (`iter.out` file);
  - information about the flow mass balance errors (`flow-mb.out` file);

- information about the transport balance errors (`tran-mb.out` file);
- unknowns (ψ and c) and related variables ($S_w, k_{rw}, \mathbf{v}$ etc) at user selected nodes (`result.OUT` file);
- hydrographs showing the atmospheric, seepage, overland (runoff) and subsurface fluxes (`atmsf-hg.out` and `nansf-hg.out` file);

◦ at the end of simulation and at user-defined selected time steps:

- unknowns (ψ and c) and related variables ($S_w, k_{rw}, \mathbf{v}$ etc) on all mesh nodes (`psi.out`, `conc.out`, `sw.out`, `ckrw.out`, `vel.out` files);
- Darcy velocities $\mathbf{v}$ on all tetrahedra (`vel-el.out` file);
- ψ, c, S_w on all surface nodes, along with a saturation index indicating whether the node is saturated or not (`psisurf.out`, `concsurf.out`, `swsurf.out`, `satsurf.out` files);
- vertical profiles of unknowns (ψ and c) on user-defined surface nodes (`vp-psi.out`, `vp-conc.out` files);

The number and the content of activated output files are chosen according to input parameters `Par%IPRT` and `Par%IPRT1` of input file `parm` (Appendix B).

Appendix C shows the whole input dataset for an example test-case.

A.2 Numerical integration

With reference to a 3D domain composed by N nodes and N_e finite elements, the solutions sought $\psi(x,y,z,t)$ and $c(x,y,z,t)$ for coupled system (4) are expressed as linear combination of the nodal discretized unknowns $\hat{\boldsymbol{\psi}} = [\hat{\psi}_1, \hat{\psi}_2, \ldots, \hat{\psi}_N]^T$ and $\hat{\mathbf{c}} = [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_N]^T$ through the linear shape functions N:

$$\psi(x,y,z,t) \approx \hat{\psi} = \sum_{I=1}^{N} N_I(x,y,z) \cdot \hat{\psi}_i(t)$$

and

$$c(x,y,z,t) \approx \hat{c} = \sum_{I=1}^{N} N_I(x,y,z) \cdot \hat{c}_i(t)$$

The linear shape function $N_I(x,y,z)$ is defined [34, 1]:

$$N_i = \frac{(a_i + b_i x + c_i y + d_i z)}{6\Omega^e} \tag{22}$$

with Ω^e the volume of the e-th tetrahedron [30] of nodes i, j, m and n is given by: $\Omega^e = ||\Delta^e||/6$ being Δ^e the determinant:

$$\Delta^e = det \begin{bmatrix} 1 & x_I & y_I & z_I \\ 1 & x_J & y_J & z_J \\ 1 & x_M & y_M & z_M \\ 1 & x_N & y_N & z_N \end{bmatrix}$$

[30] The superscript e, denoting variables related to the generic tetrahedron, will be omitted in the following where not necessary.

The coefficients a_i, b_i, c_i, d_i are the determinants of the lower rank matrices of the above matrix. For example:

$$a_i = det \begin{bmatrix} x_J & y_J & z_J \\ x_M & y_M & z_M \\ x_N & y_N & z_N \end{bmatrix}$$

$$b_i = -det \begin{bmatrix} 1 & y_J & z_J \\ 1 & y_M & z_M \\ 1 & y_N & z_N \end{bmatrix}$$

$$c_i = -det \begin{bmatrix} x_J & 1 & z_J \\ x_M & 1 & z_M \\ x_N & 1 & z_N \end{bmatrix}$$

$$d_i = -det \begin{bmatrix} x_J & y_J & 1 \\ x_M & y_M & 1 \\ x_N & y_N & 1 \end{bmatrix}$$

The other coefficients are obtained by cyclic interchange of the subscripts in the given order i,j,m and n.

The flow and transport global coefficient matrices/vectors given in equations (11) are built up by the *assembling* of elemental contributions (matrices and vectors), coming from the tetrahedra composing the computational mesh. Process of assembly of the generic matrix **X** $(N \times N)$ is usually written as:

$$\mathbf{X} = \sum_{1}^{N_e} \mathbf{X^e}$$

where $\mathbf{X^e}$ $(N \times N)$ is the generic elemental matrices of e-th tetrahedron, that has only four rows and columns different from zero: i.e., only those corresponding to the nodes belonging to the e-th tetrahedron [31].

In the following are given the coefficients of elemental matrices (4×4) and vectors (4×1) of the numerical model (11).

A.2.1 Flow equation

The coefficients of elemental stiffness (flow) matrix $\mathbf{H}^e$ are:

$$h^e_{ij} = \int_{\Omega^e} \mathbf{K}^e \nabla N_i \cdot \nabla N_j \; d\Omega = (K^e_x \frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + K^e_y \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + K^e_z \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}) \int_{\Omega^e} d\Omega \qquad (23)$$

which becomes:

$$\boxed{h^e_{ij} = k^e_{rw} \frac{(1+\epsilon\overline{c})}{(1+\epsilon'\overline{c})} \cdot \frac{(K^e_{sat0,x} b_i b_j + K^e_{sat0,y} c_i c_j + K^e_{sat0,z} d_i d_j)}{36\Omega^e}} \qquad (24)$$

[31] These elemental matrices are stored as (4×4) dense matrices and then connectivity matrix TETRA is used, via indirect addressing, to assemble them into the global matrix

where i, j is each couple of global nodes of the element e, $\overline{c}$ is the time averaged relative concentration c at the current time step: [32] $\overline{c} = \omega_f \tilde{c}^{t+\Delta t} + (1-\omega_f)\tilde{c}^t$, with $\tilde{c}$ the element-wise averaged concentration and $K^e_{sat0,i}$ is the i-th diagonal coefficient of the saturated hydraulic conductivity tensor at reference conditions (ρ_o, μ_o). Definition and assembling into left hand side matrix of system (11) of matrix $\mathbf{H}^e$ is done in `SUBROUTINE ASSPIC` and `SUBROUTINE ASSNEW` according to the value (1;2) of input integer parameter `Par%IFLOW`, that select Picard or Newton linearization scheme, respectively.

FLOW3D implementation for coefficients h_{ij} is identical to those in equation (24). The variably saturated hydraulic conductivity tensor is there simply $\mathbf{K}^e = k^e_{wr}\mathbf{K}^e_{sat0}$, since c is not present. The same stands for SATC3D code where $\mathbf{K}^e \equiv (1+\epsilon\overline{c})K^e_{sat0}$.

The coefficients of elemental mass (capacity) matrix $\mathbf{P}^e$ are:

$$p^e_{ij} = \sigma^e \int_\Omega^e N_i N_j \; d\Omega \tag{25}$$

which becomes: [33]

$$\boxed{p^e_{ij} = \sigma^e \frac{\Omega^e}{20} \cdot (1+\delta_{ij})} \tag{26}$$

where δ_{ij} is the Kronecker delta, with $\delta_{ij} = 0$ if $i \neq j$ and $\delta_{ij} = 1$ if $i = j$.

FLOW3D implementation for coefficients p_{ij} are identical to those in equation (26). The overall storage term is there simply $\sigma^e = \sigma^\star$. The same stands for SATC3D code where $\sigma^e \equiv (1+\epsilon\overline{c})S_{sat}$.

Definition and assembling into left hand side matrix of system (11) of matrix $\mathbf{H}^e$ is done in `SUBROUTINE ASSPIC` and `SUBROUTINE ASSNEW` according to the value (1;2) of input integer parameter `Par%IFLOW`, that select Picard or Newton linearization scheme, respectively.

The coefficients of elemental vector $\mathbf{q}^{\star e}$ are made up of three different contributions

$$\mathbf{q}^{\star e} = \mathbf{g}^e + \mathbf{f}^e + \mathbf{l}^e$$

taking into account the gravitational term, the term referring to the time derivative of the concentration and the term containing prescribed extracted/injected Neumann fluxes: concentrated and surface distributed as well. The gravitational term integration furnishes:

$$g^e_i = \int_\Omega^e k^e_{rw} K^e_{sat0,z} \frac{(1+\epsilon\overline{c})^2}{(1+\epsilon'\overline{c})} \frac{\partial N_I}{\partial z} \; d\Omega \tag{27}$$

which becomes:

$$\boxed{g^e_i = k^e_{rw} K^e_{sat0,z} \frac{(1+\epsilon\overline{c})^2}{(1+\epsilon'\overline{c})} d_i \frac{||\Omega^e||}{6\Omega^e}} \tag{28}$$

The term incorporating the time variation of concentration is:

$$f^e_i = \int_\Omega^e \phi^e S^e_w \epsilon \frac{\partial \overline{c}}{\partial t} N_i d\Omega \tag{29}$$

[32] Relative concentration c is considered as a constant during the solution of the flow equation.

[33] The useful integration formula here is: $\int_\Omega N_i^a N_j^b N_k^c N_l^d d\Omega = \frac{a!b!c!d!}{(a+b+c+d+3)!} 6\Omega$.

which becomes:

$$\boxed{f_i^e = \phi^e S_w^e \epsilon \frac{\overline{c}^{t+\Delta t} - \overline{c}^t}{\Delta t} \frac{||\Omega^e||}{4}} \tag{30}$$

Definition and assembling into right hand side vector of system (11) of vector $\mathrm{g}^e + \mathbf{f}^e$ is done in `SUBROUTINE RHSGRVCON` or `SUBROUTINE RHSGRV` according to the value (1;0) of input integer parameter `Par%ITRANS`, that select coupled (flow & transport) or simply flow simulator, respectively. The third term incorporating Neumann boundary fluxes is [34]:

$$l_i^e = \int_{\Gamma_2}^e q_n^e N_i d\Gamma_2 + \int_{\Omega}^e \frac{\rho}{\rho 0} q^e N_i d\Omega \tag{31}$$

which becomes:

$$\boxed{l_i^e = -q_n^e \frac{||\Gamma_2^e||}{3} - (1 + \epsilon\overline{c}) q^e \frac{||\Omega^e||}{4}} \tag{32}$$

where $||\Gamma^e||$ denotes the area of the e-th tetrahedron face where the flux is imposed. Neumann boundary condition setting into right hand side vector of system (11) is done in `SUBROUTINE BCPIC` or `SUBROUTINE BCNEW` according to the value (1;2) of input integer parameter `Par%IFLOW`.

A.2.2 Transport equation

The coefficients of elemental stiffness matrix $\mathbf{K}^e$ are made up of three different contributions:

$$\mathbf{K}^e = \mathbf{A}^e + \mathbf{B}^e + \mathbf{C}^e$$

referring to the dispersive and advective terms, and to the Cauchy total flux prescribed on the boundary.

Coefficients of elemental matrix $\mathbf{A}^e$ are:

$$a_{ij}^e = \int_{\Omega^e} \mathbf{D}^e \nabla N_i \cdot \nabla N_j \; d\Omega = (D_x^e \frac{\partial N_i'}{\partial x} \frac{\partial N_j'}{\partial x} + D_y^e \frac{\partial N_i'}{\partial y} \frac{\partial N_j'}{\partial y} + D_z^e \frac{\partial N_i'}{\partial z} \frac{\partial N_j'}{\partial z}) \int_{\Omega^e} d\Omega \tag{33}$$

which becomes:

$$\boxed{a_{ij}^e = \frac{(D_x^e b_i' b_j' + D_y^e c_i' c_j' + D_z^e d_i' d_j')}{36\Omega^e}} \tag{34}$$

where N' is the basis function calculated in the local (rotated) reference frame. SATC3D implementation for coefficients a_{ij} is identical to those in equations (34) with the coefficient of dispersivity tensor **D** calculated with water saturation $S_w = 1$ and the Darcy velocity (5) computed with $k_{rw} = 1$ and $\epsilon' = 1$.

Coefficients of elemental matrix $\mathbf{B}^e$ are:

$$b_{ij}^e = \int_{\Omega^e} \nabla \cdot (\mathbf{v}^e N_i) N_j \; d\Omega = \int_{\Omega^e} d\Omega (v_x^e \frac{\partial N_i'}{\partial x} + v_y^e \frac{\partial N_i'}{\partial x} + v_z^e \frac{\partial N_i'}{\partial x}) N_j \; d\Omega \tag{35}$$

[34] Current CODESA-3D implementation reads directly from boundary condition input file `nansfneubc` the nodal values of the fluxes as already integrated on the belonging surface or volume. This means that term l_i^e is not currently implemented.

which becomes:

$$\boxed{b^e_{ij} = (v^e_x b_i + v^e_y c_i + v^e_z d_i)\frac{||\Omega^e||}{24\Omega^e}} \tag{36}$$

Coefficients of elemental matrix $\mathbf{C}^e$ are:

$$c^e_{ij} = -\int_{\Gamma^e_4} \mathbf{v}^e \cdot \mathbf{n} N_i N_j \; d\Gamma = \mathbf{v}^e \cdot \mathbf{n} \int_{\Gamma^e_4} N_i N_j \; d\Gamma \tag{37}$$

which becomes:

$$\boxed{c^e_{ij} = -(v^e_x n_x + v^e_y n_y + v^e_z n_z)\frac{||\Gamma^e_5||}{12}(1 + \delta_{ij})} \tag{38}$$

where where $||\Gamma^e_4||$ denotes the area of the e-th tetrahedron face where the Neumann (dispersive) flux is imposed.

SATC3D implementation for coefficients a_{ij} and b_{ij} is identical to those in equations (34) and (36) with the coefficient of dispersivity tensor **D** calculated with water saturation $S_w = 1$ and the Darcy velocity (5) computed with $k_{rw} = 1$ and $\epsilon' = 1$.

The coefficients of elemental mass matrix $\mathbf{M}^e$ are:

$$m^e_{ij} = \phi^e S^e_w \int_\Omega^e N_i N_j \; d\Omega \tag{39}$$

which becomes:

$$\boxed{m^e_{ij} = \phi^e S^e_w \frac{\Omega^e}{20} \cdot (1 + \delta_{ij})} \tag{40}$$

Definition and assembling into left hand side matrix of system (11) of matrices $\mathbf{K}^e$ and $\mathbf{M}^e$ is done in `SUBROUTINE ASSTRN`, both for the Picard- or partial Newton-linearized transport equation, according to the value (1;2) of variable `Par%ITRANS`, respectively.

The coefficients of elemental vector $\mathbf{r}^{\star e}$ are made up of two different contributions

$$\mathbf{r}^{\star e} = \mathbf{rv}^e + \mathbf{rs}^e$$

taking into account volumetric and surface distributed (Neumann and Cauchy) source/sink terms, respectively [35].

The coefficients of elemental vector $\mathbf{rv}^e$ are:

$$rv^e_i = -\int_\Omega^e (q^e c\star + f^e) N_i \; d\Omega \tag{41}$$

which becomes:

$$\boxed{rv^e_i = -(q^e c\star + f^e)\frac{||\Omega^e||}{4}} \tag{42}$$

[35] Current CODESA-3D implementation reads directly from boundary condition input file `tranbc` the nodal values of the Cauchy (total) fluxes as already integrated on the belonging surface or volume. This means that term $r^{\star e}_i$ is not currently implemented.

The coefficients of elemental vector $\mathbf{rs}^e$ are:

$$\mathrm{rs}_i^e = -(\int_{\Gamma_4}^{e} q_d N_i \; d\Gamma_4 + \int_{\Gamma_5}^{e} q_c N_i \; d\Gamma_5) \tag{43}$$

which becomes:

$$\boxed{\mathrm{rs}_i^e = -q_d \frac{||\Delta\Gamma_4||}{3} - q_c \frac{||\Delta\Gamma_5||}{3}} \tag{44}$$

B Appendix B

The appendix reports the catalog of CODESA-3D variables and their hierarchy.

```
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C
C 3-D coupled saturated flow and transport
C MP/CP, Sept/91
C
C 3-D coupled unsaturated flow and transport
C GL,    Oct/96
C
C-----------------------------------------------------------------------
C
C      PROGRAM CODESA-3D
C
C
C     COupled variable DEnsity and SAturation 3-Dimensional model
C
C
C--------------------------------LIST OF DERIVED DATA USED
C
C  1.) Dim       - actual dimensioning parameters
C  2.) Par       - general parameters values and arrays
C  3.) IO        - I/O parameters and arrays
C  4.) CPU       - CPU timing parameters and arrays
C  5.) Grid      - Grid parameters and arrays
C  6.) Flow_BC   - Flow Boundary Condition (BC) parameters and arrays
C  7.) Transp_BC - Transport BC parameters and arrays
C  8.) MBal      - Mass balance parameters and arrays
C  9.) MS_prop   - Material and solute parameters and arrays
C    9.1) VG       - Van Genutchen curve data
C    9.2) HU       - Huyakorn curve data
C    9.3) BC       - Brooks-Corey curve data
C 10.) Sys       - Linear system parameters and arrays
C 11.) Out       - Model output parameters and arrays
C
C
C Notes: a.)  field ``pippo'' of data structure ``Dim'' in the code
C             will be referred to as Dim%pippo;
C        b.)  field ``pluto'' of data structure ``VG'' in the code
C             of data structure MS_prop will be referred to as
C             MS_prop%VG%pluto.
C
C
C--------------------------------DESCRIPTION OF PARAMETERS AND VARIABLES
C
C
C.....................................................................
C
C *MACHINE PRECISION for REAL NUMBERS*
C
C contained in the module ``mod_kind'' (file: mod_kind.F):
C
```

```
C
C Integer Parameters:
C
C   MY_PRECISION  - =4 for ``simple'' precision (4 bytes);
C                   =8 for ``double'' precision (8 bytes).
C.......................................................................
C
C *ACTUAL DIMENSIONING PARAMETERS*
C
C contained in the data structure ``Dim'' of derived type ``Dim_tag''
C (file: mod_Dim.F):
C
C
C Integer Parameters:
C
C   IBOT   - size of real working storage for NONSYM solver (# of
C            nonzero elements in the LU decomposition of system matrix)
C   MINBOT - minimum IBOT
C   INONSY - set to 1 when using the NONSYM direct solver for
C            unsymmetric linear systems. Can be set to 0
C            otherwise (this will reduce the storage requirements
C            when not using the NONSYM solver).
C   MAXBOT - INONSY*(maximum IBOT) + 1
C            (defined real working storage dimension for NONSYM solver)
C   INTBOT - INONSY*(MAXBOT + 6*(NMAX + INEW*NMAX) + 1) + 1
C            (defined integer working storage dimension for NONSYM
C            solver)
C   NNOD   - # of nodes in 2-d mesh. These are the surface nodes for
C            the 3-d mesh -- they are all designated as atmospheric
C            boundary condition nodes (rainfall and evaporation inputs),
C            except for those surface nodes which are specifically
C            designated as non-atmospheric BC's (see description of
C            NDIR, NDIRC, NQ, NSF, and IFATM).
C   NTRI   - # of triangles in 2-d mesh
C   NTRI3  - # of triangles in 2-d mesh multiplied by three (3)
C   NSTR   - # of vertical layers
C   NZONE  - # of material types in the porous medium
C   N1     - maximum # of element connections to a node (symmetric case)
C   N1C    - maximum # of element connections to a node (unsymmetric case)
C
C   N      - NNOD*(NSTR + 1) = # of nodes in 3-d mesh
C   NT     - 3*NTRI*NSTR = # of tetrahedra in 3-d mesh
C   NPT    - N1*N
C   NTERM  - # of nonzero elements in flow system matrices
C             (symmetric storage used for Picard scheme;
C              unsymmetric storage for Newton)
C   NTERMC - # of nonzero elements in transport system matrices
C             (unsymmetric storage)
C
C   NDIR   - # of non-atmospheric, non-seepage face Dirichlet nodes
C            in 2-d mesh. The BC's assigned to these surface nodes
C            are replicated vertically (compare NDIRC).
C   NDIRC  - # of 'fixed' non-atmospheric, non-seepage face Dirichlet
C            nodes in 3-d mesh ('fixed' in the sense that these BC's
C            are not replicated to other nodes - compare NDIR)
C   NP2C   - # of Dirichlet nodes in 2-d mesh for the transport
C            (concentration) equation
```

```
C   NPFC   - # of 'fixed' Dirichlet nodes in 3-d mesh for the
C            transport equation
C   NP     - NDIR*(NSTR + 1) + NDIRC = total # of non-atmospheric,
C            non-seepage face Dirichlet nodes in 3-d mesh
C   NPC    - NP2C*(NSTR + 1) + NPFC  = total # of Dirichlet nodes
C            in 3-d mesh for the transport equation
C   NQ     - # of non-atmospheric, non-seepage face Neumann nodes
C            in 3-d mesh
C   NMC    - # of Cauchy (third type, or mixed, BC) nodes in 3-d
C            mesh for the transport (concentration) equation
C   NSF    - # of seepage faces (see description of NSFNOD)
C   NNSFMX - maximum # of nodes on a seepage face + 1
C
C   NUMDIRMX  -  maximum total # of Dirichlet nodes in 3-d mesh
C   ITMX   - maximum allowable iterations per time step in solving
C            the nonlinear flow equation
C   ITMXC  - maximum allowable iterations per time step in solving
C            the nonlinear coupled flow and transport system
C   NR     - # of nodes selected for partial output
C   NPRT   - # of time values for detailed nodal output and element
C            velocity output (see description of IPRT, TIMPRT)
C   NUMVP  - # of surface nodes for vertical profile output
C
C.......................................................................
C
C *GENERAL PARAMETERS*
C
C contained in the data structure ``Par'' of derived type ``Par_tag''
C  (file: mod_Par.F):
C
C Integer Parameters:
C
C   IMAX   - largest integer number (machine dependent)
C
C   INDP   - =0 for input of flow uniform initial conditions (one value
C               read in)
C            =1 for input of flow nonuniform IC's (one value read in
C               for each node)
C            =2 for calculation of fully saturated vertical hydrostatic
C               equilibrium IC's (calculated in subroutine ICVHE)
C
C   INDPC  - =0 for input of concentration uniform initial conditions
C               (one value read in)
C            =1 for input of concentration nonuniform IC's
C               (one value read in for each node)
C
C   ITRANS - =1 for uncoupled model (only flow)
C            =1 for coupled model   (flow & transport)
C   IFLOW  - =1 for Picard iteration scheme
C            =2 for Newton iteration scheme
C   ISOLV  - flag for unsymmetric linear solver (flow equation)
C            =-5   BiCGSTAB (preconditioned with D$^{-1}$)
C            =-4   BiCGSTAB (not preconditioned)
C            =-3   TFQMR (preconditioned with D$^{-1}$)
C            =-2   TFQMR (not preconditioned)
C            =-1   TFQMR (preconditioned with K$^{-1}$)
C            =0    BiCGSTAB (preconditioned with K$^{-1}$)
```

```
C           =1    GRAMRB  (minimum residual)
C           =2    GCRK(5) (ORTHOMIN)
C           =3    IBM's NONSYM (direct solver)
C    ISOLVC - flag for unsymmetric linear solver  (transport equation)
C           =-5   BiCGSTAB (preconditioned with D$^{-1}$)
C           =-4   BiCGSTAB (not preconditioned)
C           =-3   TFQMR (preconditioned with D$^{-1}$)
C           =-2   TFQMR (not preconditioned)
C           =-1   TFQMR (preconditioned with K$^{-1}$)
C           =0    BiCGSTAB (preconditioned with K$^{-1}$)
C           =1    GRAMRB  (minimum residual)
C           =2    GCRK(5) (ORTHOMIN)
C           =3    IBM's NONSYM (direct solver)
C    IRELAX - flag for nonlinear relaxation (flow equation)
C           =0 no relaxation
C           =1 relaxation with constant relaxation parameter OMEGA
C           =2 relaxation with iteration-dependent relaxation parameter
C              OMEGA, calculated using Huyakorn et al's adaptation
C              (WRR 1986 22(13), pg 1795) of Cooley's empirical scheme
C              (WRR 1983 19(5), pg 1274)
C    IRELAXC- flag for nonlinear relaxation (transport equation)
C           =0 no relaxation
C           =1 relaxation with constant relaxation parameter OMEGA
C           =2 relaxation with iteration-dependent relaxation parameter
C              OMEGA, calculated using Huyakorn et al's adaptation
C              (WRR 1986 22(13), pg 1795) of Cooley's empirical scheme
C              (WRR 1983 19(5), pg 1274)
C    NSTEP  - time step index
C    LUMP   - =0 for flow distributed mass matrix; otherwise matrix is lumped
C    LUMPC  - =0 for transport distributed mass matrix;
C              otherwise matrix is lumped
C    ITER   - iteration index for flow nonlinear iterations for each time step
C    ITERC  - iteration index for coupled syst. nonlinear iterations for
C           each time step
C    ITMX1  - if ITER < ITMX1, flow eq. time step size is increased
C    ITMXC1 - if ITER < ITMX1, coupled syst. time step size is increased
C    ITMX2  - if ITMX1 <= ITER < ITMX2, flow eqn. time step size is not altered
C           if ITMX2 <= ITER < ITUNS, flow eqn. time step size is decreased
C           if ITER = ITUNS (i.e. convergence not achieved in ITUNS
C           iterations), we back-step unless time step size cannot
C           be reduced any further (DELTAT = DTMIN). Back-stepping is
C           also triggered if the linear solver failed (LSFAIL = TRUE)
C           or if the convergence or residual errors become larger
C           than ERNLMX (ERRGMX = TRUE).
C    ITMXC2 - if ITMXC1 <= ITER < ITMXC2, coupled syst. time step size
C           is not altered
C           if ITMX2 <= ITER < ITUNS, coupled syst. time step size
C           is decreased
C           if ITER = ITUNS (i.e. convergence not achieved in ITUNS
C           iterations), we back-step unless time step size cannot
C           be reduced any further (DELTAT = DTMIN). Back-stepping is
C           also triggered if the linear solver failed (LSFAIL = TRUE)
C           or if the convergence or residual errors become larger
C           than ERNLMX (ERRGMX = TRUE).
C    ITMXCGSY- maximum # of iterations for conjugate gradient linear
C             symmetric system solvers
C    ITMXCGNS- maximum # of iterations for conjugate gradient-type linear
```

```
C            unsymmetric system solvers
C   KBACK  - total number of flow eqn. back-stepping occurrences
C   KBACKC - total number of coupled syst. back-stepping occurrences
C   ITTOT  -
C   ITTOTC -
C   NITER  - number of iterations for the linear solver at each
C            nonlinear iteration of the flow equation
C   NITERC - number of iterations for the linear solver at each
C            nonlinear iteration of the coupled system of eqns.
C   NITERT - number of iterations for the linear solver at each
C            time step of the flow equation
C   NITERTC- number of iterations for the linear solver at each
C            time step of the coupled system of eqns.
C   ITLIN  - total number of iterations for the linear solver over all
C            nonlinear iterations and all time steps of the flow
C            equation
C   ITLINC - total number of iterations for the linear solver over all
C            nonlinear iterations and all time steps of the coupled
C            system of eqns.
C   ICNVRF -
C   ICNVRC -
C   KLSFAI - total number of linear solver failures for the flow eqn.
C   L2NORM - =0 to use the infinity norm in the test for convergence of
C            the nonlinear iterations; otherwise the L2 norm is used
C
C Integer Arrays:
C
C   IER   (7)          - error flags
C   IP3   (3,3)        - 3 x 3 permutation matrix
C   IP4   (4,4)        - 4 x 4 permutation matrix
C
C Real Parameters:
C
C   RMIN   - smallest double precision number (machine dependent)
C   RMAX   - largest double precision number (machine dependent)
C   TETAF  - weighting parameter for time stepping scheme
C            of the flow equation
C            (1.0 Backward Euler; 0.5 Crank-Nicolson;
C            TETAF is set to 1.0 for steady state problem)
C   TETAC  - weighting parameter for time stepping scheme
C            of the coupled system
C            (1.0 Backward Euler; 0.5 Crank-Nicolson;
C            TETAC is set to 1.0 for steady state problem)
C   DELTAT - initial and current time step size (DELTAT >= 1.0e+10
C            on input indicates steady state problem)
C   DTMIN  - minimum time step size allowed
C   DTMAX  - maximum time step size allowed
C   TMAX   - time at end of simulation (TMAX is set to 0.0 for
C            steady state problem)
C   DTAVG  - average time step size used for the simulation
C   DTSMAL - smallest time step size used during the simulation
C   DTBIG  - largest time step size used during the simulation
C   TSMAL  - first time at which DTSMAL is used
C   TBIG   - first time at which DTBIG is used
C   DTMAGA - magnification factor for time step size (additive)
C   DTMAGM - magnification factor for time step size (multiplicative)
C   DTREDS - reduction factor for time step size (subtractive)
```

```
C   DTREDM - reduction factor for time step size (multiplicative)
C   TIME   - time at current time level
C   TIMEP  - time at previous time level
C   TOLNL  - tolerance for convergence of nonlinear flow iterations
C   TOLNLC - tolerance for convergence of nonlinear coupled iterations
C   TOLCGSY- tolerance for convergence of conjugate gradient linear
C            system solvers  (symmetric matrices)
C   TOLCGNS- tolerance for convergence of conjugate gradient-type linear
C            system solvers (unsymmetric matrices)
C   OMEGA  - nonlinear relaxation parameter for the flow eqn.:
C            OMEGA > 1, over-relaxation;
C            OMEGA < 1, under-relaxation. Input value of OMEGA is used
C            only for the case NLRELX=1 (constant relaxation parameter).
C            Input value of OMEGA is ignored otherwise: for NLRELX=0
C            relaxation is not applied; for NLRELX=2 OMEGA is calculated
C            at each nonlinear iteration.
C   OMEGAC - nonlinear relaxation parameter for the coupled system of eqns.:
C            OMEGAC > 1, over-relaxation;
C            OMEGAC < 1, under-relaxation. Input value of OMEGAC is used
C            only for the case NLRELX=1 (constant relaxation parameter).
C            Input value of OMEGAC is ignored otherwise: for NLRELX=0
C            relaxation is not applied; for NLRELX=2 OMEGAC is calculated
C            at each nonlinear iteration.
C   OMEGAP - OMEGA value at previous nonlinear iteration
C   OMEGAPC- OMEGAC value at previous nonlinear iteration
C   ERNLMX - maximum allowable convergence or residual error in the
C            nonlinear solution. If the convergence or residual errors
C            become larger than ERNLMX, ERRGMX is set to TRUE and the
C            code back-steps. This avoids occurrences of overflow or
C            underflow when nonlinear iterations diverge
C
C Logical Flags:
C
C   DTGMIN - flag indicating whether the current time step size
C            is greater than the minimum allowed
C             FALSE if not greater
C             TRUE  if greater
C   LSFAIL - flag for linear solver
C             FALSE if linear solver did not fail
C             TRUE  if linear solver failed
C   NORMCV - flag for convergence of the norm of pressure head
C            differences in the nonlinear iterative procedure
C             FALSE if the norm has not converged
C             TRUE  if the norm has converged
C   ITAGEN - flag indicating whether we can iterate again
C            in the nonlinear iterative procedure
C             FALSE if we cannot iterate again
C             TRUE  if we can iterate again
C   ERRGMX - flag indicating whether the convergence or residual errors
C            have become greater than the allowed maximum
C             FALSE if not greater
C             TRUE  if greater
C   POT_BC_FLAG - flag indicating whether the BC's are read in
C                 terms of potential heads in the input file.
C                 FALSE pressure head values are read
C                 TRUE  potential head values are read
C   POT_IC_FLAG - flag indicating whether the IC's are read in
```

```
C                 terms of potential heads in the input file.
C                 FALSE pressure head values are read
C                 TRUE  potential head values are read
C   FLOW_EXIT_FLAG - flag indicating whether the flow simulation
C                    has got to an end
C                    FALSE simulation continues
C                    TRUE  simulation is finished
C   COUPLED_EXIT_FLAG - flag indicating whether the coupled simulation
C                       has got to an end
C                       FALSE simulation continues
C                       TRUE  simulation is finished
C   OSC_FLAG -          flag indicating whether the Orthogonal Subdomain
C                       Collocation (OSC) scheme is adopted
C                       FALSE OSC not used
C                       TRUE  OSC is used
C.......................................................................
C
C *Input/Output (I/O) PARAMETERS*
C
C contained in the data structure ``IO'' of derived type ``IO_tag''
C (file: mod_IO.F):
C
C Integer Parameters:
C
C   MAX_FILE_LEN  - maximum lenght for file names
C   IN<X>         - input unit number  (0 <= X <=12)
C
C List of Input Units:
C -------------------
C
C        IN0         - I/O file names ``codesa3d.fnames''
C
C        IN1   = 5   - paramteters ``parm''
C        IN2   = 8   - grid info   ``grid''
C        IN3   = 9   - nodes with non-atmospheric, non-seepage face
C                      Neumann and Dirichlet BC's ``flow-nansfbc''
C        IN4   = 10  - non-atmospheric, non-seepage face
C                      Dirichlet BC's
C                      (see subroutines BCONE and BCNXT for
C                      unit IIN4 input) ``flow-nansfbc-dirich-time''
C        IN5   = 11  - non-atmospheric, non-seepage face
C                      Neumann BC's
C                      (see subroutines BCONE and BCNXT for
C                      unit IIN5 input) ``flow-nansfbc-neumann-time''
C        IN6   = 12  - atmospheric BC's (rainfall/evaporation rates)
C                      ``flow-atmbc''
C        IN7   = 13  - seepage face BC's ``flow-sfbc''
C        IN8   = 14  - soil characteristics ``flow-soil''
C        IN9   = 15  - initial conditions for flow  ``flow-ic''
C        IN10  = 16  - initial conditions for transp ``transp-ic''
C        IN11  = 17  - solute properties ``transp-solute''
C        IN12  = 18  - transport BC's  ``transp-bc''
C
C
C   TERM             - output unit number for the screen
C   DB<X>            - output debug unit number (1 <= X <=4)
C   OUT<X>           - output unit number (1 <= X <=26)
```

```
C
C List of Output Units:
C --------------------
C
C     TERM   = 4   - terminal output
C     OUT1   = 6   - main output
C     OUT2   = 25  - X, Y, Z coordinate values
C     OUT3   = 26  - flow: convergence behavior and error norms
C     OUT4   = 27  - mass balance and convergence behavior at each
C                    time step when solving the flow eqn.
C     OUT5   = 28  - convergence behavior at each nonlinear it. of
C                    each time step when solving the coupled syst. of
C                    eqns.
C     OUT6   = 29  - vertical profile output of pressure heads,
C                    water satur. (SW) and rel. hydr. conduct.(CKRW)
C     OUT7   = 30  - vertical profile output of concentrations
C     OUT8   = 31  - atm. and seep. face's hydrograph output
C     OUT9   = 32  - non-atm. non-seep. face's hydrograph output
C     OUT10  = 33  - detailed HGFLAG output
C     OUT11  = 134 - detailed SFFLAG output
C     OUT12  = 35  - pressure head output at all nodes
C     OUT13  = 36  - velocity output at all nodes
C     OUT14  = 37  - concentration output at all nodes
C     OUT15  = 38  - SW output at all nodes
C                    for input to DUAL3d and TRAN3d codes
C     OUT16  = 39  - CKRW output at all nodes
C     OUT17  = 40  - velocity output at all elements,
C                    for input to DUAL3d and TRAN3d codes
C     OUT18  = 41  - pressure head output at surface nodes
C     OUT19  = 42  - SATSUR (see description file) output at surface nodes
C     OUT20  = 43  - SW output at surface nodes
C     OUT21  = 44  - concentration output at surface nodes
C     OUT22  = 45  - non-atmospheric, non-seepage face Dirichlet BC's
C                    at each time step
C     OUT23  = 46  - non-atmospheric, non-seepage face Neumann BC's
C                    at each time step
C     OUT24  = 47  - detailed seepage face hydrograph output
C     OUT25  = 48  - detailed non-atmospheric, non-seepage face
C                    Dirichlet hydrograph output
C     OUT26  = 49  - detailed non-atmospheric, non-seepage face
C                     Neumann hydrograph output
C
C
C   IPRT   - flag for detailed output at all nodes and velocity and
C            water saturation output at all elements (velocity and
C            water saturation output in the case IPRT=4 can be used as
C            input to TRAN3D and DUAL3D codes)
C            =0 don't print nodal pressure, velocity, water saturation,
C               or relative conductivity values
C            =1 print only nodal pressure head values
C            =2 print nodal pressure head and velocity values
C            =3 print nodal pressure, velocity, and relative
C               conductivity values
C            =4 print nodal pressure, velocity, relative conductivity,
C               and overall storage coefficient values, and print
C               element velocity and nodal water saturation values
C   IPRT1  - flag for output of input and coordinate data
```

```
C            in subroutines DATIN and GEN3D
C            =0 prints parameters only (default)
C            =1 prints parameters + b.c. + geom. char.
C            =2 prints parameters + b.c. + geom. char. + grid info
C            =3 prints parameters + b.c. + geom. char. + grid info,
C               X, Y, Z coordinate values in subroutine GEN3D, and then
C               terminates program execution
C   KPRT   - index to current time value for detailed output
C   ISEC   - horizontal section number for which the potential head
C            and concentration values will be output if IPRT>=1.
C            If ISEC=0 then the output will be global, otherwise
C            only that section will be considered for the detailed ouput
C
C Integer Arrays:
C
C   CONTR (NR)          - node #'s for partial output
C   NODVP (NUMVP)       - node #'s for surface nodes selected for
C                         vertical profile output
C
C Real Arrays:
C
C   TIMPRT(NPRT)        - time values for detailed output. Detailed
C                         output is produced at initial conditions
C                         (TIME=0), at time values indicated in TIMPRT,
C                         and at the end of the simulation (TIME=TMAX).
C                         Detailed output consists of: values of pressure
C                         head, velocity, water saturation, and relative
C                         conductivity (depending on setting of IPRT)
C                         at all nodes; velocity, and water saturation
C                         (depending on setting of IPRT) at all elements;
C                         vertical profiles of pressure head, water
C                         saturation, and relative conductivity for the
C                         NODVP surface nodes; pressure head, water
C                         saturation, and SATSUR values at the
C                         surface nodes
C
C Character Strings:
C
C   IFN<X>          - input unit filename  (0 <= X <=12)
C   OFN<X>          - output unit filename (1 <= X <=26)
C   DBFN<X>         - output debug unit filename (1 <= X <=4)
C
C............................................................
C
C *Central Processing Unit (CPU) timing PARAMETERS*
C
C contained in the data structure ``CPU'' of derived type ``CPU_tag''
C  (file: mod_CPU.F):
C
C Real Parameters:
C
C   MN  - total cpu time for the simulation
C   NL  - total cpu time for nonlinear scheme
C   OVH - total cpu time for overhead:
C                       - data input, initialization, and output of
C                         initial conditions (once)
C                       - construction of tetrahedral elements (once)
```

```
C                      - volume calculations (once)
C                      - set up of storage indices and pointers (once)
C                      - velocity calculations (every time step for the
C                        case IPRT > 1)
C                      - hydrograph calculation (every time step)
C                      - input, interpolation, and switching control of
C                        atmospheric boundary conditions for the next
C                        time level (every time step)
C                      - update of pressure heads for the next time
C                        level (every time step)
C                      - back-stepping procedure (when needed)
C                      - final output (once)
C   LIN - total cpu time for assembly and solution of linear system
C   BAL - total cpu time for back-calculation of fluxes at
C            Dirichlet nodes, for mass balance calculation,
C            and for hydrograph calculation
C
C Real*4 Arrays:
C
C  VEC(9)          - cpu times for different sections of nonlinear schemes:
C                        (1) unsat characteristics
C                        (2) initialization of system matrices
C                        (3) assembly of local system components into
C                            global matrices
C                        (4) calculation of RHS without boundary cond.
C                        (5) construction of global LHS system matrix
C                        (6) calculation of BC contributions to RHS
C                        (7) linear solver and calculation of residual
C                        (8) extraction of pressure head solution
C                            from the difference solution and
C                            re-setting of solution and of COEF1 for
C                            Dirichlet nodes
C                        (9) back-calculation of fluxes at all Dirichlet
C                            nodes, mass balance calculation,
C                            application of nonlinear relaxation scheme
C                            (if required), calculation of nonlinear
C                            convergence and residual error norms,
C                            switching control of atmospheric boundary
C                            conditions, and calculation of new position
C                            of the exit point along each seepage face
C.......................................................................
C
C *GRID PARAMETERS & DATA*
C
C contained in the data structure ``Grid'' of derived type ``Grid_tag''
C (file: mod_Grid.F):
C
C
C Integer Parameters:
C
C   IVERT  - =0 each vertical layer will be parallel to the surface,
C               including the base of the 3-d grid. ZRATIO is applied to
C               each vertical cross section.
C            =1 base of the 3-d grid will be flat, and ZRATIO is applied
C               to each vertical cross section
C            =2 base of the 3-d grid will be flat, as will the NSTR-1
C               horizontal cross sections above it. ZRATIO is applied
```

```
C               only to the vertical cross section having the lowest
C               elevation
C   ISP    - =0 for flat surface layer (only one Z value is read in, and
C            is replicated to all surface nodes); otherwise surface
C            layer is not flat (Z values read in for each surface node)
C            (for ISP=0, IVERT=0, 1, and 2 yield the same 3-d mesh,
C            given the same values of BASE and ZRATIO)
C
C Real Parameters:
C
C   BASE   - value which defines the thickness or base of the 3-d mesh.
C            For IVERT=0, BASE is subtracted from each surface elevation
C            value, so that each vertical cross section will be of
C            thickness BASE, and the base of the 3-d mesh will be
C            parallel to the surface. For IVERT=1 or 2, BASE is
C            subtracted from the lowest surface elevation value, say
C            ZMIN, so that each vertical cross section will be of
C            thickness (Z - ZMIN) + BASE, where Z is the surface
C            elevation for that cross section. The base of the 3-d mesh
C            will thus be flat
C
C Integer Arrays:
C
C   TP    (N)          - # of elements connecting to each node
C   IVOL  (NT)         - sign of the volume of each element
C   TRIANG(4,NTRI)     - element connectivities in 2-d mesh (TRIANG(4,I)
C                        indicates material type for 2-d element I)
C   TETRA (5,NT)       - element connectivities in 3-d mesh (TETRA(5,I)
C                        indicates material type for 3-d element I)
C Real Arrays:
C
C   X     (N)          - x-coordinates (for 2-d mesh on input)
C   Y     (N)          - y-coordinates (for 2-d mesh on input)
C   Z     (N)          - z-coordinates (surface elevation values on
C                        input - see description of ISP)
C   VOLNOD(N)          - absolute value of volume assigned to each node
C   VOLU  (NT)         - absolute value of the volume of each element
C   VOLUR (NT)         - reciprocal of VOLU
C   ZRATIO(NSTR)       - fraction of total grid height that each layer
C                        is to occupy (see also description of IVERT).
C                        ZRATIO(1) is for the surface-most layer.
C                        ZRATIO values must sum to 1
C...........................................................
C
C *FLOW BC PARAMETERS & DATA*
C
C contained in the data structure ``Flow_BC'' of derived type ``Flow_BC_tag''
C (file: mod_Flow_BC.F):
C
C
C Integer Parameters:
C
C   NUMDIR - total # of Dirichlet nodes in 3-d mesh
C   ISFONE - =0 seepage face exit point updating performed by
C               checking all nodes on a seepage face
C            =1 seepage face exit point updating performed by
C               checking only the one node above and one node
```

```
C              below the current exit point position
C   ISFCVG - =0 convergence of seepage face exit points is not a
C              condition for convergence of the nonlinear iterative
C              procedure
C           =1 convergence of seepage face exit points is a condition
C              for convergence of the nonlinear iterative procedure
C   KSF    - number of seepage face exit points which did not converge
C           at each nonlinear iteration
C   KSFCV  - total number of seepage face exit point convergence failure
C           occurrences (over all nonlinear iterations and all time
C           steps)
C   KSFCVT - total number of seepage face exit point convergence
C           failures (over all seepage faces, all nonlinear iterations,
C           and all time steps)
C   HTIDIR - =0 for temporally variable non-atmospheric, non-seepage
C           face Dirichlet boundary conditions inputs; otherwise
C           non-atmospheric, non-seepage face Dirichlet boundary
C           conditions inputs are homogeneous in time (see also
C           notes following description of QPOLD)
C   HTINEU - =0 for temporally variable non-atmospheric, non-seepage
C           face Neumann boundary conditions inputs; otherwise
C           non-atmospheric, non-seepage face Neumann boundary
C           conditions inputs are homogeneous in time (see also
C           notes following description of QPOLD)
C   HSPATM - =0 for spatially variable atmospheric boundary condition
C           inputs; blank or =9999 if unit IIN6 input is to be ignored;
C           otherwise atmospheric BC's are homogeneous in space
C   HTIATM - =0 for temporally variable atmospheric boundary condition
C           inputs; otherwise atmospheric BC's are homogeneous in time
C           (see also notes following description of ATMINP)
C
C Integer Arrays:
C
C   SFFLAG(5)          - counter for anomalous, implausible, or
C                        erroneous occurrences along seepage faces
C                        (see output statements 2100,2200 in subroutine
C                        SFINIT, 2100,2200,2300,2400 in subroutines
C                        EXTONE, EXTALL, and 2500 in subroutine FLUXMB)
C   HGFLAG(8)          - counter for anomalous, implausible, or
C                        erroneous atmospheric inflow, outflow, and
C                        runoff occurrences (see subroutine HGRAPH)
C   CONTP2(NDIR)       - non-atmospheric, non-seepage face Dirichlet
C                        node #'s in 2-d mesh
C   CONTP (NP)         - non-atmospheric, non-seepage face Dirichlet
C                        node #'s in 3-d mesh
C   CONTQ (NQ)         - non-atmospheric, non-seepage face Neumann
C                        node #'s in 3-d mesh
C   NODDIR(NUMDIR)     - node #'s for all Dirichlet nodes in 3-d mesh
C   NSFNUM(NSF)        - # of nodes on each seepage face
C   SFEX  (NSF)        - the exit point on each seepage face. The
C                        seepage face nodes above the exit point are
C                        'potential' seepage face nodes, are treated as
C                        zero flux Neumann BC's, and the pressure heads
C                        here should be negative (unsaturated). The
C                        seepage face nodes below the exit point (and
C                        including the exit point) are 'actual' seepage
C                        face nodes, are treated as zero pressure
```

```
C                          head Dirichlet BC's (saturated), and the
C                          back-calculated fluxes here should be
C                          negative (outflow).
C                          The position of the exit point for the
C                          first time step is calculated from the initial
C                          conditions. The new position of the exit point
C                          is calculated after every nonlinear iteration
C                          of every time step, and the boundary conditions
C                          for the seepage face nodes are adjusted to
C                          reflect changes in the position of the exit
C                          point.
C                          For the case where seepage face I is
C                          completely saturated (all seepage face
C                          nodes are 'actual'), SFEX(I)=1. For the case
C                          where seepage face I is completely unsaturated
C                          (all seepage face nodes are 'potential' and
C                          there is no exit point), SFEX(I)=NSFNUM(I)+1.
C                          This convention simplifies the handling of
C                          seepage face nodes (relying on the fact that
C                          FORTRAN 77 does not execute a DO loop if the
C                          iteration count is zero or negative)
C   SFEXIT(NSF)          - SFEX values at previous nonlinear iteration
C   SFEXP (NSF)          - SFEX values at previous time level
C   IFATM (NNOD)         - IFATM(I)=0  if surface node I is a Neumann
C                          atmospheric boundary condition node
C                          IFATM(I)=1  if surface node I is a Dirichlet
C                          atmospheric boundary condition node
C                          IFATM(I)=-1 if surface node I is not an
C                          atmospheric boundary condition node
C                          Note: surface nodes are numbered 1,...,NNOD in
C                          the 3-d mesh, so there is no need for a pointer
C                          array giving the node #'s for the surface nodes
C   IFATMP(NNOD)         - IFATM values at previous time level
C   SATSUR(NNOD)         - SATSUR(I)=1  if surface node I is unsaturated
C                          SATSUR(I)=2  if surface node I is Horton
C                          saturated (infiltration excess mechanism)
C                          SATSUR(I)=3  if surface node I is Dunne
C                          saturated (saturation excess mechanism)
C   NSFNOD(NSF,NNSFMX) - node #'s on each seepage face. The node #'s for
C                          each seepage face must be input in descending
C                          order by elevation. That is, along seepage face
C                          I, Z(NSFNOD(I,J)) .GE. Z(NSFNOD(I,J+1)) must
C                          hold for J=1,...,NSFNUM(I)-1. Seepage faces can
C                          be defined, for instance, above a well, along a
C                          stream bank, or along a combination of stream
C                          bank and surface nodes. For a configuration
C                          of seepage face and stream, the stream nodes
C                          should be designated as non-atmospheric,
C                          non-seepage face nodes, for instance as
C                          Dirichlet nodes with a pressure head
C                          distribution in hydrostatic equilibrium, the
C                          node at the surface of the stream being
C                          assigned a pressure head value of zero.
C                          For output purposes, we set
C                          NSFNOD(I,NSFNUM(I)+1)=-9999
C
C Real Arrays:
```

```
C
C   ATMTIM(3)          - most current input time values for atmospheric
C                        BC's, with ATMTIM(1) < ATMTIM(2) < ATMTIM(3)
C                        and ATMTIM(2) < TIME <= ATMTIM(3)
C   PTIM(3)            - most current input time values for
C                        non-atmospheric, non-seepage face Dirichlet
C                        BC's, with PTIM(1) < PTIM(2) < PTIM(3)
C                        and PTIM(2) < TIME <= PTIM(3)
C   QTIM(3)            - most current input time values for
C                        non-atmospheric, non-seepage face Neumann
C                        BC's, with QTIM(1) < QTIM(2) < QTIM(3)
C                        and QTIM(2) < TIME <= QTIM(3)
C   PRESC (NP)         - non-atmospheric, non-seepage face Dirichlet
C                        values at current time level
C   Q     (NQ)         - non-atmospheric, non-seepage face Neumann
C                        values at current time level
C   QPNEW (NP)         - back-calculated flux values at non-atmospheric,
C                        non-seepage face Dirichlet nodes at current
C                        time level
C   QPOLD (NP)         - QPNEW values at previous time level
C   Notes: (a) For a simulation using temporally homogeneous
C              non-atmospheric, non-seepage face Dirichlet
C              (Neumann) BC's, input data on unit IIN8 (IIN9) should
C              contain a single value of PTIM (QTIM) (0.0)
C              and a single set of PINP (QINP) data.
C              Alternatively, to properly handle the case where the
C              datasets for different simulations are kept in the same
C              file (separated by blank lines), the input data on unit
C              IIN8 (IIN9) for temporally homogeneous
C              non-atmospheric, non-seepage face Dirichlet
C              (Neumann) BC's should contain, as above, a value of PTIM
C              (QTIM) of 0.0 followed by the PINP
C              (QINP) values, and then a value of PTIM
C              (QINP) equal to or larger than TMAX
C              (1.0e+10, say) followed by the same PINP (QINP)
C              values specified at time 0.0.
C          (b) If the first input time value is greater than 0.0, we set
C              the initial (time 0.0) non-atmospheric, non-seepage face
C              Dirichlet (Neumann) BC inputs to 0.0
C          (c) If TIME is larger than the last PTIM (QTIM)
C              value on unit IIN8 (IIN9), HTIDIR (HTINEU)
C              is set to 1 and the last input values are used for the
C              rest of the simulation. To properly handle the case where
C              the datasets for different simulations are kept in the
C              same file (separated by blank lines), follow the
C              procedure described in (a)
C   ATMPOT(NNOD)       - precipitation (+ve) / evaporation (-ve) fluxes
C                        at current time level for each surface node.
C                        These are potential infiltration/exfiltration
C                        values.
C   ATMACT(NNOD)       - actual fluxes (infiltration/exfiltration
C                        values) for atmospheric boundary
C                        condition nodes at current time level.
C                        For IFATM(I)=0,  ATMACT(I) = ATMPOT(I);
C                        For IFATM(I)=1,  ATMACT(I) = back-calculated
C                        flux value;
C                        For IFATM(I)=-1, ATMACT(I) is disregarded
```

```
C   ATMOLD(NNOD)       - ATMACT values at previous time level
C   ATMINP(3,NNOD)     - input atmospheric rainfall/evaporation rates
C                        corresponding to ATMTIM times. ATMPOT(I) is
C                        obtained from ATMINP(2,I) and ATMINP(3,I) by
C                        linear interpolation and conversion of rate
C                        to volumetric flux. ATMINP(1,I) values are
C                        needed in the event that, after back-stepping,
C                        we have ATMTIM(1) < TIME <= ATMTIM(2)
C   Notes: (a) For a simulation using temporally homogeneous atmospheric
C              rates, input data on unit IIN6 should contain a single
C              value of ATMTIM (0.0) and a single set of ATMINP data.
C              Alternatively, to properly handle the case where the
C              datasets for different simulations are kept in the same
C              file (separated by blank lines), the input data on
C              unit IIN6 for temporally homogeneous rates should
C              contain, as above, a value of ATMTIM of 0.0 followed by
C              the ATMINP rates, and then a value of ATMTIM equal to or
C              larger than TMAX (1.0e+10, say) followed by the same
C              ATMINP rates specified at time 0.0.
C          (b) If there is no ATMTIM, ATMINP input, HTIATM is set
C              to 1 (homogeneous in time) and atmospheric input rates
C              are set to 0.0.
C          (c) If the first input time value is greater than 0.0, we set
C              the initial (time 0.0) atmospheric input rates to 0.0.
C          (d) If TIME is larger than the last ATMTIM value on unit
C              IIN6, HTIATM is set to 1 and the last input atmospheric
C              rates are used for the rest of the simulation. To
C              properly handle the case where the datasets for different
C              simulations are kept in the same file (separated by blank
C              lines), follow the procedure described in (a).
C          (e) If HSPATM is nonzero and not equal to 9999 (spatially
C              homogeneous), each set of ATMINP data should consist of
C              a single value which gets copied to all surface nodes.
C              If HSPATM is zero (spatially variable), each set of
C              ATMINP data should consist of NNOD values (note that
C              we read in a value for each surface node, including
C              surface nodes which may be designated as non-atmospheric
C              Dirichlet or Neumann boundary conditions. IFATM controls
C              whether the atmospheric input for a given surface node
C              is actually used)
C   PINP(3,NP)         - non-atmospheric, non-seepage face Dirichlet
C                        values corresponding to PTIM times.
C                        PRESC(I) is obtained from PINP(2,I) and
C                        PINP(3,I) by linear interpolation.
C                        PINP(1,I) values are needed in the event that,
C                        after back-stepping, we have
C                        PTIM(1) < TIME <= PTIM(2)
C   QINP(3,NP)         - non-atmospheric, non-seepage face Neumann
C                        values corresponding to QTIM times.
C                        Q(I) is obtained from QINP(2,I) and
C                        QINP(3,I) by linear interpolation.
C                        QINP(1,I) values are needed in the event that,
C                        after back-stepping, we have
C                        QTIM(1) < TIME <= QTIM(2)
C   SFQ   (NSF,NNSFMX) - back-calculated flux values at actual seepage
C                        face nodes at current time level
C   SFQP  (NSF,NNSFMX) - SFQ values at previous time level
```

```
C
C Logical Flags:
C
C   SFCHEK - flag indicating whether it is necessary to check for
C            seepage face exit point convergence as a condition for
C            convergence of the nonlinear iterative procedure
C             FALSE if it is not necessary to check
C             TRUE  if it is necessary to check
C   KSFZER - flag for number of seepage face exit points which did
C            not converge at each nonlinear iteration
C             FALSE if one or more exit points did not converge
C             TRUE  if all exit points converged (i.e. KSF=0)
C.......................................................................
C
C *TRANSPORT BC PARAMETERS & DATA*
C
C contained in the data structure ``Transp_BC'' of derived type
C ``Transp_BC_tag'' (file: mod_Transp_BC.F):
C
C
C Integer Arrays:
C
C   NNP2C (NP2C)       - Dirichlet node #'s in 2-d mesh for the
C                        transport equation
C   NNPC  (NPC)        - Dirichlet node #'s in 3-d mesh for the
C                        transport equation
C   NNMC  (NMC)        - Cauchy node #'s in 3-d mesh for the transport
C                        equation
C
C Real Arrays:
C
C   PC    (NPC)        - Dirichlet values for the transport equation
C                        (constant in time)
C   MC1   (NMC)        - total Cauchy values for the transport equation
C   MC2   (NMC)        - advective component of the Cauchy values for
C                        the transport equation (if the advective
C                        component of the Cauchy boundary conditions is
C                        zero, the Cauchy BC's become Neumann boundary
C                        conditions)
C                        (the Cauchy BC's MC1 and MC2 are constant in
C                        time for TIME > 0; for time 0 Cauchy
C                        values are taken to be zero. This is done to
C                        avoid oscillations in the Crank-Nicolson
C                        scheme.)
C   BKTNEW(NPC+NMC)    - back-calculated flux values at Dirichlet and
C                        Cauchy nodes for the transport equation at
C                        current time level
C   BKTOLD(NPC+NMC)    - BKTNEW values at previous time level
C.......................................................................
C
C *MASS BALANCE PARAMETERS & DATA*
C
C contained in the data structure ``MBal'' of derived type
C ``MBal_tag'' (file: mod_MBal.F):
C
C
C Real Parameters:
```

```
C
C   --Abbreviations: na  = non-atmospheric
C                    nsf = non-seepage face
C   NDIN   - tot inflow  flux from na, nsf Dir nodes at curr time level
C   NDOUT  - tot outflow flux from na, nsf Dir nodes at curr time level
C   NDINP  - tot inflow  flux from na, nsf Dir nodes at prev time level
C   NDOUTP - tot outflow flux from na, nsf Dir nodes at prev time level
C   NNIN   - tot inflow  flux from na, nsf Neu nodes at curr time level
C   NNOUT  - tot outflow flux from na, nsf Neu nodes at curr time level
C   NNINP  - tot inflow  flux from na, nsf Neu nodes at prev time level
C   NNOUTP - tot outflow flux from na, nsf Neu nodes at prev time level
C   VNDIN  - tot inflow  volu from na, nsf Dir nodes over curr time step
C   VNDOUT - tot outflow volu from na, nsf Dir nodes over curr time step
C   VNNIN  - tot inflow  volu from na, nsf Neu nodes over curr time step
C   VNNOUT - tot outflow volu from na, nsf Neu nodes over curr time step
C   VIN    - VADIN  + VNDIN  + VANIN  + VNNIN  = total inflow  volume
C            between current and previous time levels (> 0)
C   VOUT   - VADOUT + VNDOUT + VANOUT + VNNOUT + VSFFLW = total outflow
C            volume between current and previous time levels (< 0)
C   DSTORE - total volume of storage change between current and
C            previous time levels (> 0 for net increase in storage)
C   ERRAS  - absolute volume ("mass") balance error over the current
C            time step
C   ERREL  - relative (percent) mass balance error over the current
C            time step
C   ETOT   - cumulative (over all time steps) absolute mass balance
C            error ERRAS
C   VTOT   - cumulative (over all time steps) total net volume
C            VIN + VOUT
C   VTOTI  - cumulative (over all time steps) total VIN
C   VTOTO  - cumulative (over all time steps) total VOUT
C   TDIN   - tot mass flux in  from Dirichlet nodes at curr time level
C   TDOUT  - tot mass flux out from Dirichlet nodes at curr time level
C   TDINP  - tot mass flux in  from Dirichlet nodes at prev time level
C   TDOUTP - tot mass flux out from Dirichlet nodes at prev time level
C   TCIN   - tot mass flux in  from Cauchy    nodes at curr time level
C   TCOUT  - tot mass flux out from Cauchy    nodes at curr time level
C   TCINP  - tot mass flux in  from Cauchy    nodes at prev time level
C   TCOUTP - tot mass flux out from Cauchy    nodes at prev time level
C            (above fluxes are mass fluxes (volumetric flux * density)
C            of solute computed from the transport equation)
C   MTDIN  - tot solute mass in  from Dirich nodes over curr time step
C   MTDOUT - tot solute mass out from Dirich nodes over curr time step
C   MTCIN  - tot solute mass in  from Cauchy nodes over curr time step
C   MTCOUT - tot solute mass out from Cauchy nodes over curr time step
C   TMIN   - MTDIN  + MTCIN  = total solute mass into the porous
C            medium between current and previous time levels (> 0)
C   TMOUT  - MTDOUT + MTCOUT = total solute mass out of the porous
C            medium between current and previous time levels (< 0)
C   MASPOR - total mass of solute change between current and
C            previous time levels (> 0 for net increase in solute mass)
C            for the transport equation. MASPOR is computed from the
C            (porosity * retardation factor) coefficient in the time
C            derivative of concentration term in the transport equation.
C            Note that we neglect the contributions of the specific
C            storage and the (density ratio * EPSLON * concentration)
C            terms in computing MASPOR. Indeed these terms are not
```

```
C           considered at all in the transport equation being solved
C           in this code, and thus are neglected also in the assembly
C           of the system matrices. (See pg 22 of the ENEL code manual,
C           "Parte 1: Modello Matematico", G. Gambolati & G. Pini,
C           Padova, December 1989)
C   ERRAST - absolute solute mass balance error over the
C           current time step for the transport equation
C   ERRELT - relative (percent) solute mass balance error over the
C           current time step for the transport equation
C   ETOTT  - cumulative (over all time steps) absolute mass balance
C           error ERRAST for the transport equation
C   MTOTT  - cumulative (over all time steps) total net solute mass
C           TMIN + TMOUT
C   MTOTTI - cumulative (over all time steps) total TMIN
C   MTOTTO - cumulative (over all time steps) total TMOUT
C   OVFLOW - total overland flow (surface runoff) flux produced at
C           atmospheric surface nodes. Overland flow occurs during
C           rainfall periods when the actual flux is less than the
C           potential flux, and accounts for both Horton and Dunne
C           saturation mechanisms.
C   REFLOW - total return flow flux produced at atmospheric surface
C           nodes. Return flow occurs during rainfall periods when
C           the actual flux is negative (outflow rather than inflow).
C           In this case all of the potential flux becomes overland
C           flow, and the magnitude of the actual flux becomes the
C           return flow component of surface runoff
C   SFFLW  - total subsurface flow flux produced at seepage faces
C           at the current time level
C   SFFLWP - total subsurface flow flux produced at seepage faces
C           at the previous time level
C   VSFFLW - total subsurface flow volume produced at seepage faces
C           between current and previous time levels
C   APOT   - total atmospheric potential flux at the current time level,
C           used for hydrograph output. Note that we disregard
C           contribution of non-atmospheric, non-seepage face
C           surface nodes in the calculation of APOT.
C   AACT   - total atmospheric actual flux at the current time level,
C           used for hydrograph output. AACT=ADIN+ADOUT+ANIN+ANOUT.
C   ADIN   - tot inflow  flux from atmosph Dir nodes at curr time level
C   ADOUT  - tot outflow flux from atmosph Dir nodes at curr time level
C   ADINP  - tot inflow  flux from atmosph Dir nodes at prev time level
C   ADOUTP - tot outflow flux from atmosph Dir nodes at prev time level
C   ANIN   - tot inflow  flux from atmosph Neu nodes at curr time level
C   ANOUT  - tot outflow flux from atmosph Neu nodes at curr time level
C   ANINP  - tot inflow  flux from atmosph Neu nodes at prev time level
C   ANOUTP - tot outflow flux from atmosph Neu nodes at prev time level
C   VADIN  - tot inflow  volu from atmosph Dir nodes over curr time step
C   VADOUT - tot outflow volu from atmosph Dir nodes over curr time step
C   VNDIN  - tot inflow  volu from na, nsf Dir nodes over curr time step
C   VNDOUT - tot outflow volu from na, nsf Dir nodes over curr time step
C...................................................................
C
C *MATERIAL and SOLUTE PARAMETERS & DATA*
C
C contained in the data structure ``MS_prop'' of derived type
C ``MS_prop_tag'' (file: mod_MS_prop.F):
C
```

```
C Integer Parameters:
C
C   IVGHU  - =0 for van Genuchten moisture curves
C            =1 for extended van Genuchten moisture curves
C            =2 for moisture curves from Huyakorn et al (WRR 20(8) 1984,
C               WRR 22(13) 1986) with Kr=Se**n conductivity relationship
C            =3 for moisture curves from Huyakorn et al (WRR 20(8) 1984,
C               WRR 22(13) 1986) with conductivity relationship from
C               Table 3 of 1984 paper (log_10 Kr(Se) curve)
C            =4 for Brooks-Corey moisture curves
C   KSLOPE - =0 for analytical differentiation of moisture curves
C            =1 for "chord slope" and analytical differentiation
C            =2 for "chord slope" and centered difference formulas
C            =3 for localized "chord slope" and analytical
C               differentiation
C            =4 for localized "tangent slope" differentiation
C            (the "chord slope" formula is the tangent approximation
C            suggested by Huyakorn et al (WRR 20(8) 1984), wherein
C            derivatives are approximated using pressure heads at
C            the current and previous nonlinear iterations; "tangent
C            slope" differentiation is a different tangent approximation
C            wherein derivatives are approximated using pressure heads
C            at the endpoints of a given range (eg: endpoints PKRL, PKRR
C            for the derivative of relative hydraulic conductivity). For
C            KSLOPE=1,2 the chord slope formula is used at every
C            iteration and at all nodes (with some exceptions as
C            dictated by TOLKSL). For KSLOPE=3 or 4 the chord or tangent
C            slope formulas are used only at those nodes whose pressure
C            heads fall within given ranges (see PKRL, PKRR, etc), hence
C            'localized'; for nodes whose pressure heads fall outside
C            these ranges, analytical differentiation is used.)
C
C Real Parameters:
C
C   PKRL,  - left and right endpoints of the pressure head range within
C   PKRR     which the chord slope (case KSLOPE=3) or tangent slope
C            (case KSLOPE=4) formula is used to evaluate the derivative
C            of relative hydraulic conductivity
C   PSEL,  - left and right endpoints of the pressure head range within
C   PSER     which the chord slope (case KSLOPE=3) or tangent slope
C            (case KSLOPE=4) formula is used to evaluate the derivative
C            of effective saturation (moisture content for the case
C            of extended van Genuchten curves, IVGHU=1)
C   PDSE1L,- left and right endpoints of the two pressure head ranges
C   PDSE1R,  within which the chord slope (case KSLOPE=3) or tangent
C   PDSE2L,  slope (case KSLOPE=4) formula is used to evaluate the
C   PDSE2R   second derivative of effective saturation (moisture content
C            for the case of extended van Genuchten curves, IVGHU=1).
C            (Two ranges are specified since in general d(Se)/dP is
C            non-monotonic.)
C   DKRTAN - tangent slope approximation of d(Kr)/dP, the derivative of
C            relative hydraulic conductivity Kr wrt to pressure head P.
C            i.e. DKRTAN = (Kr(PKRR) - Kr(PKRL))/(PKRR - PKRL)
C   TOLKSL - tolerance for chord slope formula. Whenever the chord slope
C            formula is to be applied (for KSLOPE=1 or 2 at every iter-
C            ation and at all nodes; for KSLOPE=3 at those nodes whose
C            pressure heads fall within given ranges), it is applied
```

```
C           only if the absolute pressure head difference (between the
C           current and previous nonlinear iterations) is larger than
C           TOLKSL. If the difference is smaller than TOLKSL, then
C           differentiation is done either analytically (KSLOPE=1,3) or
C           with a centered difference formula (KSLOPE=2)
C   RHO0   - density of fresh water
C           Note: density of water in solution, RHO, is defined
C           as RHO = RHO0 * (1 + EPSLON * CNEW), which gives
C           RHO0 / RHO = 1 / (1 + EPSLON * CNEW). We use this latter
C           formula, and hence RHO is never explicitly used in this
C           code.)
C   EPSLON - density difference ratio
C           (EPSLON = (RHOMAX - RHO0) / RHO0, where RHOMAX is the
C           maximum density of water in solution)
C   DIFFUS - molecular diffusion coefficient
C   PMIN   - 'air dry' pressure head value (for switching control of
C           atmospheric boundary conditions during evaporation)
C
C Real Arrays:
C
C   SNODI (N)          - specific storage at each node
C   PNODI (N)          - porosity at each node
C   RNODI (N)          - retardation factor at each node
C
C   ETAI  (N)          - overall storage coefficient (general storage
C                        term) at each node
C   DETAI (N)          - derivative of ETAI wrt press. head at each node
C   CKRW  (N)          - relative hydraulic conductivity at each node
C   SW    (N)          - water saturation (moisture content/porosity)
C                        at each node
C   DSETAN(N)          - tangent slope approximation of d(Se)/dP, the
C                        derivative of effective saturation Se (moisture
C                        content for case IVGHU=1) wrt to pressure
C                        head P.
C                        i.e. DSETAN = (Se(PSER) - Se(PSEL)) /
C                                           (PSER - PSEL)
C   DDSE1T(N)          - tangent slope approximations of dd(Se)/dPP, the
C   DDSE2T(N)            second derivative of effective saturation Se
C                        (moisture content for case IVGHU=1) wrt to
C                        pressure head P.
C                        i.e. DDSE1T = (DSe(PDSE1R) - DSe(PDSE1L)) /
C                                            (PDSE1R - PDSE1L)
C                             DDSE2T = (DSe(PDSE2R) - DSe(PDSE2L)) /
C                                            (PDSE2R - PDSE2L)
C                        where DSe is the derivative of Se.
C                        (DSETAN, DDSE1T, and DDSE2T contain tangent
C                        slope values at each node only for the case
C                        IVGHU=1; for the other IVGHU cases the tangent
C                        slope values are constant for all nodes and are
C                        stored in DSETAN(1), DDSE1T(1), and DDSE2T(1).)
C   ETAE  (NT)         - overall storage coefficient for each element
C   CKRWE (NT)         - relative hydraulic conductivity for each
C                        element
C   SWE   (NT)         - water saturation (moisture content/porosity)
C   PERMX (NSTR,NZONE) - saturated hydraulic conductivity-xx
C   PERMY (NSTR,NZONE) - saturated hydraulic conductivity-yy
C   PERMZ (NSTR,NZONE) - saturated hydraulic conductivity-zz
```

```
C   ALFAL (NSTR,NZONE) - longitudinal dispersivity
C   ALFAT (NSTR,NZONE) - transverse dispersivity
C   ELSTOR(NSTR,NZONE) - specific storage
C   POROS (NSTR,NZONE) - porosity (moisture content at saturation)
C                        at each element
C   RETARD(NSTR,NZONE) - retardation factor
C
C Derived data type VG (Van Genutchen curve coefficients) in file
C       mod_VG.F:
C
C
C Real Parameters:
C
C   N,   - parameters for van Genuchten and extended van Genuchten
C   M,     moisture curves (other 'VG' parameters - specific storage,
C   RMC,   porosity, and VGPNOT - are assigned nodally). VGM is
C   PSAT   derived from VGN. VGRMC is residual moisture content.
C            For IVGHU=0, VGPNOT is (porosity - VGRMC)/porosity,
C            or (1 - residual water saturation).
C            For IVGHU=1, VGPNOT is a continuity parameter, derived by
C            imposing a continuity requirement on the derivative of
C            moisture content with respect to pressure head
C
C Real Arrays:
C
C   PNOT(N)          - (porosity - VGRMC)/porosity for van Genuchten
C                        curves (IVGHU=0); continuity parameter 'PNOT'
C                        for extended van Genuchten curves (IVGHU=1)
C
C Derived data type HU (Huyakorn curve coefficients) in file
C       mod_HU.F:
C
C Real Parameters:
C
C   N,   - parameters for moisture curves from
C   A,     Huyakorn et al (WRR 20(8) 1984, WRR 22(13) 1986)
C   B,     (other 'HU' parameters - specific storage
C   ALFA,  and porosity - are assigned nodally). HUN is
C   BETA,  only used for IVGHU=2; HUA and HUB are only used
C   GAMA,  for IVGHU=3. HUSWR is residual water saturation, which
C   PSIA,  is equivalent to residual moisture content/porosity.
C   SWR
C
C Derived data type BC (Brooks-Corey curve coefficients) in file
C       mod_BC.F:
C
C
C Real Parameters:
C
C   BETA,- parameters for Brooks-Corey moisture curves (other 'BC'
C   RMC,   parameters - specific storage and porosity - are assigned
C   PSAT   nodally). BCRMC is residual moisture content.
C
C Real Arrays:
C
C   PORM(N)          - (porosity - BCRMC)/porosity for Brooks-Corey
C                        curves (IVGHU=4)
```

```
C.....................................................................
C
C *LINEAR SYSTEM PARAMETERS & DATA*
C
C contained in the data structure ``Sys'' of derived type
C ``Sys_tag'' (file: mod_Sys.F):
C
C Integer Parameters:
C
C   NDZ    - # of zero elements on the diagonal of the system matrices
C            (signals an error condition)
C
C Integer Arrays:
C
C   IA    (NTERM)      - row indices in storage of system matrices for
C                        unsymmetric case
C   IAC   (NTERMC)     - row indices in storage of system matrices for
C                        the transport equation
C   TOPOL (N+1)        - pointer to first nonzero element of each row
C                        which is stored in the system matrices (the
C                        diagonal entry in symmetric storage case)
C   TOPOLC(N+1)        - pointer to first nonzero element of each row
C                        which is stored in the system matrices for the
C                        transport equation
C   JA    (N1*N)       - column indices (in ascending order) in storage
C                        of system matrices
C   JAC   (N1*N)       - column indices (in ascending order) in storage
C                        of system matrices for the transport equation
C   TETJA (4,4,NT)     - gives the index within JA (global position) of
C                        each component of the 4 x 4 local system
C                        matrices (upper triangle of 4 x 4 arrays only
C                        in this case since the system is symmetric)
C   TETJAC(4,4,NT)     - gives the index within JAC (global position)
C                        of each component of the 4 x 4 local system
C                        matrices
C
C Real Arrays:
C
C   INSYM (INONSY*(6*  - integer scratch vector for NONSYM solver
C          N + IBOT + 1) + 1)
C   RNSYM (INONSY*IBOT - real scratch vector for NONSYM solver
C          + 1)
C
C   COEF1 (NTERM)      - global stiffness matrix; also used to store
C                        the LHS system matrix, which is the Jacobian
C                        in the Newton case
C   COEF2 (NTERM)      - global mass matrix
C   COEF3 (NTERM)      - derivative term components of the Jacobian for
C                        Newton scheme; also used as a scratch vector
C   COEF1C(NTERMC)     - global stiffness matrix for the transport
C                        equation (ITRANS=1) or Jacobian matrix for
C                        the Newton transport equation (ITRANS=3);
C                        also used to store the LHS system matrix
C   COEF2C(NTERMC)     - global mass matrix for the transport equation
C   SCR1  (NTERMC)     - scratch vector
C   SCR1  (NTERMC)     - scratch vector
C   TNOTI (N)          - RHS system vector for the flow equation
```

```
C   TNOTIC(N)          - RHS system vector for the transport equation
C   XT5   (N)          - TNOTI before imposition of Dirichlet boundary
C                        conditions (needed for back-calculation of
C                        fluxes used in mass balance calculations)
C   XT5C  (N)          - TNOTIC before imposition of Dirichlet boundary
C                        conditions (needed for back-calculation of
C                        fluxes used in mass balance calculations)
C   LHSP  (NP)         - values of diagonal elements of LHS system
C                        matrix for the flow equation corresponding
C                        to Dirichlet nodes before imposition of
C                        Dirichlet BC's (needed for back-calculation of
C                        fluxes used in mass balance calculations)
C   LHSC  (NPC)        - values of diagonal elements of LHS system
C                        matrix for the transport equation corresponding
C                        to Dirichlet nodes before imposition of
C                        Dirichlet BC's (needed for back-calculation of
C                        fluxes used in mass balance calculations)
C   LHSATM(NNOD)       - values of diagonal elements of LHS system
C                        matrix corresponding to atmospheric Dirichlet
C                        nodes before imposition of Dirichlet
C                        BC's (needed for back-calculation of fluxes
C                        used in mass balance calculations, and for
C                        switching control of atmospheric BC's)
C   LMASS (4,4)        - local mass matrix for the flow equation,
C                        without the specific storage term and
C                        without the volume term
C   LMASSC(4,4)        - local mass matrix for the transport equation,
C                        without the porosity term, retardation factor,
C                        and volume term
C   BI    (4,NT)       - coefficients 'b-i / 6' of the basis functions
C   CI    (4,NT)       - coefficients 'c-i / 6' of the basis functions
C   DI    (4,NT)       - coefficients 'd-i / 6' of the basis functions
C   LHSSF (NSF,NNSFMX) - values of diagonal elements of LHS system
C                        matrix corresponding to seepage face Dirichlet
C                        nodes before imposition of Dirichlet
C                        BC's (needed for back-calculation of fluxes
C                        used in mass balance calculations, and for
C                        calculation of new position of the exit
C                        point along each seepage face)
C...............................................................
C
C *MODEL OUTPUT PARAMETERS & DATA*
C
C contained in the data structure ``Out'' of derived type
C ``Out_tag'' (file: mod_Out.F):
C
C Integer Parameters:
C
C   CNODE  - node in which there is the maximum norm of the
C            concentration
C   IKMAX  - node with largest pressure head difference in absolute
C            value between current and previous nonlinear iterations
C   IKMAXC - node with largest concentration difference in absolute
C            value between current and previous nonlinear iterations
C   ISURMX -
C   ISURMN -
C
```

```
C Integer Arrays:
C
C   IKMAXV(ITMX)       - for each nonlinear iteration in solving the
C                        flow eqn., gives the
C                        node number at which the largest solution
C                        difference, in normalized pressure head
C                        between the current and previous nonlinear
C                        iterations was found
C   IKMAXVC(ITMXC)     - for each nonlinear iteration in solving the
C                        coupled flow and transport system, gives the
C                        node number at which the largest solution
C                        difference, in either normalized potential head
C                        or in concentration (already normalized - see
C                        description of CNEW), between the current and
C                        previous nonlinear iterations was found
C
C Real Parameters:
C
C   PINF   - absolute value of pressure head difference at node IKMAX
C            (i.e. infinity norm of the convergence error), used in
C            comparison with TOLUNS for convergence test in the case
C            L2NORM=0
C   PL2    - square root of the sum of squares of pressure head
C            differences over all nodes (i.e. L2 norm of the convergence
C            error), used in comparison with TOLUNS for convergence test
C            in the case L2NORM nonzero
C   FINF   - residual error in the nonlinear solution calculated using
C            the infinity norm (for the nonlinear system f(x)=0, the
C            residual error at iteration "m" is the norm of f(x$^m$))
C   FL2    - residual error in the nonlinear solution calculated using
C            the L2 norm
C   DIFPSI - pressure head difference at the
C            node given in IKMAX
C
C Real Arrays:
C
C   UU    (NT)         - velocity-x for each element
C   VV    (NT)         - velocity-y for each element
C   WW    (NT)         - velocity-z for each element
C   UNOD  (N)          - velocity-x at each node
C   VNOD  (N)          - velocity-y at each node
C   WNOD  (N)          - velocity-z at each node
C   PNEW  (N)          - potential heads at current time level (and at
C                        current nonlinear iteration when solving the
C                        coupled flow and transport system)
C   CNEW  (N)          - solute concentrations at current time level,
C                        current nonlinear iteration
C                        Note: concentrations referred to, input, and
C                        used in this code are normalized, with
C                        0 <= conc <= 1, since the transport equation
C                        modeled in this code is written using relative
C                        (dimensionless) concentrations
C   PSINEW(N)          - pressure heads at current time level, current
C                        nonlinear iteration
C   POLD  (N)          - potential heads at previous nonlinear iteration
C   COLD  (N)          - concentrations at previous nonlinear iteration
C   PSIOLD(N)          - pressure heads at previous nonlinear iteration
```

```
C   PTNEW (N)          - weighted potential heads at current nonlinear
C                        iteration (using weighting parameter TETAC)
C   CTNEW (N)          - weighted concentrations at current nonlinear
C                        iteration (using weighting parameter TETAC)
C   PSITNEW(N)         - weighted pressure heads at current nonlinear
C                        iteration
C   PTOLD (N)          - weighted potential heads at previous
C                        nonlinear iteration
C   PSITOLD (N)        - weighted pressure heads at previous nonlinear
C                        iteration
C   PTIMEP(N)          - potential heads at previous time level (initial
C                        conditions on input)
C   CTIMEP(N)          - concentrations at previous time level (initial
C                        conditions on input)
C   PDIFF (N)          - difference in potential heads between nonlinear
C                        iterations
C   CDIFF (N)          - difference in cincentrations between nonlinear
C                        iterations
C   PSIDIFF (N)        - difference in pressure heads between nonlinear
C                        iterations
C   PL2V    (ITMX)     - PL2 values for each nonlinear flow iteration
C   FINFV   (ITMX)     - FINF values for each nonlinear flow iteration
C   FL2V    (ITMX)     - FL2 values for each nonlinear flow iteration
C   DIFPSIV (ITMX)     - normalized pressure head difference at the
C                        node given in IKMAXV for each nonlinear
C                        iteration
C   PSIMAXNV(ITMX)     - current iteration pressure head value used in
C                        calculation of DIFFPSIV
C   PSIMAXOV(ITMX)     - previous iteration pressure head value used in
C                        calculation of DIFFPSIV
C   DIFFPV  (ITMXC)    - normalized potential head difference at the
C                        node given in IKMAXV for each nonlinear
C                        iteration
C   PMAXNV  (ITMXC)    - current iteration potential head value used in
C                        calculation of DIFFPV
C   PMAXOV  (ITMXC)    - previous iteration potential head value used in
C                        calculation of DIFFPV
C   DIFPMX  (ITMXC)    - the largest potential head difference between
C                        the current and previous nonlinear iterations
C   DIFFCV  (ITMXC)    - concentration difference at the node given in
C                        IKMAXV for each nonlinear iteration
C   CMAXNV  (ITMXC)    - current iteration concentration value used in
C                        calculation of DIFFCV
C   CMAXOV  (ITMXC)    - previous iteration concentration value used in
C                        calculation of DIFFCV
C   DIFCMX  (ITMXC)    - the largest concentration difference between
C                        the current and previous nonlinear iterations
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
```

C Appendix C

The appendix reports the whole input dataset for the Henry test-case. It also contains the screen output of the CODESA-3D run on a Silicon Graphics (SGI) computer equipped with RISC10000 processor, 512 Mbyte of RAM and running IRIX/6.4 operating system (*filuferru.crs4.it*). In what follows, refer to Appendix B for the meaning of variables.

C.1 List of I/O files: the `codesa3d.fnames` file

The whole list of input and output file names is specified in the input file `codesa3d.fnames` which is read at the beginning of the code run.

```
c-----------------------------------------------------------------------
parm                                          unit 5  ! << input files>>
grid                                          unit 8
nansfbcnod.flow                               unit 9
nansfdirbc.flow                               unit 10
nansfneubc.flow                               unit 11
atmbc.flow-none                               unit 12
sfbc.flow-none                                unit 13
soil                                          unit 14
ic.flow                                       unit 15
ic.tran                                       unit 16
solute                                        unit 17
bc.tran                                       unit 18
result.OUT                                           unit 4  ! << output file >>
xyz.out                                              unit 25
iter.out                                             unit 26
flow-mbe.out                                         unit 27
coupled-mbe.out                                      unit 28
vp-pot.out                                           unit 29
vp-conc.out                                          unit 30
atmsf-hg.out                                         unit 31
nansf-hg.out                                         unit 32
hgflag.out                                           unit 33
sfflag.out                                           unit 34
psi.out                                              unit 35
vel.out                                              unit 36
conc.out                                             unit 37
sw.out                                               unit 38
ckrw.out                                             unit 39
vel-el.out                                           unit 40
potsurf.out                                          unit 41
satsurf.out                                          unit 42
swsurf.out                                           unit 43
concsurf.out                                         unit 44
nansfbc-dir-time.out                                 unit 45
nansfbc-neu-time.out                                 unit 46
sf-hg.out                                            unit 47
```

```
nansf-hg-dir.out                                      unit 48
nansf-hg-neu.out                                      unit 49
debug1.out                                            unit 61
debug2.out                                            unit 62
debug3.out                                            unit 63
debug4.out                                            unit 64
c-----------------------------------------------------------------------
```

C.2 Basic parameters: the `parm` file

The file `parm` contains the numerical simulation parameters, the time-stepping constants,and the output options, along with the specification of the partial output nodes.

```
c-----------------------------------------------------------------------
0                                            IPRT1
2  1  .TRUE.                                 IFLOW ITRANS OSC-FLAG
0  .01                                       KSLOPE TOLKSL
    -3.0     -1.0      -3.0      -1.0        PKRL     PKRR    PSEL PSER
    -3.0     -2.5      -1.5      -1.0        PDSE1L   PDSE1R  PDSE2L PDSE2R
  0   1                                      ISFONE ISFCVG
    .5      0         .5      0              TETAF  LUMP   TETAC  LUMPC
  100        12        20       1.e-4        ITMX     ITMX1   ITMX2    TOLNL
  100        7        12        .01          ITMXC  ITMXC1 ITMXC2 TOLNLC
    0        1.0e+20                         L2NORM   ERNLMX
   -3        0       .8                      ISOLV  IRELAX OMEGA
   -1       0         1.0                    ISOLVC   IRELAXC OMEGAC
  100       1.0e-10  5500      1.0e-10       ITMXCGSY TOLCGSY ITMXCGNS TOLCGNS
    .0     1.e22  10.  200.  1080.           TIMEP   DELTAT  DTMIN   DTMAX  TMAX
  .0       1.25       .0       .6            DTMAGA   DTMAGM  DTREDS   DTREDM
    3        0         0                     IPRT   ISEC   NPRT   TIMPRT
    3        1         2         3           NUMVP  NODVP
   47                                        NR (# OF OUPUT  NODES)
    2    5    8   11   62
  318  321  324  327  330
  333  336  339  342  345
  348  351  354  357  360
  363  366  369  372  375
  378  633  636  639  642
  645  648  651  654  657
  660  663  666  669  672
  675  678  681  684  687
  690  693
c-----------------------------------------------------------------------
```

C.3 2-D ground surface: the `grid` file

The file `grid` contains the definition of the 2-D ground surface grid wich is automatically replicated to create the 3-D box. The file is virtually subdived in three parts: the first one contains the 3-D grid generation options, the second contains the topology array `Grid%TRIANG`, which defines element connectivity along with the hydrogeological zone, and the third one contains the 2-D ground surface coordinate array `Grid%COORD`.

```
c-----------------------------------------------------------------
 1 10 20                        NZONE NSTR N1
 231 400                        NOD NE
 2 0 1.0                        IVERT ISP BASE
 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1  ZRATIO(1:NSTR)
 .0                             Z(1)

 1 2 13 1
 1 13 12 1
 2 3 14 1
 2 14 13 1
 3 4 15 1
 3 15 14 1
 4 5 16 1
 4 16 15 1
 5 6 17 1
 . . . (omitted). . .
 214 226 225 1
 215 216 227 1
 215 227 226 1
 216 217 228 1
 216 228 227 1
 217 218 229 1
 217 229 228 1
 218 219 230 1
 218 230 229 1
 219 220 231 1
 219 231 230 1                  TRIANG(:,NTRI)

  .00000E+00  1.0000
  .00000E+00  .90000
  .00000E+00  .80000
  .00000E+00  .70000
  .00000E+00  .60000
  .00000E+00  .50000
  .00000E+00  .40000
  .00000E+00  .30000
  .00000E+00  .20000
  .00000E+00  .10000
  .00000E+00  .00000E+00
  .10000      1.0000
  .10000      .90000
  .10000      .80000
  .10000      .70000
  .10000      .60000
  .10000      .50000
  .10000      .40000
  .10000      .30000
  .10000      .20000
  .10000      .10000
```

```
 .10000      .00000E+00
. . . (omitted). . .
 2.0000      .80000
 2.0000      .70000
 2.0000      .60000
 2.0000      .50000
 2.0000      .40000
 2.0000      .30000
 2.0000      .20000
 2.0000      .10000
 2.0000      .00000E+00       COORD(:,NNOD)
c------------------------------------------------------------------------
```

C.4 Boundary condition files

C.4.1 Flow equation

The file `nansfbcnod.flow` contains the list of non-atmospheric non-seepage face nodes (Dirichlet and Neumann) for the flow equation. The first line in the file contains the logical flag to select total h head as input variable for flow boundary conditions.

```
c------------------------------------------------------------------------
 .T.         << the BC values are in terms of total head h >>
 0           NDIR
 121         NDIRC
 221 222 223 224 225 226 227 228 229 230 231
 452 453 454 455 456 457 458 459 460 461 462
 683 684 685 686 687 688 689 690 691 692 693
 914 915 916 917 918 919 920 921 922 923 924
 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155
 1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386
 1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617
 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848
 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079
 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310
 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541
 121         NQ
 1 2 3 4 5 6 7 8 9 10 11
 232 233 234 235 236 237 238 239 240 241 242
 463 464 465 466 467 468 469 470 471 472 473
 694 695 696 697 698 699 700 701 702 703 704
 925 926 927 928 929 930 931 932 933 934 935
 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166
 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397
 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628
 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859
 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090
 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321
c------------------------------------------------------------------------
```

The file `nansfdirbc.flow` contains the boundary Dirichlet equivalentfreshwater total heads h at the sea side (on the left side of Figure 6) of the Henry domain. Since only one time sequence of values is given, the BC's are assumed fixed in time.

```
c----------------------------------------------------------------------
.0  TIME
.0       .0       .0       .0       .0       .0       .0       .0       .0       .0
.0
.25E-02  .25E-02  .25E-02  .25E-02  .25E-02  .25E-02  .25E-02  .25E-02  .25E-02  .25E-02
.25E-02
.50E-02  .50E-02  .50E-02  .50E-02  .50E-02  .50E-02  .50E-02  .50E-02  .50E-02  .50E-02
.50E-02
.75E-02  .75E-02  .75E-02  .75E-02  .75E-02  .75E-02  .75E-02  .75E-02  .75E-02  .75E-02
.75E-02
.10E-01  .10E-01  .10E-01  .10E-01  .10E-01  .10E-01  .10E-01  .10E-01  .10E-01  .10E-01
.10E-01
.13E-01  .13E-01  .13E-01  .13E-01  .13E-01  .13E-01  .13E-01  .13E-01  .13E-01  .13E-01
.13E-01
.15E-01  .15E-01  .15E-01  .15E-01  .15E-01  .15E-01  .15E-01  .15E-01  .15E-01  .15E-01
.15E-01
.18E-01  .18E-01  .18E-01  .18E-01  .18E-01  .18E-01  .18E-01  .18E-01  .18E-01  .18E-01
.18E-01
.20E-01  .20E-01  .20E-01  .20E-01  .20E-01  .20E-01  .20E-01  .20E-01  .20E-01  .20E-01
.20E-01
.23E-01  .23E-01  .23E-01  .23E-01  .23E-01  .23E-01  .23E-01  .23E-01  .23E-01  .23E-01
.23E-01
.25E-01  .25E-01  .25E-01  .25E-01  .25E-01  .25E-01  .25E-01  .25E-01  .25E-01  .25E-01
.25E-01
c----------------------------------------------------------------------
```

The file `nansfneubc.flow` contains the Neumann fluxes at the inland side of the Henry domain (on the right side of Figure 6). Since only one time sequence of values is given, the BC's are assumed fixed in time.

```
c----------------------------------------------------------------------
0.         TIME
.165E-06
.33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06
.165E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
```

```
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.33E-06
.66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06  .66E-06
.33E-06
.165E-06
.33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06  .33E-06
.165E-06
c-----------------------------------------------------------------------
```

For the Henry problem there are no atmospheric and seepage face boundary conditions so the contents of files `atmbc.flow-none` and `sfbc.flow-none` are:

```
c-----------------------------------------------------------------------
9999      HSPATM  <<no atmospheric boundary conditions>>
c-----------------------------------------------------------------------
```

and:

```
c-----------------------------------------------------------------------
0         NSF     <<no seepage face's boundary conditions>>
c-----------------------------------------------------------------------
```

respectively.

C.4.2 Transport equation

The file `bc.tran`, differently from the flow boundary condition files, contains both the nodes and the values of boundary conditions for the transport equation.

```
c-----------------------------------------------------------------------
0              NP2C
187            NPFC
1 2   3   4   5   6   7   8   9  10 11
232  233  234  235  236  237  238  239  240  241 242
463  464  465  466  467  468  469  470  471  472 473
694  695  696  697  698  699  700  701  702  703 704
925  926  927  928  929  930  931  932  933  934 935
1156  1157  1158  1159  1160  1161  1162  1163  1164  1165  1166
1387  1388  1389  1390  1391  1392  1393  1394  1395  1396  1397
1618  1619  1620  1621  1622  1623  1624  1625  1626  1627  1628
1849  1850  1851  1852  1853  1854  1855  1856  1857  1858  1859
```

```
2080  2081  2082  2083  2084  2085  2086  2087  2088  2089  2090
2311  2312  2313  2314  2315  2316  2317  2318  2319  2320  2321
1376 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386
1607 1608 1609 1610 1611 1612 1613 1614 1615 1616 1617
1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848
2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079
2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310
2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0
1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1
1  1  1  1  1  1  1  1  1  1  1
0              NMC
c-----------------------------------------------------------------------
```

C.5 Initial condition files

C.5.1 Flow equation

The file `ic.flow` contains the initial conditions of the equivalent freshwater total heads.

```
c-----------------------------------------------------------------------
 .T.        << total heads flag  (when false ICs are given as pressure heads)>>
 0          INDP   h  [m] << homogeneous condition>>
.0E+00
c-----------------------------------------------------------------------
```

C.5.2 Transport equation

The file `ic.tran` contains the initial conditions of the equivalent freshwater total heads.

```
c-----------------------------------------------------------------------
 0          INDPC  << homogeneous condition>>
.0E+00
c-----------------------------------------------------------------------
```

C.6 Material and solute properties files

The material (soil and aquifer) properties are described in file `soil`:

```
c-----------------------------------------------------------------------
  -10                                PMIN  (m)
   3                                 IVGHU(0 VG, 1 XVG, 2 HU **n, 3 HU **G, 4 BC)
   3.35  0.08   -3.0                 VGN,VGRMC,VGPSAT
   0.015 2.0     3.0  -10.0   0.01   HUALFA,HUBETA,HUGAMA,HUPSIA,HUSWR
   2.0                               HUN
   2.0   3.5                         HUA,HUB
   3.3   0.02   -0.25                BCBETA,BCRMC,BCPSAT
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        1.0E-02        1.0E-02        1.0E-02      1.00E-03       0.35E+00
        PERMX          PERMY          PERMZ        ELSTOR         POROS
c-----------------------------------------------------------------------
```

The solute properties are described in file `solute`:

```
c-----------------------------------------------------------------------
    1000.0    0.025  6.6e-6  0.0    RHO0   EPSLON   DIFFUS  EPSLON1
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
    0.0   0.0   1.0
   ALFAL ALFAT  RETARD
c-----------------------------------------------------------------------
```

C.7 Screen output: the `result.OUT` file

What follows is the screen output of the CODESA-3D model for the Henry steady state case:

```
c-----------------------------------------------------------------------
    CODESA-3D: COUPLED FLOW AND TRANSPORT CODE: 2 NxN SYSTEMS SOLVED


     SOLUTION OF THE COUPLED PROBLEM
```

```
     NEWTON SCHEME FOR FLOW EQUATION
     PICARD SCHEME FOR TRANSPORT EQUATION

IPRT1   (FOR OUTPUT OF DATA)                =      0
IPRT    (FOR DETAILED NODAL OUTPUT)         =      3
ISEC    (HOR. SEC. FOR DET. OUTPUT)         =      0
NPRT    (# OF TIME VALUES FOR DET OUTPUT) =        0
NUMVP   (# OF SURF NODES FOR VP OUTPUT)     =      3
NR      (# OF NODES FOR PARTIAL OUTPUT)     =     47

IFLOW   (1 PICARD, 2 NEWTON, 3 HUY-NEWT)  =        2
ITRANS  (0 FLOW ONLY, 1 PICARD, 2 NEWTON) =        1
OSC_FLG (.F. W/O OSC, .T. W/ OSC)           =      T
TETAF   (1 BKWD EULER, .5 C-N; FLOW EQ.)  =       5.00000E-01
LUMP    (MASS LUMPING IF NOT 0; FLOW EQ.) =        0
TETAC   (1 BKWD EULER, .5 C-N; TRNSPT EQ) =       5.00000E-01
LUMPC   (MASS LUMPING IF NOT 0; TRNSP EQ) =        0

ITMX    (MAX NONLINEAR ITER; FLOW)          =    100
ITMX1   (DELTAT INCREASE THRESHOLD)         =     12
ITMX2   (DELTAT DECREASE THRESHOLD)         =     20
TOLNL   (TOLERANCE FOR NONLINEAR ITER)      =     1.00000E-04

ITMXC   (MAX NONLINEAR ITER; COUPLED SYS) =      100
ITMXC1  (DELTAT INCREASE THRESHOLD)         =      7
ITMXC2  (DELTAT DECREASE THRESHOLD)         =     12
TOLNLC  (TOLERANCE FOR NONLINEAR ITER)      =     1.00000E-02
L2NORM  (0 L_INF, ELSE L_2 NORM FLOW EQ.) =        0
ERNLMX  (MAX CVG OR RESID ERR FLOW EQ.)     =     1.00000E+20
IRELAX  (0 NORELX,1 CONS RELX,2 VAR RELX)=       0
IRELAXC (0 NORELX,1 CONS RELX,2 VAR RELX)=       0

ISOLV   (-5 BiCGSTAB w/ diag precond,
         -4 BiCGSTAB without precond,
         -3 TFQMR   w/ diag precond,
         -2 TFQMR   without precond,
         -1 TFQMR   w/ K^-1  precond,
          0 BiCGSTAB w/ K^-1  precond,
          1 GRAMRB (min residual),
          2 GCRK(5) (ORTHOMIN),
          3 NONSYM (direct solver))         =     -3
ISOLVC  (0 GCSTAB,1 GRAMRB,2 GCRK,3 NSYM) =       -1
ITCGSY  (MAX ITER FOR CG LIN SYM SOLVER)  =      100
TLCGSY  (TOLER. FOR CG LIN SYM SOLVER)      =     1.00000E-10
ITCGNS  (MAX ITER FOR CG LIN NONSYM SLVR) =     5500
TLCGNS  (TOLER. FOR CG LIN NONSYM SOLVER) =       1.00000E-10

TIMEP   (INITIAL TIME)                      =     0.00000E+00
DELTAT  (INITIAL TIME STEP SIZE)            =     1.00000E+23
DTMIN   (MINIMUM TIME STEP SIZE)            =     1.00000E+02
DTMAX   (MAXIMUM TIME STEP SIZE)            =     2.00000E+03
TMAX    (TIME AT END OF SIMULATION)         =     1.08000E+04
DTMAGA  (MAG. FACTOR FOR DELTAT, ADD.)      =     0.00000E+00
```

```
DTMAGM  (MAG. FACTOR FOR DELTAT, MULT.)   =     1.25000E+00
DTREDS  (RED. FACTOR FOR DELTAT, SUB.)    =     0.00000E+00
DTREDM  (RED. FACTOR FOR DELTAT, MULT.)   =     6.00000E-01

NNOD   (NUM. NODI RET. BIDIMENSIONALE)  =    231
NTRI   (NUM. TRIANGOLI RET. BIDIM.)     =    400
NZONE  (NUMERO ZONE (MATERIAL TYPES))   =      1
NSTR   (NUMERO STRATI)                  =     10
N1     (NUM. MAX CONTATTI NODALI)       =     20

IVERT  (TYPE OF VERTICAL DISCRETIZATION)=      2
ISP    (0 FLAT SURFACE,
       (1 NOT FLAT reads upp. surf values)
       (2 NOT FLAT reads upper and bottom)     0
BASE   (THICKNESS OR BASE OF 3-D MESH)  =     1.00000E+00
LAYER   1     ZRATIO =   1.00000E-01
LAYER   2     ZRATIO =   1.00000E-01
LAYER   3     ZRATIO =   1.00000E-01
LAYER   4     ZRATIO =   1.00000E-01
LAYER   5     ZRATIO =   1.00000E-01
LAYER   6     ZRATIO =   1.00000E-01
LAYER   7     ZRATIO =   1.00000E-01
LAYER   8     ZRATIO =   1.00000E-01
LAYER   9     ZRATIO =   1.00000E-01
LAYER  10     ZRATIO =   1.00000E-01

NP   (TOT # OF DIRICH NODES; FLOW EQ.)  =    121

PMIN  (AIR DRY PRESSURE HEAD VALUE)           =    -9.99999E+09
IVGHU (0 VG, 1 XVG, 2 HU **n, 3 HU **G, 4 BC) =      3
HUALFA                                        =     1.50000E-02
HUBETA                                        =     2.00000E+00
HUGAMA                                        =     3.00000E+00
HUPSIA                                        =    -1.00000E+01
HUSWR                                         =     1.00000E-02
HUA                                           =     2.00000E+00
HUB                                           =     3.50000E+00


   SATURATED HYDRAULIC CONDUCTIVITY, SPECIFIC STORAGE, AND POROSITY VALUES
LAYER MAT.TYPE  X-PERM       Y-PERM       Z-PERM       STORAGE      POROSITY
  1       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  2       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  3       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  4       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  5       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  6       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  7       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  8       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
  9       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
 10       1    1.00000E-02  1.00000E-02  1.00000E-02  1.00000E-03  3.50000E-01
```

```
   LONGITUDINAL AND TRANSVERSE DISPERSIVITY VALUES AND RETARDATION FACTORS
 LAYER MAT.TYPE  ALFA-L       ALFA-T      RETARD FAC

   RHO0    (DENSITY OF FRESH WATER)         =     1.00000E+03
   EPSLON  (DENSITY DIFFERENCE RATIO)       =     2.50000E-02
   DIFFUS  (MOLECULAR DIFFUSION COEFFICIENT)=     6.60000E-06
   EPSLON1 (VISCOSITY DIFFERENCE RATIO)     =     0.00000E+00
  1      1    0.00000E+00  0.00000E+00  1.00000E+00
  2      1    0.00000E+00  0.00000E+00  1.00000E+00
  3      1    0.00000E+00  0.00000E+00  1.00000E+00
  4      1    0.00000E+00  0.00000E+00  1.00000E+00
  5      1    0.00000E+00  0.00000E+00  1.00000E+00
  6      1    0.00000E+00  0.00000E+00  1.00000E+00
  7      1    0.00000E+00  0.00000E+00  1.00000E+00
  8      1    0.00000E+00  0.00000E+00  1.00000E+00
  9      1    0.00000E+00  0.00000E+00  1.00000E+00
 10      1    0.00000E+00  0.00000E+00  1.00000E+00

   NPC  (TOT # OF DIRICH NODES; TRNSPT EQ) =    187

   N      (# OF NODES IN 3-D MESH)         =      2541
   NT     (# OF TETRAHEDRA IN 3-D MESH)    =     12000
   NTERM  (# OF NONZERO TERMS; FLOW EQ)    =     33621
   NTERMC (# OF NONZERO TERMS; TRANSP EQ)   =  33621


    TIME STEP:     1    DELTAT:   1.7000E+38    TIME:   1.7000E+38
   ****************************************************************


>>> COUPLED SYST. NONLINEAR STEP:     1 FOR THE TIME:   1.7000E+38

 >>> LINEAR SOLUTION OF FLOW EQ. >>>:
 128   6.530405E-11   1.949089E-10  <<NONSYMMETRIC SOLVER>>
 >>> LINEAR SOLUTION OF FLOW EQ. >>>:
  98   9.180640E-11   2.719063E-12  <<NONSYMMETRIC SOLVER>>
 >>> LINEAR SOLUTION OF FLOW EQ. >>>:
  98   7.518996E-11   8.566516E-13  <<NONSYMMETRIC SOLVER>>

FLOW EQ. NONLINEAR CONVERGENCE BEHAVIOR FOR THE TIME:      1.7000E+38
iter at node  psinew psiold press-head-dif  L2norm-press-head-dif res-err-L2 res-err-infty

1  2541   1.0250E+00   1.0000E+00   2.5000E-02   4.2229E-01   6.2700E-06   6.6000E-07
2  2520   1.0217E+00   1.0007E+00   2.1041E-02   6.2899E-01   1.3755E-04   2.3000E-05
3    11   2.5691E-02   2.5691E-02  -5.2434E-14   8.5472E-13   3.7433E-16   3.2919E-17
 FLOW EQ. CONVERGENCE ACHIEVED IN    3 NONLINEAR ITERATIONS
 USING A TOTAL OF  324 LINEAR ITERATIONS

 >>> LINEAR SOLUTION OF TRANSPORT EQ. >>>:
  11   5.775928E-11   3.056887E-19  <<NONSYMMETRIC SOLVER>>
```

. . . (omitted). . .

```
 >>> COUPLED SYST. NONLINEAR STEP:    31 FOR THE TIME:   1.7000E+38
     cnvrg parameter (DIFMAX):   1.2108E-02 exiting tolerance:   1.0000E-02


  >>> LINEAR SOLUTION OF FLOW EQ. >>>:
  114   9.731387E-11   1.021949E-12  <<NONSYMMETRIC SOLVER>>
  >>> LINEAR SOLUTION OF FLOW EQ. >>>:
  115   8.448196E-11   1.576940E-12  <<NONSYMMETRIC SOLVER>>

FLOW EQ. NONLINEAR CONVERGENCE BEHAVIOR FOR THE TIME:      1.7000E+38
iter at node  psinew psiold press-head-dif  L2norm-press-head-dif res-err-L2 res-err-infty

1  2486   1.0232E+00   1.0234E+00  -1.6146E-04   1.4981E-03   2.9729E-07   4.3138E-08
2  2475   1.0228E+00   1.0227E+00   9.9298E-05   8.8282E-04   3.7089E-07   5.2470E-08
 FLOW EQ. CONVERGENCE ACHIEVED IN    2 NONLINEAR ITERATIONS
 USING A TOTAL OF  229 LINEAR ITERATIONS

  >>> LINEAR SOLUTION OF TRANSPORT EQ. >>>:
   14   8.999280E-11   2.894268E-19  <<NONSYMMETRIC SOLVER>>


  COUPLED SYSTEM NONLINEAR CONVERGENCE BEHAVIOR FOR THE TIME:   1.7000E+38
  iter at node    pnew       pold       pdiff       cnew       cold       cdiff
     1  1826  1.705E-02  0.000E+00  6.556E-01  1.003E+00  0.000E+00  1.003E+00
     2  2454  2.013E-02  1.847E-02  6.390E-02  7.361E-01  4.442E-02  6.917E-01
     3   915  7.500E-03  7.500E-03  0.000E+00  7.474E-01  2.924E-01  4.550E-01
     4  1785  1.734E-02  1.760E-02 -1.001E-02  6.837E-01  3.557E-01  3.280E-01
     5  2432  2.115E-02  2.123E-02 -2.746E-03  4.974E-01  2.430E-01  2.545E-01
     6  2212  2.067E-02  2.045E-02  8.464E-03  3.464E-01  5.720E-01 -2.256E-01
     7  2225  2.038E-02  2.074E-02 -1.375E-02  7.163E-01  4.952E-01  2.211E-01
     8  2006  1.924E-02  1.899E-02  9.426E-03  5.269E-01  6.937E-01 -1.668E-01
     9  2442  2.156E-02  2.147E-02  3.333E-03  3.157E-01  4.473E-01 -1.316E-01
    10  2432  2.135E-02  2.160E-02 -9.411E-03  4.796E-01  3.359E-01  1.437E-01
    11  2002  1.926E-02  1.919E-02  2.714E-03  4.145E-01  5.366E-01 -1.222E-01
    12  2013  1.904E-02  1.917E-02 -4.827E-03  6.611E-01  5.554E-01  1.057E-01
    13  1793  1.742E-02  1.736E-02  2.311E-03  5.268E-01  6.119E-01 -8.512E-02
    14  2440  2.162E-02  2.152E-02  4.063E-03  3.719E-01  4.532E-01 -8.136E-02
    15  2453  2.175E-02  2.195E-02 -7.582E-03  6.086E-01  5.270E-01  8.160E-02
    16  2002  1.924E-02  1.918E-02  2.141E-03  4.487E-01  5.172E-01 -6.856E-02
    17  2013  1.908E-02  1.916E-02 -3.024E-03  6.373E-01  5.801E-01  5.726E-02
    18  2442  2.147E-02  2.147E-02 -1.077E-05  4.288E-01  3.835E-01  4.530E-02
    19  2439  2.161E-02  2.152E-02  3.589E-03  3.972E-01  4.414E-01 -4.429E-02
    20  2453  2.182E-02  2.196E-02 -5.300E-03  5.911E-01  5.488E-01  4.221E-02
    21  2002  1.923E-02  1.919E-02  1.346E-03  4.669E-01  5.010E-01 -3.408E-02
    22  2013  1.910E-02  1.914E-02 -1.631E-03  6.234E-01  5.959E-01  2.748E-02
    23  2433  2.154E-02  2.156E-02 -7.882E-04  4.340E-01  4.077E-01  2.633E-02
    24  2213  2.059E-02  2.057E-02  8.043E-04  4.830E-01  5.074E-01 -2.444E-02
    25  1993  1.919E-02  1.921E-02 -6.356E-04  4.870E-01  4.654E-01  2.164E-02
    26  2004  1.913E-02  1.910E-02  9.388E-04  6.008E-01  6.184E-01 -1.764E-02
    27  1784  1.738E-02  1.739E-02 -3.888E-04  5.698E-01  5.557E-01  1.407E-02
    28  2432  2.158E-02  2.160E-02 -9.226E-04  4.366E-01  4.208E-01  1.582E-02
```

```
   29  2212  2.058E-02  2.056E-02  6.878E-04  4.868E-01  5.011E-01 -1.430E-02
   30  1993  1.920E-02  1.921E-02 -4.186E-04  4.821E-01  4.700E-01  1.211E-02
   31  2004  1.912E-02  1.911E-02  5.380E-04  6.049E-01  6.146E-01 -9.704E-03
 COUPL. SYSTEM CONVERGENCE ACHIEVED IN   31 NONLINEAR ITERATIONS
USING A TOTAL OF  434 TRANSP. LINEAR ITERATIONS


 INFLOW (I) AND OUTFLOW (O) FROM ATM (A) AND NON-ATM, NON-SEEP FACE (N) BC'S;
 'C F' CURRENT FLUX; 'P F' PREVIOUS FLUX; 'VOL' VOLUME
     IA DIRIC OA DIRIC IN DIRIC ON DIRIC IA NEUMN OA NEUMN IN NEUMN ON NEUMN
C F   0.0E+00  0.0E+00  2.8E-05 -9.4E-05  0.0E+00  0.0E+00  6.6E-05  0.0E+00
P F   0.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00  0.0E+00  6.6E-05  0.0E+00
VOL   0.0E+00  0.0E+00  2.8E-05 -9.4E-05  0.0E+00  0.0E+00  1.3E-04  0.0E+00

TOTAL I VOL  TOTAL O VOL  STOR CHNG VOL | ABS MASS BAL ERR   REL MASS BAL ERR,%
  1.599E-04   -9.393E-05    0.00000E+00 |   6.60000000E-05    4.1267158384E+01


PARTIAL OUTPUT:  FLOW EQUATION
  NODE  POTENL HEAD   NODE  POTENL HEAD   NODE  POTENL HEAD   NODE  POTENL HEAD
     2   2.5777E-02      5   2.5747E-02      8   2.5737E-02     11   2.5623E-02
    62   2.2297E-02    318   2.0807E-02    321   2.0023E-02    324   2.0019E-02
   327   2.0015E-02    330   2.0011E-02    333   1.9189E-02    336   1.9184E-02
   339   1.9179E-02    342   1.8310E-02    345   1.8305E-02    348   1.8300E-02
   351   1.8294E-02    354   1.7368E-02    357   1.7361E-02    360   1.7355E-02
   363   1.7349E-02    366   1.6354E-02    369   1.6346E-02    372   1.6339E-02
   375   1.5263E-02    378   1.5254E-02    633   1.2976E-02    636   1.2965E-02
   639   1.1606E-02    642   1.1592E-02    645   1.1581E-02    648   1.1567E-02
   651   1.0073E-02    654   1.0058E-02    657   1.0046E-02    660   1.0031E-02
   663   8.3971E-03    666   8.3852E-03    669   8.3738E-03    672   6.6465E-03
   675   6.6356E-03    678   6.6294E-03    681   6.6209E-03    684   5.0000E-03
   687   5.0000E-03    690   5.0000E-03    693   5.0000E-03


PARTIAL OUTPUT:  TRANSPORT EQUATION
  NODE  CONCENTRATN   NODE  CONCENTRATN   NODE  CONCENTRATN   NODE  CONCENTRATN
     2   0.0000E+00      5   0.0000E+00      8   0.0000E+00     11   0.0000E+00
    62   3.5008E-11    318   2.3631E-09    321   1.9850E-08    324   2.5675E-08
   327   1.3909E-08    330   8.3781E-08    333   2.8305E-07    336   2.7811E-07
   339   2.8411E-07    342   2.4207E-06    345   2.1352E-06    348   2.0337E-06
   351   2.7239E-06    354   1.0203E-05    357   1.1195E-05    360   1.1387E-05
   363   8.6353E-06    366   4.9959E-05    369   4.8377E-05    372   4.8314E-05
   375   1.8193E-04    378   1.4866E-04    633   7.7202E-03    636   7.9781E-03
   639   2.0067E-02    642   1.9035E-02    645   1.9222E-02    648   1.8780E-02
   651   3.9927E-02    654   4.0337E-02    657   4.0546E-02    660   4.0639E-02
   663   8.5258E-02    666   8.6244E-02    669   8.7777E-02    672   2.1326E-01
   675   2.0627E-01    678   2.0761E-01    681   2.0707E-01    684   3.7073E-01
   687   3.7180E-01    690   3.7301E-01    693   3.7536E-01


SOLUZIONE SUI NODI DI OUTPUT
 NODE  PRESSIONE  SW         CKRW        NODE  PRESSIONE  SW         CKRW
    2  2.578E-02  1.000E+00  1.000E+00      5  2.575E-02  1.000E+00  1.000E+00
```

```
    8  2.574E-02  1.000E+00  1.000E+00    11  2.562E-02  1.000E+00  1.000E+00
   62  2.230E-02  1.000E+00  1.000E+00   318  1.208E-01  1.000E+00  1.000E+00
  321  1.200E-01  1.000E+00  1.000E+00   324  1.200E-01  1.000E+00  1.000E+00
  327  1.200E-01  1.000E+00  1.000E+00   330  1.200E-01  1.000E+00  1.000E+00
  333  1.192E-01  1.000E+00  1.000E+00   336  1.192E-01  1.000E+00  1.000E+00
  339  1.192E-01  1.000E+00  1.000E+00   342  1.183E-01  1.000E+00  1.000E+00
  345  1.183E-01  1.000E+00  1.000E+00   348  1.183E-01  1.000E+00  1.000E+00
  351  1.183E-01  1.000E+00  1.000E+00   354  1.174E-01  1.000E+00  1.000E+00
  357  1.174E-01  1.000E+00  1.000E+00   360  1.174E-01  1.000E+00  1.000E+00
  363  1.173E-01  1.000E+00  1.000E+00   366  1.164E-01  1.000E+00  1.000E+00
  369  1.163E-01  1.000E+00  1.000E+00   372  1.163E-01  1.000E+00  1.000E+00
  375  1.153E-01  1.000E+00  1.000E+00   378  1.153E-01  1.000E+00  1.000E+00
  633  2.130E-01  1.000E+00  1.000E+00   636  2.130E-01  1.000E+00  1.000E+00
  639  2.116E-01  1.000E+00  1.000E+00   642  2.116E-01  1.000E+00  1.000E+00
  645  2.116E-01  1.000E+00  1.000E+00   648  2.116E-01  1.000E+00  1.000E+00
  651  2.101E-01  1.000E+00  1.000E+00   654  2.101E-01  1.000E+00  1.000E+00
  657  2.100E-01  1.000E+00  1.000E+00   660  2.100E-01  1.000E+00  1.000E+00
  663  2.084E-01  1.000E+00  1.000E+00   666  2.084E-01  1.000E+00  1.000E+00
  669  2.084E-01  1.000E+00  1.000E+00   672  2.066E-01  1.000E+00  1.000E+00
  675  2.066E-01  1.000E+00  1.000E+00   678  2.066E-01  1.000E+00  1.000E+00
  681  2.066E-01  1.000E+00  1.000E+00   684  2.050E-01  1.000E+00  1.000E+00
  687  2.050E-01  1.000E+00  1.000E+00   690  2.050E-01  1.000E+00  1.000E+00
  693  2.050E-01  1.000E+00  1.000E+00

HGFLAG:    (1)    (2)    (3)    (4)    (5)    (6)    (7)    (8)
            0      0      0      0      0      0      0      0

 TOTAL NUMBER OF BACK-STEPPING OCCURRENCES      :     0
 TOTAL NUMBER OF LINEAR SOLVER FAILURES         :     0


 TOT CPU FOR THE SIMULATION           =     45.00 SECONDS  100.00 %
 TOT CPU FOR COUPLED SYSTEM ITERAT.   =     43.00 SECONDS   95.56 %
    (OF WHICH    29.00 SECONDS FOR NONLINEAR FLOW ITERAT  64.44 %)
 TOT CPU FOR OVERHEAD (SEE CODE DESC.) =     2.00 SECONDS    4.44 %

 TOT CPU FOR UNSAT CHARACTERISTICS          =      1.00 SECONDS    2.22 %
 TOT CPU FOR INIT. OF FLOW SYSTEM MATRICES  =      1.00 SECONDS    2.22 %
 TOT CPU FOR LOCAL FLOW SYSTEM ASSEMBLY     =      3.00 SECONDS    6.67 %
 TOT CPU FOR FLOW RHS CALCULATION W/O BC'S  =      0.00 SECONDS    0.00 %
 TOT CPU FOR FLOW GLOBAL LHS SYSTEM MATRIX  =      0.00 SECONDS    0.00 %
 TOT CPU FOR FLOW BC CONTRIBUTIONS TO RHS   =      2.00 SECONDS    4.44 %
 TOT CPU FOR FLOW LINEAR SOLVER & RESIDUAL  =     21.00 SECONDS   46.67 %
---------------------------------------------------------------------------
SUM OF FLOW TIME VEC(1:7)                   =     28.00 SECONDS   62.22 %

 TOT CPU FOR ASSEMB. & SOLUT. FLOW. EQ.      =     24.00 SECONDS   53.33 %
 TOT CPU FOR FLOW MASS BALANCE CALCULATION   =      0.00 SECONDS    0.00 %

 TOT CPU FOR INIT. OF TRANSP. SYSTEM MATRICES   =      0.00 SECONDS    0.00 %
 TOT CPU FOR LOCAL TRANSP. SYSTEM ASSEMBLY      =      4.00 SECONDS    8.89 %
 TOT CPU FOR TRANSP. RHS CALCULATION W/O BC'S   =      0.00 SECONDS    0.00 %
 TOT CPU FOR TRANSP. GLOBAL LHS SYSTEM MATRIX   =      0.00 SECONDS    0.00 %
```

```
 TOT CPU FOR TRANSP. BC CONTRIBUTIONS TO RHS    =      0.00 SECONDS    0.00 %
 TOT CPU FOR TRANSP. LINEAR SOLVER & RESIDUAL   =      9.00 SECONDS   20.00 %
-------------------------------------------------------------------------------
SUM OF TRANSP TIME VEC(1:7)                     =     13.00 SECONDS   28.89 %

 TOT CPU FOR ASSEMB. & SOLUT. TRANSP. EQ.     =     13.00 SECONDS   28.89 %
 TOT CPU FOR TRANSP MASS BALANCE CALCULATION  =      0.00 SECONDS    0.00 %

 TOT # OF TIME STEPS                 =         1
 SMALLEST TIME STEP SIZE             =    1.700E+38  AT TIME   1.700E+38
 LARGEST  TIME STEP SIZE             =    1.700E+38  AT TIME   1.700E+38
 AVERAGE  TIME STEP SIZE             =    1.700E+38
 TOT CPU TIME / (# OF TIME STEPS)    =    4.500E+01 SECONDS

 TOT # OF NONLIN ITER FOR FLOW EQ      =        62
 TOT # OF LIN ITER FOR FLOW EQ         =      6660
    ( =  0 IF ISOLV = 3)
 AVG NL ITER FOR FLOW EQ  / TIME STEP  =     62.00
 AVG LIN ITER FOR FLOW EQ / TIME STEP  =   6660.00
    ( =  0 IF ISOLV = 3)
 AVG LIN ITER FOR FLOW EQ / NONLIN ITR  =    107.42
    ( =  0 IF ISOLV = 3)
 FLOW NONLIN ITER TIME / TIME STEP     =     29.00 SECONDS
 FLOW NONLIN ITER TIME / NONLIN STEP   =      0.47 SECONDS
 TOT SIMULATION TIME   / NONLIN STEP   =      0.73 SECONDS

 TOT # OF NONLIN ITER FOR COUPLED SYS  =        31
 TOT # OF LIN ITER FOR COUPLED SYS     =       434
    ( =  0 IF ISOLV = 3)
 AVG NL ITER FOR CPLD SYS / TIME STEP  =     31.00
 AVG LIN ITER FOR CPLD SYS / TIME STEP  =    434.00
    ( =  0 IF ISOLV = 3)
 AVG LIN ITER FOR CPLD SYS / NONLIN ITR =     14.00
    ( =  0 IF ISOLV = 3)
 CPLD SYS NONLIN ITER TIME / TIME STEP  =     43.00 SECONDS
 CPLD SYS NONLIN ITER TIME / NONLIN STEP =      1.39 SECONDS
 TOT SIMULATION TIME / NONLIN STEP      =      1.45 SECONDS
c-----------------------------------------------------------------------
```

References

[1] Akin, J. E. *Application and implementation of Finite Element Methods*. Academic Press, London, 1984.

[2] Barrett, R., and *et al*. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[3] Bear, J. *Hydraulics of Groundwater*. McGraw-Hill, New York, NY, 1979.

[4] Bixio, A. C., and Mazzia, A. Strongly coupled flow and transport problems: validation of the SATC3D code and other numerical developments. Tech. rep., Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, Università di Padova, Padova, Italia, 1998.

[5] Comincioli, V. *Analisi Numerica*. McGraw-Hill, Milano, 1990.

[6] Ennabli, M. *Etude sur modèl mathématique des aquifères du Nord-Est de la Tunisie*. PhD thesis, CIG-Ecoles des Mines, Paris, France, 1977.

[7] Freeze, R. A., and Cherry, J. A. *Groundwater*. Prentice-Hall, New Jersey, 1979.

[8] Freund, R. W. A transpose free quasi-minimal residual algorithm for non-hermitian linear systems. *SIAMjsc 14*, 2 (1993), 470–482.

[9] Frind, E. O. Simulation of long-term transient density-dependent transport in groundwater. *Adv. Water Resour. 5* (1982), 73–88.

[10] Gambolati, G., Paniconi, C., and Putti, M. Numerical modeling of contaminant transport in groundwater. In *Migration and Fate of Pollutants in Soils and Subsoils, series = NATO ASI Series G: Ecological Sciences* (Berlin, 1993), vol. 32, Springer-Verlag, pp. 381–410.

[11] Gambolati, G., Pini, G., Putti, M., and Paniconi, C. Codici 2-D e 3-D agli elementi finiti per la dispersione ed il trasporto di polveri di carbone in terreni saturi ed insaturi. Relazione tecnica finale: Manuale FLOW2D. Rapporto tecnico, Dip. Metodi e Modelli Matematici per le Scienze Applicate, Università di Padova, Gennaio 1993.

[12] Gambolati, G., Pini, G., Putti, M., and Paniconi, C. Finite element modeling of the transport of reactive contaminants in variably saturated soils with LEA and non-LEA sorption. In *Environmental Modeling, Vol. II: Computer Methods and Software for Simulating Environmental Pollution and its Adverse Effects*. Computational Mechanics Publications, Southampton, UK, 1994, ch. 7, pp. 173–212.

[13] Gambolati, G., Pini, G., Putti, M., Paniconi, C., and Ferraris, S. Codici 2-D e 3-D agli elementi finiti per la dispersione ed il trasporto di polveri di carbone in terreni saturi ed insaturi. Relazione tecnica finale: Modelli 3D LEA e non-LEA. Rapporti Tecnici, Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, Università di Padova, Padova, Italy, 1994.

[14] Gambolati, G., Pini, G., Putti, M., and Sartoretto, F. Studio per l'impostazione e l'elaborazione di un modello 3-D agli elementi finiti per l'intrusione del cuneo salino in acquiferi costieri. Rapporti Tecnici, Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate, Università di Padova, Padova, Italy, 1991.

[15] Gambolati, G., Putti, M., and Paniconi, C. Three-dimensional model of coupled density-dependent flow and miscible salt transport. In *Seawater Intrusion in Coastal Aquifers — Concepts, Methods and Practices*, J. Bear, A. H.-D. Cheng, S. Sorek, D. Ouazar, and I. Herrera, Eds. Kluwer Academic, Dordrecht, The Netherlands, 1999, ch. 10, pp. 315–362.

[16] Henry, H. R. Effects of dispersion on salt encroachment in coastal aquifers. In *Sea Water in Coastal Aquifers* (1964), U.S. Geol. Surv. Water Supply Paper, No. 1613-C, pp. 70–84.

[17] Hestenes, M. R., and Stiefel, E. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards 49*, 6 (1952), 409–436.

[18] Hughes, T. J. R. *The Finite Element Method*. Prentice-Hall, London, 1987.

[19] Khlaifi, I., Lecca, G., Giacomelli, A., Tarhouni, J., and Paniconi, C. GIS data analysis and application of a 3-D variably saturated seawater intrusion model to the Korba aquifer. In *Abstracts of the Chapman Conference on Application of GIS, Remote Sensing, Geostatistics, and Solute Transport Modeling to the Assessment of Non-Point Source Pollutants in the Vadose Zone* (Riverside, California, CA, October 19-24, 1997).

[20] Kolditz, O., Ratke, R., Diersch, H. G., and Zielke, W. Coupled groundwater flow and transport: 1. Verification of variable density flow and transport models. *Adv. Water Resour. 21*, 1 (1998), 27–46.

[21] Lecca, G., Khlaifi, I., Leonardi, E., Bettio, F., Muscas, L., Tarhouni, J., and Paniconi, C. A modular approach to the Korba aquifer seawater intrusion study, 2, Simulation, data manipulation, and visualization for the 3-d model. In *Proc. of the 15th Salt Water Intrusion Meeting (SWIM)* (Ghent, Belgium, 1999), W. D. Breuck and L. Walschot, Eds., vol. 79, Flemish Journal of Natural Science, pp. 62–68.

[22] Lecca, G., Khlaifi, I., Tarhouni, J., and Paniconi, C. Modeling seawater intrusion in the Korba aquifer (Tunisia). In *Proc. of the XII International Conference on Computational Methods in Water Resources* (Southampton, UK, 1998), V. N. Burganos, G. P. Karatzas, A. C. Payatakes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, Eds., vol. II, Computational Mechanics Publications, pp. 209–216.

[23] Maidment, D. R. *Handbook of hydrology*. McGraw-Hill, New York, NY, 1993.

[24] Muscas, L., Paniconi, C., Saleri, F., and Sciabica, M. G. Modello matematico di un sistema acquifero. il caso di Capoterra (Sardegna). *Geologica Romana 30* (1994), 327–333.

[25] Paniconi, C. Soil moisture characteristic equations for variably saturated flow codes. Unpublished, 1995.

[26] Paniconi, C., and Putti, M. A comparison of Picard and Newton iteration in the numerical solution of multidimensional variably saturated flow problems. *Water Resour. Res. 30*, 12 (1994), 3357–3374.

[27] Putti, M., and Paniconi, C. Finite element modeling of saltwater intrusion problems. In *Advanced Methods for Groundwater Pollution Control* (New York, NY, 1995), vol. 364 of *CISM (Int. Centre for Mechanical Sciences) Courses and Lectures*, Springer-Verlag, pp. 65–84.

[28] Putti, M., and Paniconi, C. Picard and Newton linearization for the coupled model of saltwater intrusion in aquifers. *Adv. Water Resour. 18*, 3 (1995), 159–170.

[29] Rumynin, V. G., Mironenko, V. A., Sindalovsky, Boronina, A. V., Konosavsky, Gallo, C., and Leonardi, E. Conceptual and numerical modelling of density induced migration of radioactive contaminants at the Lake Karachai waste disposal site. In *Calibration and Reliability in Groundwater Modelling, series = Proceedings of the ModelCARE 99 Conference, Zurich, Switzerland, September 1999* (2000), vol. 265, IAHS Publ.

[30] Saad, Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Co., 1996.

[31] Saad, Y., and Schultz, M. H. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAMjssc 7* (1986), 856–869.

[32] van der Vorst, H. A. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAMjssc 13* (1992), 631–644.

[33] Walraevens, K. SWIMCA salt water management in coastal aquifers:, Development of water resource management tools for problems of seawater intrusion and contamination of fresh-water resources in coastal aquifers. Final Report. Tech. rep., EC Initiative Avicenna, AVI-CT95-73, Ghent, Belgium, 2000.

[34] Zienkiewicz, O. C. *The Finite Element Method*. McGraw-Hill, New York, NY, 1986.