DHI-WASY Software

FEFLOW®

Finite Element Subsurface Flow
& Transport Simulation System

White Papers
Vol. III

DHI-WASY GmbH

DHI-WASY GmbH,
Waltersdorfer Straße 105, D-12526 Berlin, Germany
Phone: +49-30-67 99 98-0, Fax: +49-30-67 99 98-99
E-Mail: mail@dhi-wasy.de

Contents

Contents

1 Modeling variable-density problems in 2D horizontally schematized aquifers using projected gravity

H.-J. G. Diersch

WASY Institute for Water Resources Planning and Systems Research, Berlin, Germany

Abstract
A new feature in FEFLOW allows the modeling of variable-density problems even in 2D horizontally schematized aquifers with a sloped or curved geometry. The theoretical basis, the limitations of the approach and typical applications are discussed in the paper.

1.1 Introduction

Variable-density problems plays an important role in numerous natural and engineered systems. Saltwater intrusion and geothermal transport processes are here the most typical applications in the field of geosciences. A detailed review on variable-density processes is given in Diersch and Kolditz[1]. The numerical effort in solving variable-density problems has shown generally high due to the potential need for an expensive spatial and temporal discretization. This has serious consequences particularly in modeling of 3D problems, where the meshes must be appropriately refined in all coordinate directions. In a 3D model, even if the aquifer is thin relative to its horizontal dimension, a sufficient vertical discretization is commonly required (Fig. 1.1).

However, there is a special case for which a fully three-dimensional meshing of the problem can be given up if the following conditions hold:

- There is a thin aquifer with an essentially horizontal (aquifer-type) flow for which the vertical flow components can be neglected. The horizontal extent of the aquifer is much larger compared to the aquifer thickness. Accordingly, flow and transport equations can be vertically integrated. This procedure is associated with the well-known Dupuit assumption[2].
- The aquifer is slightly sloped or curved so that gravity can effect the movement of a solute (or heat) in such an aquifer.
- The aquifer is confined.

A typical application refers to the brine movement in a large-scale deep aquifer of a basin form. The brine moves down in deeper locations of the basin by gravity effects. The process is density-driven due to the sloped geometry of the aquifer layer. Under such conditions there is a way to model the variable-density solute distribution only in two dimensions. It is based on a 2D

horizontally schematized aquifer described by vertically integrated equations and a projected gravity field. This approach is available with the FEFLOW release 5.1.

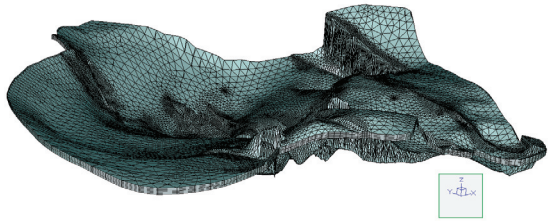

Figure 1.1 Three-dimensionally discretized basin (vertical exaggeration 6 : 1).

1.2 Basic Equations

The 2D vertically averaged flow and transport equations valid for a *confined aquifer* can be summarized as (*cf.*[2]):

$$\bar{S}_o \frac{\partial h}{\partial t} + \nabla \cdot \bar{\boldsymbol{q}}^f = \bar{Q}_\rho \tag{1-1}$$

$$\bar{\boldsymbol{q}}^f = -\boldsymbol{T} f_\mu \cdot \left(\nabla h + \frac{\rho^f - \rho_o^f}{\rho_o^f} \boldsymbol{e}' \right) \tag{1-2}$$

$$\frac{\partial}{\partial t}(\varepsilon \bar{R} C) + \nabla \cdot (\bar{\boldsymbol{q}}^f C - \bar{\boldsymbol{D}} \cdot \nabla C) + \varepsilon \bar{R} \vartheta C = \bar{Q}_C \tag{1-3}$$

$$\frac{\partial}{\partial t}[B(\varepsilon \rho^f c^f + (1-\varepsilon)\rho^s c^s) T] + \nabla \cdot (\rho^f \bar{\boldsymbol{q}}^f c^f T - \bar{\underline{\lambda}} \cdot \nabla T) = \bar{Q}_T \tag{1-4}$$

with

$$\begin{aligned}
B &= x_l^{top} - x_l^{bottom} \\
\bar{\boldsymbol{D}} &= B(\varepsilon D_d + \beta_T \|\bar{\boldsymbol{q}}^f\|)\boldsymbol{I} + (\beta_L - \beta_T)\frac{\bar{\boldsymbol{q}}^f \otimes \bar{\boldsymbol{q}}^f}{\|\bar{\boldsymbol{q}}^f\|} \\
\bar{R} &= B\left[1 + \frac{(1-\varepsilon)}{\varepsilon}\chi(C)\right] \\
\bar{\underline{\lambda}} &= \bar{\underline{\lambda}}^{cond} + \bar{\underline{\lambda}}^{disp} \qquad \bar{Q}_T = B[\varepsilon \rho^f Q_T^f + (1-\varepsilon)\rho^s Q_T^s] \\
\bar{\underline{\lambda}}^{cond} &= B[\varepsilon \lambda^f + (1-\varepsilon)\lambda^s]\boldsymbol{I} \qquad \bar{\underline{\lambda}}^{disp} = \rho^f c^f \left[\beta_T \|\bar{\boldsymbol{q}}^f\| \boldsymbol{I} + (\beta_L - \beta_T)\frac{\bar{\boldsymbol{q}}^f \otimes \bar{\boldsymbol{q}}^f}{\|\bar{\boldsymbol{q}}^f\|}\right] \\
\rho^f &= \rho_o^f \left[1 + \frac{\bar{\alpha}}{(C_s - C_o)}(C - C_o) - \bar{\beta}(T - T_o)\right] \\
\boldsymbol{e}' &= -\frac{\boldsymbol{g}'}{\|\boldsymbol{g}'\|}
\end{aligned} \tag{1-5}$$

The symbols are explained in Appendix A. The important difference to the standard formulation for 2D horizontal problems in confined aquifers (see the Reference Manual[2]) is the gravity term $(\rho^f - \rho_o^f)\boldsymbol{e}'/\rho_o^f$ appearing in (1-2). This term normally vanishes for a 'perfect' horizontal aquifer geometry because the gravity acts always perpendicular (vertical) to the aquifer horizon. However, if the aquifer is sloped or curved there are components of the gravity directed along the layer of the aquifer.

1.3 Local (Layer-oriented) Coordinates x' and the Projected Gravity Term

Let us consider the situation of an inclined aquifer layer as shown in Fig. 1.2. We introduce local coordinates $\boldsymbol{x}'$ at a local point on the inclined aquifer in such a manner that x' and y' form the principal axes corre-

lated with the geological layer structure, while z' is directed perpendicular to the actual 2D $x'-y'$-computational plane.

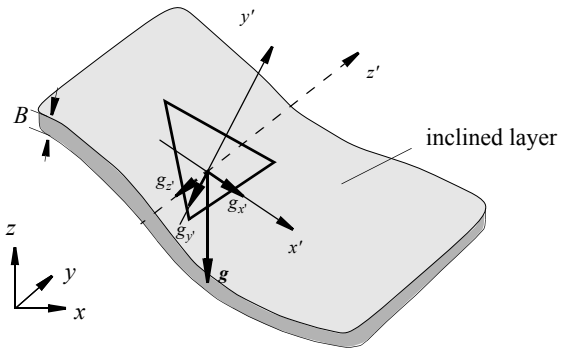

Figure 1.2 Global $x-y-z$-coordinate system and local (rotated) $x'-y'-z'$-coordinate system for a finite element located on an inclined aquifer layer. Global gravity vector $\boldsymbol{g}$ dissected by its local components $g_{x'}, g_{y'}, g_{z'}$.

The coordinate transformation between the global coordinates $\boldsymbol{x}$ and the local (layer-oriented) coordinates $\boldsymbol{x}'$ is described by the rotation matrix $\boldsymbol{a}$ as

$$\boldsymbol{x}' = \boldsymbol{a} \cdot \boldsymbol{x}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{1-6}$$

Accordingly, the gravity components in the local coordinates are given by the transformation

$$\boldsymbol{g}' = \boldsymbol{a} \cdot \boldsymbol{g}$$

$$\begin{bmatrix} g_{x'} \\ g_{y'} \\ g_{z'} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} g_x \\ g_x \\ g_z \end{bmatrix} \tag{1-7}$$

The rotation matrix $\boldsymbol{a}$ is performed for each finite element e as $\boldsymbol{a}^e$, which is thoroughly described in[3]. The components of $\boldsymbol{a}^e$ have the form:

$$a_{ij}^e = \cos(\boldsymbol{u}_i, \boldsymbol{e}_j) = \frac{\boldsymbol{u}_i^T \cdot \boldsymbol{e}_j}{\|\boldsymbol{u}_i\|} \quad \text{for} \quad \begin{matrix} i = 1, 2 \\ j = 1, 2, 3 \end{matrix}$$

$$\boldsymbol{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \qquad \boldsymbol{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{1-8}$$

where $\boldsymbol{u}_i$ are directional vectors, which are evaluated for each finite element e in the 3D space[3].

The complete set of governing equations (1-1) to (1-5) are formulated in the local coordinates $\boldsymbol{x}'$, where the gravitational unit vector $\boldsymbol{e}'$ in (1-2) is also written for the local $x'-y'$-components directed along the principal axes of the inclined layer. They can be computed by the projection

$$\begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix} = \boldsymbol{e}' = -\frac{\boldsymbol{g}'}{\|\boldsymbol{g}'\|} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \tag{1-9}$$

Note, in (1-9) it is $\|\boldsymbol{g}'\| = \|\boldsymbol{g}\|$ and it is assumed that the gravity acts strictly downwards parallel to the global z-axis, i.e., $\boldsymbol{g}^T = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}$ and $\|\boldsymbol{g}\| = g$, where g is the

gravitational acceleration constant.

In using this transformation procedure the 3D problem is mapped onto a 2D geometry so as exemplified for an idealized hemispherical basin geometry in Fig. 1.3. The variable-density effect is illustrated in Fig. 1.4 for this example. It shows how a dense solute sinks down to the centre of the hemispherical basin in time caused by an exclusive action of gravity (i.e., forced by free convection). The density-driven solute movement is strongly dependent on the parameter heterogeneity so as indicated.

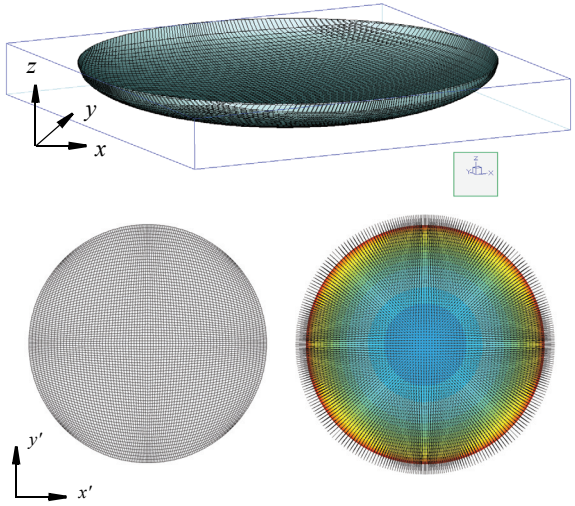

Figure 1.3 Transformation of a 3D hemispherical basin geo-metry into a 2D projected domain: a) 3D geometry in global coordinates, b) 2D projected 'horizontal' mesh to be solved in local coordinates, and c) plot of projected gravity components in the $x'-y'$ plane.

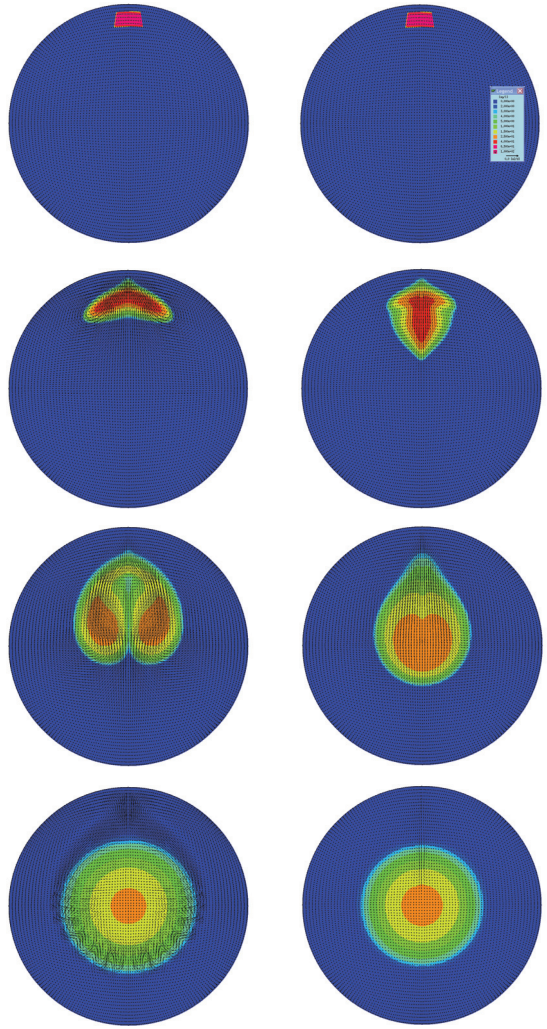

Figure 1.4 Sinking down of a brine into a hemispherical basin in time: (left column) homogeneous transmissivity, (right column) heterogeneous transmissivity distribution.

1.4 FEFLOW Option Settings

The density-couple problem which will to be solved by such a 2D horizontal aquifer schematization with a projected gravity is to be defined in FEFLOW (from release 5.1) as follows:

- 2D horizontal problem for a *confined* aquifer,
- flow and mass, flow and heat or thermohaline problem type.

With these problem definitions FEFLOW's Flow Material Data editor provides the new specific entries ***Density ratio (related)*** or/and ***Expansion coefficient (related)*** as shown in Fig. 1.5. If selecting the button(s) a dialog appears which allows the input of the density ratio or the expansion coefficient constants as well as the required link to a reference distribution, which must contain the global *z*-coordinates of the aquifer layer. If the density or the expansion coefficient constants are set to zero (as default) there will be no density coupling.

Figure 1.5 Input of the so-called *related* density parameters: a) solute density ratio or expansion coefficient constant(s), b) a link to the reference distribution containing the *z*-coordinates of the aquifer, and c) preview of the resulting projected gravity field.

These entries for the *related* density parameters are only attainable if a (at least one) reference distribution is introduced before in FEFLOW's Reference Data Editor. In this Reference Data Editor the global *z*-coordinates of the aquifer layer have to be assigned at each node of the 2D mesh. Accordingly, the slope or curvature of the aquifer layer can be described locally in an arbitrary manner. The global *z*-coordinates can be extracted from an elevation distribution of an already existing 3D model or via the common regionalization techniques based on sample point data.

Important note: The 2D coordinates of the finite element mesh have to be always input in the standard *global* *x*-*y*-coordinates, i.e., x and y are coordinates projected on the exact horizon. The transformation into the $x'-y'$-coordinates is done completely internally in the simulator and is hidden by the user. The computational results are again shown and evaluated strictly in the *global* *x*-*y*-coordinates.

1.5 Limitations

This approach is applicable for relatively thin aquifers in which flow and density effects in the z'-direction perpendicular to layer-oriented principal directions are negligible. This can be often assumed for aquifer layers having a small slope or low curvature in their elevations. Furthermore, the solute (or heat) must be assumed invariable over the aquifer thickness (that means along z'). If the aquifer slope is becoming larger and the density effects are increasing, the density-driven convection process modeled by such a projected gravity field must be more and more inaccurate due to

the fact that the approach suppresses vertical velocities when becoming dominant in a convection process. The inaccuracy particularly increases with the increasing slope of the layer for free convection at a high density contrast (high Rayleigh number) when the solute (or heat) movement is fully gravity-driven. For mixed convection problems (combined with a forced flow dynamics induced, for instance, by pumping), however, a larger slope in the geometry can often be tolerated. Generally, it is not possible to fix limits in form of critical slopes and curvature because it is context-dependent. In case of doubt a full 3D model should be used for cross-checking purposes.

1.6 Benchmark Example

Let us compare the proposed gravity-projection approach against a standard solution for a density-driven convection problem (Fig. 1.6). We choose a transient fingering problem with a high density contrast (Rayleigh number should be 1667) which refers to the most critical problem class in the present context. We will find that even for such a difficult example the agreement of the results is very well.

The *reference problem* is a vertical fingering convection process. In a closed box a dense solute sinks down to the bottom producing a characteristic fingering pattern in time as shown in Fig. 1.7 (left). The density ratio $\bar{\alpha}$ is $5 \cdot 10^{-3}$. The measure of the box is 1 m x 1 m. The conductivity K is 10^{-4} m/s, the porosity ε is 0.3, molecular diffusion D_d is 10^{-9} m^2/s, there is no dispersivity $\beta_L = \beta_T = 0.0$. The box is impervious with respect to both the flow and the solute.

We compare the results to a sloped aquifer layer approach. The slope of the layer θ is assumed uniform with $5°$. To get a full physical equivalence to the reference problem we have to choose as follows: measure of the x-y-projected area is 0.9962 m ($\cos\theta$) x 1m, the transmissivity T is 10^{-4} m^2/s, the density ratio $\bar{\alpha}$ must be increased to $5 \cdot 10^{-3} / \sin\theta = 5.7369 \cdot 10^{-2}$. The remaining parameters are the same.

The solutions of the reference (standard vertical) problem and the solutions of the projected convection problem are compared in Fig. 1.7. It reveals a very good agreement between the reference problem and the solution for the sloped layer with gravity projection.

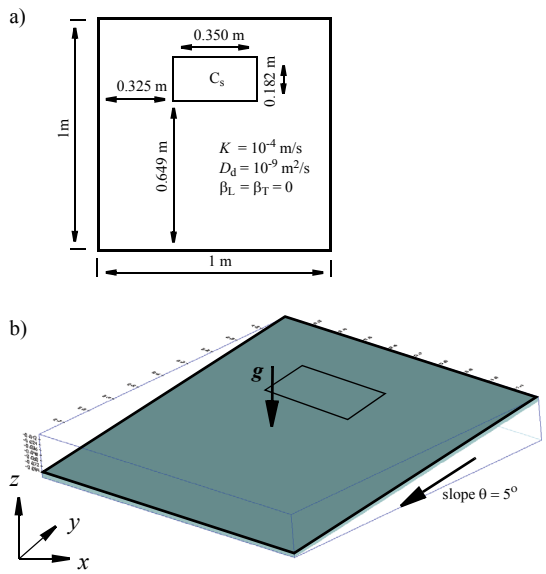

Figure 1.6 Benchmark example: a) reference problem of a vertical domain, b) equivalent sloped layer problem with projected gravity field.

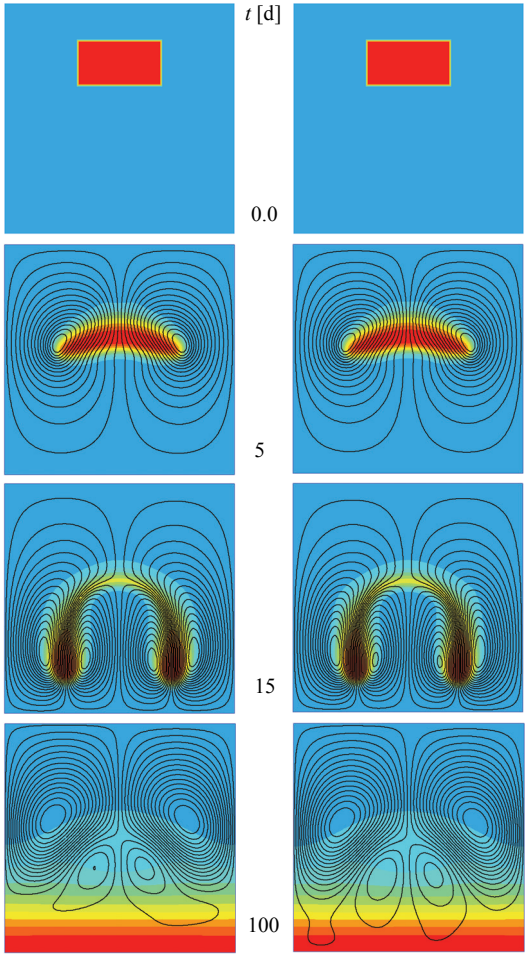

Figure 1.7 Comparison between the vertical (left column) and the sloped layer (right column) fingering convection of a dense solute sinking down in a square closed box (concentrations and streamlines are shown at selected times in the *x*-*y*-plane, fringes are upscaled to the maximum time values).

1.7 Conclusion

FEFLOW (release 5.1 and further) is now capable of solving density-driven problems in a 2D horizontal schematization by using a projected gravity field. It represents an important and cost-effective alternative to a full 3D solution whenever the density-dependent mass and/or heat transport problem can be modeled for a single confined aquifer with a small to moderate sloped or curved geometry. It should be particularly useful in modeling large-scale brine transport for mining and hydrogeological applications.

References

1. Diersch HJG, Kolditz O. Variable-density flow and transport in porous media: approaches and challenges. Advances in Water Resources 2003;25:899-944 [see also: FEFLOW White Papers Vol. II, Chapter 1; WASY Ltd, Berlin; 2001:5-111].
2. Diersch HJG. FEFLOW Reference Manual. WASY Ltd, Berlin, 2001.
3. Diersch HJG. FEFLOW White Papers Vol. I, Chapter 9; WASY Ltd, Berlin; 2001:147-190.

Appendix A

Nomenclature

Latin symbols

$\boldsymbol{a}$	1	rotation matrix;
B	L	aquifer thickness;
C, C_o	ML^{-3}	concentration and reference

		concentration (salinity), respectively;
C_s	ML^{-3}	maximum concentration;
c	$L^2T^{-2}\Theta^{-1}$	specific heat capacity of fluid;
$\overline{\boldsymbol{D}}$	L^3T^{-1}	tensor of mechanical dispersion;
D_d	L^2T^{-1}	molecular diffusion in the porous medium;
$\boldsymbol{e}'$	1	gravitational unit vector with respect to local coordinates;
f_μ	1	fluid viscosity relation function;
$\boldsymbol{g}$	LT^{-2}	gravity vector in global coordinate directions;
$\boldsymbol{g}'$	LT^{-2}	gravity vector in local coordinate directions;
g	LT^{-2}	gravitational acceleration;
h	L	hydraulic (piezometric) head;
$\boldsymbol{I}$	1	unit vector;
$\overline{Q}_\rho$	LT^{-1}	depth-integrated flow sink/source;
$\overline{Q}_C$	$ML^{-2}T^{-1}$	depth-integrated mass sink/source;
$\overline{Q}_T$	MT^{-3}	depth-integrated thermal sink/source;
$\overline{\boldsymbol{q}}^f$	L^2T^{-1}	depth-integrated Darcy flux vector;
$\overline{R}$	L	depth-integrated retardation factor;
$\overline{S}_o$	1	depth-integrated storage coefficient;
$\boldsymbol{T}$	L^2T^{-1}	tensor of transmissivity;
T, T_o	Θ	temperature and reference temperature, respectively;
t	T	time;
$\boldsymbol{u}$	L	directional vector;
$\boldsymbol{x}$	L	global Cartesian coordinate vector;
x, y, z	L	global Cartesian coordinates;
$\boldsymbol{x}'$	L	local Cartesian coordinate vector;
x', y', z'	L	local Cartesian coordinates;

Greek symbols

$\overline{\alpha}$	1	solutal expansion coefficient;
$\overline{\beta}$	Θ^{-1}	thermal expansion coefficient;
β_L, β_T	L	longitudinal and transverse dispersivity, respectively;
ε	1	porosity;
θ	°	slope of aquifer layer;
$\underline{\Lambda}$	$ML^2T^{-3}\Theta^{-1}$	depth-integrated tensor of thermal hydrodynamic dispersion;
λ^f, λ^s	$MLT^{-3}\Theta^{-1}$	thermal conductivity for fluid and solid, respectively;
ρ^f, ρ^f_o	ML^{-3}	fluid density and reference fluid density;
$\chi(C)$	1	adsorption function;
∇	L^{-1}	Nabla (vector) operator with respect to local coordinates;

Subscripts

i, j	coordinate indices;
o	reference value;
x, y, z	global coordinate directions;
x', y', z'	local coordinate directions;

Superscripts

e	element;
f	fluid (water) phase;
s	solid phase;

Derivation of the coefficients of thermal expansion and compressibility for use in FEFLOW

2

F. Magri

GeoForschungszentrum Potsdam, Division 4.3, Potsdam-Telegrafenberg, Germany

2.1 Fluid Density Equation of State (EOS)

Generally, the fluid density ρ^f varies with pressure p, temperature T and concentration of various components C_k (where C_k stands for the C_k's of all components present in the fluid) according to relations called equations of state:

$$\rho^f = \rho^f(p, T, C_k) \tag{2-1}$$

Symbols are summarized in the Appendix A. From (2-1) it follows that

$$\begin{aligned} d\rho^f &= \left.\frac{\partial \rho^f}{\partial T}\right|_{p,\,C_k} dT + \left.\frac{\partial \rho^f}{\partial p}\right|_{T,\,C_k} dp + \sum_k \left.\frac{\partial \rho^f}{\partial C_k}\right|_{p,\,T} dC_k \\ &= \rho^f(\beta dT + \gamma dp + \alpha_k dC_k) \end{aligned} \tag{2-2}$$

where

$$\left.\begin{aligned} \beta &= \frac{1}{\rho^f}\left.\frac{\partial \rho^f}{\partial T}\right|_{p,\,C_k} \\ \gamma &= \frac{1}{\rho^f}\left.\frac{\partial \rho^f}{\partial p}\right|_{T,\,C_k} \\ \alpha_k &= \frac{1}{\rho^f}\left.\frac{\partial \rho^f}{\partial C_k}\right|_{p,\,T} \end{aligned}\right\} \tag{2-3}$$

with:

β : coefficient of thermal expansion at constant pressure and concentration,

γ : coefficient of compressibility of the fluid at constant temperature and concentration,

α_k : introduces the effect of a density change due to the concentration of a kth component at constant temperature and pressure.

If, in certain ranges of p, T and C_k, the coefficients β, γ and α_k are constants or can be approximated as such for a given fluid, the equation of state (2-1) takes on the specific form[1]

$$\rho^f = \rho_o^f \exp\left[-\beta(T-T_o)+\gamma(p-p_o)+\sum_k \alpha_k(C_k-C_{ko})\right] \approx \rho_o^f\left(1-\bar{\beta}(T-T_o)+\bar{\gamma}(p-p_o)+\sum_k \bar{\alpha}_k(C_k-C_{ko})\right) \quad (2\text{-}4)$$

with

$$\left.\begin{aligned}\bar{\beta} &= \frac{1}{\rho_o^f}\frac{\partial \rho^f}{\partial T}\Big|_{p,\,C_k}\\ \bar{\gamma} &= \frac{1}{\rho_o^f}\frac{\partial \rho^f}{\partial p}\Big|_{T,\,C_k}\\ \bar{\alpha}_k &= \frac{1}{\rho_o^f}\frac{\partial \rho^f}{\partial C_k}\Big|_{p,\,T}\end{aligned}\right\} \quad (2\text{-}5)$$

where $\rho^f = \rho_o^f$ when $T = T_o$, $p = p_o$ and $C_k = C_{ko}$ that is when T, p and C_k are respectively equal to the reference temperature T_o, reference pressure p_o and reference concentration C_{ko}. Equation (2-4) states that the density ρ^f can be approximated by a linear form.

In FEFLOW the following EOS for fluid density is implemented[2]:

$$\rho^f = \rho_o^f\left(1-\bar{\beta}(T-T_o)+\frac{\bar{\alpha}}{(C_s-C_o)}(C-C_o)\right) \quad (2\text{-}6)$$

with $\bar{\beta}$ defined in (2-5) and $\bar{\alpha}$ is normalized by the saturation concentration of the solute at saturation, C_s.

It is important to notice that the EOS for the fluid density coded in FEFLOW version 5.0 (Eq. (2-6)) is valid only in a rage of 0-100°C. Moreover it does *not* take in account the coefficient of compressibility $\bar{\gamma}$ and only *one* component can be considered in the effect of a density change due to its concentration. The following empirical relationship is given for $\bar{\alpha}$

$$\bar{\alpha} = \frac{\rho^f(C_s)-\rho_o^f}{\rho_o^f} \quad (2\text{-}7)$$

While the above linear approximation for $\bar{\alpha}$ is normally sufficient for the most practical needs, wide ranges of pressure and temperature require variable thermal fluid expansion $\bar{\beta}$ (Eq. (2-5)) and fluid compressibility $\bar{\gamma}$ (Eq. (2-5)) within the state equation of density (2-6).

2.2 Extended EOS

The coefficients of thermal expansion $\bar{\beta}(p, T)$ and compressibility $\bar{\gamma}(p, T)$ for water will be derived for a wide range of pressure $p_{\mathrm{Sat}} < p \le 100$ MPa and temperature $0 \le T \le 350$ °C and coded for the finite-element program FEFLOW 5.1.

Figure 2.1 shows the well-known pressure-temperature diagram of water. Since we are interested in liquid phase in a range of 0-350 °C for the temperature and less or equal than 100 MPa for the pressure, we focus our study in region 1, which boundaries are $p_{\mathrm{Sat}} < p \le 100$ MPa and $0 \le T \le 350$ °C, where p_{Sat} is the saturation pressure of water.

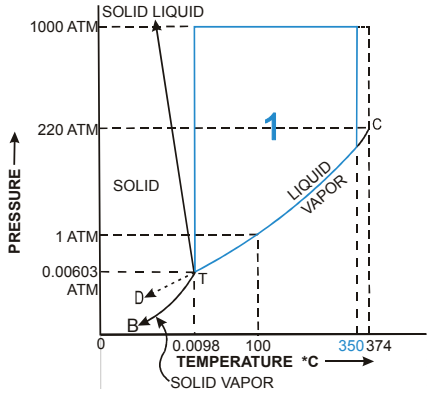

Figure 2.1 Pressure-temperature diagram of water. Domain of interest is region 1, which boundaries are $p_{Sat} < p \le 100$ MPa and $0 \le T \le 350$ °C .

The following expression provides a good approximation with an accuracy of 0.5% for ρ^f in the region 1 (Fig. 2.1):

$$\rho^f(T,p) = a(p) + b(p)T + c(p)T^2 + d(p)T^3 + e(p)T^4 + f(p)T^5 + g(p)T^6 \quad \text{in } [\text{kg/m}^3] \tag{2-8}$$

$0 \le T \le 350$ °C and $p_{Sat} < p \le 100$ MPa

where

$$\left.\begin{aligned} a(p) &= a_o + a_1 p + a_2 p^2 \\ b(p) &= b_o + b_1 p + b_2 p^2 \\ &\cdots \\ g(p) &= g_o + g_1 p + g_2 p^2 \end{aligned}\right\} \tag{2-9}$$

with temperature T in °C and the pressure p in kPa. The coefficients for fresh water are listed in Tab. 2.1.

Table 2.1 Coefficients of the polynomial surface fitting of freshwater density $\rho^f(T,p)$ in region 1 as expressed in Eq. (2-9) and its derivatives (T in °C and p in kPa)

Coefficient	Value
a_o	9.99792877961606E+02
a_1	5.07605113140940E-04
a_2	-5.28425478164183E-10
b_o	5.13864847162196E-02
b_1	-3.61991396354483E-06
b_2	7.97204102509724E-12
c_o	-7.53557031774437E-03
c_1	6.32712093275576E-08
c_2	-1.66203631393248E-13
d_o	4.60380647957350E-05
d_1	-5.61299059722121E-10
d_2	1.80924436489400E-15
e_o	-2.26651454175013E-07
e_1	3.36874416675978E-12
e_2	-1.30352149261326E-17
f_o	6.14889851856743E-10
f_1	-1.06165223196756E-14
f_2	4.75014903737416E-20

Table 2.1 Coefficients of the polynomial surface fitting of freshwater density $\rho^f(T,p)$ in region 1 as expressed in Eq. (2-9) and its derivatives (T in °C and p in kPa) (continued)

Coefficient	Value
g_o	-7.39221950969522E-13
g_1	1.42790422913922E-17
g_2	-7.13130230531541E-23

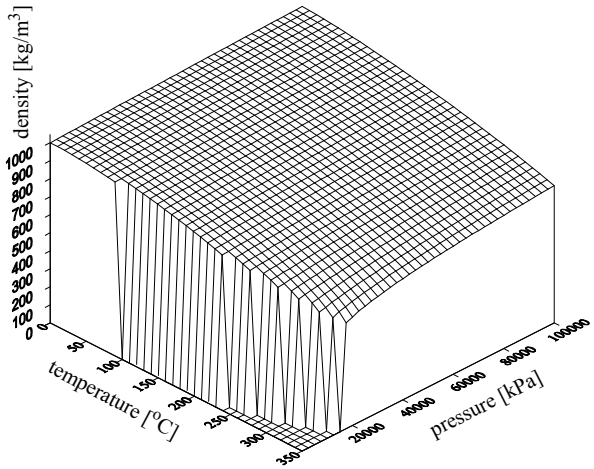

Figure 2.2 Freshwater density ρ^f as a function of pressure and temperature in region 1. For pictorial clarity, ρ^f is set to zero outside region 1.

The coefficients have been derived for freshwater conditions, so the surface (Fig. 2.2) given by Eq. (2-8) is related to a reference concentration $C_o = 0$. Therefore introducing a reference temperature T_o and a reference pressure p_o we can derive an expression for the reference fluid density ρ_o^f from (2-8), *viz.*,

$$\rho_o^f(T_o, C_o, p_o) = a(p_o)\big|_{C_o} + b(p_o)\big|_{C_o} T_o + c(p_o)\big|_{C_o} T_o^2 + d(p_o)\big|_{C_o} T_o^3 + e(p_o)\big|_{C_o} T_o^4 + f(p_o)\big|_{C_o} T_o^5 + g(p_o)\big|_{C_o} T_o^6 \quad (2\text{-}10)$$

For instance, if we take atmospheric pressure (p_o = 100 kPa) and $T_o = 0$ °C as the reference pressure and temperature, respectively, from (2-10) and the coefficients definition in (2-9) we obtain:

$$\rho_o^f(T_o, C_o, p_o) = a(p_o) = a_o + a_1 p_o + a_2 p_o^2 = 998.8396 \text{ [g/l]} \quad (2\text{-}11)$$

The goal is to find the thermally variable fluid density and the variable fluid compressibility of the fluid density function written in the following form

$$\rho^f = \rho_o^f\Big[1 + \underbrace{\bar{\gamma}(T,p)}_{\text{variable expansion}}(p - p_o) - \underbrace{\bar{\beta}(T,p)}_{\text{variable expansion}}(T - T_o)\Big] \quad (2\text{-}12)$$

where no effects in the fluid density from concentration of a component are taken in account.

For purposes of implementation in FEFLOW, we consider a Taylor series expansion for the fluid density around T_o and p_o where a 6th order approximation is used for the temperature T and a 2nd order approximation is used for the pressure p, *viz.*,

$$\begin{aligned}\rho^f(T,p) = {} & \rho^f(T_o,p_o) + (\rho^f)^{(1,0)}\big|_{T_o,p_o}(T-T_o) + \frac{1}{2!}(\rho^f)^{(2,0)}\big|_{T_o,p_o}(T-T_o)^2 + \frac{1}{3!}(\rho^f)^{(3,0)}\Big|_{T_o,p_o}(T-T_o)^3 + \frac{1}{4!}(\rho^f)^{(4,0)}\Big|_{T_o,p_o}(T-T_o)^4 \\ & + \frac{1}{5!}(\rho^f)^{(5,0)}\Big|_{T_o,p_o}(T-T_o)^5 + \frac{1}{6!}(\rho^f)^{(6,0)}\Big|_{T_o,p_o}(T-T_o)^6 \\ & + (p-p_o)\Big[(\rho^f)^{(0,1)}\big|_{T_o,p_o} + (\rho^f)^{(1,1)}\big|_{T_o,p_o}(T-T_o) + \frac{1}{2!}(\rho^f)^{(2,1)}\big|_{T_o,p_o}(T-T_o)^2 + \frac{1}{3!}(\rho^f)^{(3,1)}\Big|_{T_o,p_o}(T-T_o)^3 + \frac{1}{4!}(\rho^f)^{(4,1)}\Big|_{T_o,p_o}(T-T_o)^4 \\ & + \frac{1}{5!}(\rho^f)^{(5,1)}\Big|_{T_o,p_o}(T-T_o)^5 + \frac{1}{6!}(\rho^f)^{(6,1)}\Big|_{T_o,p_o}(T-T_o)^6\Big] \\ & + (p-p_o)^2\Big[(\rho^f)^{(0,2)}\big|_{T_o,p_o} + \frac{1}{2}(\rho^f)^{(1,2)}\Big|_{T_o,p_o}(T-T_o) + \frac{1}{4}(\rho^f)^{(2,2)}\big|_{T_o,p_o}(T-T_o)^2 + \frac{1}{12}(\rho^f)^{(3,2)}\Big|_{T_o,p_o}(T-T_o)^3 + \frac{1}{48}(\rho^f)^{(4,2)}\Big|_{T_o,p_o}(T-T_o)^4 \\ & + \frac{1}{240}(\rho^f)^{(5,2)}\Big|_{T_o,p_o}(T-T_o)^5 + \frac{1}{1440}(\rho^f)^{(6,2)}\Big|_{T_o,p_o}(T-T_o)^6\Big]\end{aligned} \tag{2-13}$$

By utilizing Eq. (2-8) we can calculate the above derivatives at p_o and T_o leading to the following equation:

$$\begin{aligned}\rho^f(T,p) = {} & a_0 + p_0a_1 + p_0^2a_2 + T_0(b_0 + p_0b_1 + p_0^2b_2) + T_0^2(c_0 + p_0c_1 + p_0^2c_2) + T_0^3(d_0 + p_0d_1 + p_0^2d_2) + \\ & T_0^4(e_0 + p_0e_1 + p_0^2e_2) + T_0^5(f_0 + p_0f_1 + p_0^2f_2) + T_0^6(g_0 + p_0g_1 + p_0^2g_2) + \\ & (T-T_0)^6(g_0 + p_0g_1 + p_0^2g_2) + (T-T_0)^5\left(f_0 + p_0f_1 + p_0^2f_2 + 6T_0(g_0 + p_0g_1 + p_0^2g_2)\right) + \\ & (T-T_0)^4\left(e_0 + p_0e_1 + p_0^2e_2 + 5T_0(f_0 + p_0f_1 + p_0^2f_2) + 15T_0^2(g_0 + p_0g_1 + p_0^2g_2)\right) + \\ & (T-T_0)^3\left(d_0 + p_0d_1 + p_0^2d_2 + 4T_0(e_0 + p_0e_1 + p_0^2e_2) + 10T_0^2(f_0 + p_0f_1 + p_0^2f_2) + 20T_0^3(g_0 + p_0g_1 + p_0^2g_2)\right) + \\ & (T-T_0)^2\left(c_0 + p_0c_1 + p_0^2c_2 + 3T_0(d_0 + p_0d_1 + p_0^2d_2) + 6T_0^2(e_0 + p_0e_1 + p_0^2e_2) + 10T_0^3(f_0 + p_0f_1 + p_0^2f_2) + \right. \\ & \left. 15T_0^4(g_0 + p_0g_1 + p_0^2g_2)\right) + \\ & (T-T_0)\left(b_0 + p_0b_1 + p_0^2b_2 + 2T_0(c_0 + p_0c_1 + p_0^2c_2) + 3T_0^2(d_0 + p_0d_1 + p_0^2d_2) + 4T_0^3(e_0 + p_0e_1 + p_0^2e_2) + \right. \\ & \left. 5T_0^4(f_0 + p_0f_1 + p_0^2f_2) + 6T_0^5(g_0 + p_0g_1 + p_0^2g_2)\right) + \\ & (p-p_0)^2\left(a_2 + T_0b_2 + T_0^2c_2 + T_0^3d_2 + T_0^4e_2 + T_0^5f_2 + T_0^6g_2 + (T-T_0)^6g_2 + (T-T_0)^5(f_2 + 6T_0g_2) + \right. \\ & (T-T_0)^4(e_2 + 5T_0f_2 + 15T_0^2g_2) + (T-T_0)^3(d_2 + 4T_0e_2 + 10T_0^2f_2 + 20T_0^3g_2) + \\ & (T-T_0)^2(c_2 + 3T_0d_2 + 6T_0^2e_2 + 10T_0^3f_2 + 15T_0^4g_2) + \\ & \left.(T-T_0)(b_2 + 2T_0c_2 + 3T_0^2d_2 + 4T_0^3e_2 + 5T_0^4f_2 + 6T_0^5g_2)\right) + \\ & (p-p_0)\Big[a_1 + 2p_0a_2 + T_0(b_1 + 2p_0b_2) + T_0^2(c_1 + 2p_0c_2) + T_0^3(d_1 + 2p_0d_2) + T_0^4(e_1 + 2p_0e_2) + \\ & T_0^5(f_1 + 2p_0f_2) + T_0^6(g_1 + 2p_0g_2) + (T-T_0)^6(g_1 + 2p_0g_2) + (T-T_0)^5\left(f_1 + 2p_0f_2 + 6T_0(g_1 + 2p_0g_2)\right) + \\ & (T-T_0)^4\left(e_1 + 2p_0e_2 + 5T_0(f_1 + 2p_0f_2) + 15T_0^2(g_1 + 2p_0g_2)\right) + \\ & (T-T_0)^3\left(d_1 + 2p_0d_2 + 4T_0(e_1 + 2p_0e_2) + 10T_0^2(f_1 + 2p_0f_2) + 20T_0^3(g_1 + 2p_0g_2)\right) + \\ & (T-T_0)^2\left(c_1 + 2p_0c_2 + 3T_0(d_1 + 2p_0d_2) + 6T_0^2(e_1 + 2p_0e_2) + 10T_0^3(f_1 + 2p_0f_2) + 15T_0^4(g_1 + 2p_0g_2)\right) + \\ & (T-T_0)\left(b_1 + 2p_0b_2 + 2T_0(c_1 + 2p_0c_2) + 3T_0^2(d_1 + 2p_0d_2) + 4T_0^3(e_1 + 2p_0e_2) + \right. \\ & \left.5T_0^4(f_1 + 2p_0f_2) + 6T_0^5(g_1 + 2p_0g_2)\right)\Big]\end{aligned} \tag{2-14}$$

2. Derivation of the coefficients of thermal expansion and compressibility for use in FEFLOW

Comparing the above equation with the EOS for the fluid density Eq. (2-12) we finally obtain the expression for computing the coefficient of thermal expansion and the coefficient of compressibility, *viz.*,

$$\begin{aligned}\bar{\beta}(T,p)=&\frac{1}{\rho_0'}\Big[(T-T_0)^5(g_0+p_0g_1+p_0^2g_2)+(T-T_0)^4\left(f_0+p_0f_1+p_0^2f_2+6T_0(g_0+p_0g_1+p_0^2g_2)\right)+\\&(T-T_0)^3\left(e_0+p_0e_1+p_0^2e_2+5T_0(f_0+p_0f_1+p_0^2f_2)+15T_0^2(g_0+p_0g_1+p_0^2g_2)\right)+\\&(T-T_0)^2\left(d_0+p_0d_1+p_0^2d_2+4T_0(e_0+p_0e_1+p_0^2e_2)+10T_0^2(f_0+p_0f_1+p_0^2f_2)+20T_0^3(g_0+p_0g_1+p_0^2g_2)\right)+\\&(T-T_0)\left(c_0+p_0c_1+p_0^2c_2+3T_0(d_0+p_0d_1+p_0^2d_2)+6T_0^2(e_0+p_0e_1+p_0^2e_2)+10T_0^3(f_0+p_0f_1+p_0^2f_2)+\right.\\&\left.15T_0^4(g_0+p_0g_1+p_0^2g_2)\right)+\\&\left(b_0+p_0b_1+p_0^2b_2+2T_0(c_0+p_0c_1+p_0^2c_2)+3T_0^2(d_0+p_0d_1+p_0^2d_2)+4T_0^3(e_0+p_0e_1+p_0^2e_2)+\right.\\&\left.5T_0^4(f_0+p_0f_1+p_0^2f_2)+6T_0^5(g_0+p_0g_1+p_0^2g_2)\right)\Big]\end{aligned} \tag{2-15}$$

and

$$\begin{aligned}\bar{\gamma}(T,p)=&\frac{1}{\rho_0'}\Big\{(p-p_0)\Big(a_2+Tb_2+T_0^2c_2+T_0^3d_2+T_0^4e_2+T_0^5f_2+T_0^6g_2+(T-T_0)^6g_2+(T-T_0)^5(f_2+6T_0g_2)+\\&(T-T_0)^4(e_2+5T_0f_2+15T_0^2g_2)+(T-T_0)^3(d_2+4T_0e_2+10T_0^2f_2+20T_0^3g_2)+\\&(T-T_0)^2(c_2+3T_0d_2+6T_0^2e_2+10T_0^3f_2+15T_0^4g_2)+\\&(T-T_0)(b_2+2T_0c_2+3T_0^2d_2+4T_0^3e_2+5T_0^4f_2+6T_0^5g_2)\Big)+\\&\Big[a_1+2p_0a_2+T_0(b_1+2p_0b_2)+T_0^2(c_1+2p_0c_2)+T_0^3(d_1+2p_0d_2)+T_0^4(e_1+2p_0e_2)+\\&T_0^5(f_1+2p_0f_2)+T_0^6(g_1+2p_0g_2)+(T-T_0)^6(g_1+2p_0g_2)+(T-T_0)^5\left(f_1+2p_0f_2+6T_0(g_1+2p_0g_2)\right)+\\&(T-T_0)^4\left(e_1+2p_0e_2+5T_0(f_1+2p_0f_2)+15T_0^2(g_1+2p_0g_2)\right)+\\&(T-T_0)^3\left(d_1+2p_0d_2+4T_0(e_1+2p_0e_2)+10T_0^2(f_1+2p_0f_2)+20T_0^3(g_1+2p_0g_2)\right)+\\&(T-T_0)^2\left(c_1+2p_0c_2+3T_0(d_1+2p_0d_2)+6T_0^2(e_1+2p_0e_2)+10T_0^3(f_1+2p_0f_2)+15T_0^4(g_1+2p_0g_2)\right)+\\&(T-T_0)\left(b_1+2p_0b_2+2T_0(c_1+2p_0c_2)+3T_0^2(d_1+2p_0d_2)+4T_0^3(e_1+2p_0e_2)+\right.\\&\left.5T_0^4(f_1+2p_0f_2)+6T_0^5(g_1+2p_0g_2)\right)\Big]\Big\}\end{aligned} \tag{2-16}$$

with ρ_o^f computed from (2-11) and the coefficients $(a_i, b_i, c_i, d_i, e_i, f_i, g_i)_{i=0,1,2}$ are given in Table 2.1.

The implementation of the extended form of EOS is done in FEFLOW by using FEFLOW's programming interface IFM. Appendix B summarizes the steps of implementation providing the IFM programming code `Beta_Gamma.c`.

References

1. Bear, J. and Corapcioglu, M.Y. Transport processes in porous media. *Proceedings of the NATO Advanced Study Institute on Transport Processes in Porous Media*, Pullmann, Washington D.C, July 9-18, 1989.
2. Diersch HJG. FEFLOW Reference Manual. WASY Ltd, Berlin, 2001.

Appendix A

Nomenclature

Latin symbols

C, C_o	ML^{-3}	concentration and reference concentration (salinity), respectively;
C_s	ML^{-3}	maximum concentration;
p, p_o	$ML^{-1}T^{-2}$	hydrodynamic fluid pressure and reference pressure, respectively;
T, T_o	Θ	temperature and reference temperature, respectively;

Greek symbols

$\bar{\alpha}$	1	solutal expansion coefficient;
$\bar{\beta}$	Θ^{-1}	thermal expansion coefficient;
$\bar{\gamma}$	$M^{-1}LT^2$	fluid compressibility;
ρ^f, ρ_o^f	ML^{-3}	fluid density and reference fluid density, respectively;

Subscripts

i	coefficient index;
k	chemical component;
o	reference value;
Sat	saturated;

Superscripts

f	fluid (water) phase;

Abbreviations

API	application programming interface;
EOS	equation of state;
IFM	interface manager;

Appendix B

Implementation of the coefficient of thermal expansion and compressibility

for use with FEFLOW through the IFM programming interface

In this appendix we will describe how to implement in FEFLOW the equations we have derived for the coefficient of thermal expansion and compressibility through the IFM programming interface.

Goal

Our goal is to incorporate in FEFLOW the following extended EOS

$$\rho^f = \rho_o^f\Big(1-\bar{\beta}(T-T_o)+\bar{\gamma}(T,p)(p-p_o) + \frac{\bar{\alpha}}{(C_s-C_o)}(C-C_o)\Big) \qquad \text{(B1)}$$

in order to reproduce the fluid density for a wide range of temperature and pressure ($p_{\mathrm{Sat}} < p \le 100$ MPa and $0 \le T \le 350$ °C). This is important for modeling heat transfer in deep geothermal reservoirs where high temperature and pressure have to be involved in the simulations.

For this purpose Eq. (B1) must include the equations for $\bar{\beta}(T,p)$ and $\bar{\gamma}(T,p)$ derived in the previous section (Eqs. (2-15) and (2-16)), which, for simplicity, will be referred henceforth to freshwater condition (C_o = 0 g/l) at the atmospheric pressure (p_o = 100 kPa) and at triple point temperature (T_o = 0 °C), i.e.,

$$\bar{\beta}(T,p) = -\frac{1}{\rho_o^f}[T^5(g_o+p_o g_1+p_o^2 g_2)+ T^4(f_o+p_o f_1+p_o^2 f_2)+T^3(e_o+p_o e_1+p_o^2 e_2)+ T^2(d_o+p_o d_1+p_o^2 d_2)+T(c_o+p_o c_1+p_o^2 c_2)+ (b_o+p_o b_1+p_o^2 b_2)] \qquad \text{(B2)}$$

and

$$\bar{\gamma}(T,p) = \frac{1}{\rho_o^f}[(p-p_o)(a_2+T^6 g_2+T^5 f_2+T^4 e_2+ T^3 d_2+T^2 c_2+T b_2)+a_1+2p_o a_2+T^6(g_1+2p_o g_2)+ T^5(f_1+2p_o f_2)+T^4(e_1+2p_o e_2)+T^3(d_1+2p_o d_2)+ T^2(c_1+2p_o c_2)+T(b_1+2p_o b_2)] \qquad \text{(B3)}$$

where $\rho_o^f(T_o, C_o, p_o)$ = 998.8396 g/l and the coefficients are given in Tab. 2.1.

We remind that in FEFLOW the following EOS for the fluid density is already incorporated, *viz.*,

$$\rho^f = \rho_o^f\Big(1-\bar{\beta}(T-T_o)+\frac{\bar{\alpha}}{(C_s-C_o)}(C-C_o)\Big) \qquad \text{(B4)}$$

Equation to implement

For achieving our goal, we will implement in the EOS (B4) an external module coding the new expres-

sion of $\bar{\beta}(T,p)$ (B2) and $\bar{\gamma}(T,p)$ (B3). The implementation of $\bar{\beta}(T,p)$ can be done directly by the use of the IFM through the API `IfmSetMatFlowExpansionCoeff`. On the other hand, since the EOS for the fluid density present in FEFLOW (B4) does not take in account the coefficient of compressibility $\bar{\gamma}(T,p)$, no related API interface exists for this coefficient, hence it's not possible to implement directly the expression for $\bar{\gamma}(T,p)$ (B3) into (B4). Therefore, to make up for this lack, we have to use a little trick:

Let's consider the EOS for the fluid density that we want to incorporate into FEFLOW (B1) and apply a simple factorization in the following way:

$$\begin{aligned}\rho^f &= \rho_o^f\left(1-\bar{\beta}(T,p)(T-T_o)+\bar{\gamma}(T,p)(p-p_o)+\frac{\bar{\alpha}}{(C_s-C_o)}(C-C_o)\right)\\ &= \rho_o^f\left[1-\left(\bar{\beta}(T,p)-\bar{\gamma}(T,p)\frac{(p-p_o)}{(T-T_o)}\right)(T-T_o)+\frac{\bar{\alpha}}{(C_s-C_o)}(C-C_o)\right]\\ &= \rho_o^f\left(1-\bar{\beta}^*(T,p)(T-T_o)+\frac{\bar{\alpha}}{(C_s-C_o)}(C-C_o)\right)\end{aligned} \quad \text{(B5)}$$

with

$$\underbrace{\bar{\beta}^*(T,p)}_{\substack{\text{defined as BETASTAR}\\ \text{in the source code}}} = \underbrace{\bar{\beta}(T,p)}_{\substack{\text{defined as BETA}\\ \text{in the source code}}} - \underbrace{\bar{\gamma}(T,p)\frac{(p-p_o)}{(T-T_o)}}_{\substack{\text{defined as GAMMASTAR}\\ \text{in the source code}}},\ T\neq T_o \quad \text{(B6)}$$

and $\bar{\beta}(T,p)$, $\bar{\gamma}(T,p)$ expressed in Eq. (B2) and Eq. (B3), respectively.

In this way, we have defined a new variable, $\bar{\beta}^*(T,p)$, which takes in account the coefficient of thermal expansion $\bar{\beta}(T,p)$ and compressibility $\bar{\gamma}(T,p)$ and that can be directly implemented in the EOS of the fluid density present in FEFLOW (B4) through the API `IfmSetMatFlowExpansionCoeff` leading to the EOS (B1).

Description of the source code `Beta_Gamma.c`

The input data required by the `IfmSetMatFlowExpansionCoeff` function are the temperature T and pressure p of each element of the FE mesh. These values can be calculated by the use of the API functions `IfmGetResultsTransportHeatValue` and `IfmGetResultsFlowPressureValue`, which load respectively the nodal values of the temperature and pressure calculated from the simulations. Once the code has derived the mean temperature and mean pressure of each element of the grid, by summing the nodal values of the considered physical parameters and then dividing by the number of nodes of the element, it calculates $\bar{\beta}^*(T,p)$ using Eq. (B6) and then set its value into the API function `IfmSetMatFlowExpansionCoeff`.

Below the complete source code `Beta_Gamma.c` is attached. The code is programmed in C language through the IFM tool in the `Simulation` interface under the callback function `PostTimeStep`.

```
static void PostTimeStep (IfmDocument pDoc)
{
  int    e, i;
  double T, p, A, B;
  double beta, gammastar, BETASTAR;

  /* Coefficients for the fitting and its derivatives */
  /* Useless a0 = 9.99792877961606e+02; */
  double a1 = 5.07605113140940e-04;
  double a2 = -5.28425478164183e-10;
  double b0 = 5.13864847162196e-02;
  double b1 = -3.61991396354483e-06;
```

```
double b2 = 7.97204102509724e-12;
double c0 = -7.53557031774437e-03;
double c1 = 6.32712093275576e-08;
double c2 = -1.66203631393248e-13;
double d0 = 4.60380647957350e-05;
double d1 = -5.61299059722121e-10;
double d2 = 1.80924436489400e-15;
double e0 = -2.26651454175013e-07;
double e1 = 3.36874416675978e-12;
double e2 = -1.30352149261326e-17;
double f0 = 6.14889851856743e-10;
double f1 = -1.06165223196756e-14;
double f2 = 4.75014903737416e-20;
double g0 = -7.39221950969522e-13;
double g1 = 1.42790422913922e-17;
double g2 = -7.13130230531541e-23;
/* Reference pressure is 100kPa while
   REFERENCE TEMPERATURE IS 0!!! */
double p0 = 100.;

/* Useless ap0 = a0+a1*p0+a2*p0*p0; */
double bp0 = b0+(b1+b2*p0)*p0;
double cp0 = c0+(c1+c2*p0)*p0;
double dp0 = d0+(d1+d2*p0)*p0;
double ep0 = e0+(e1+e2*p0)*p0;
double fp0 = f0+(f1+f2*p0)*p0;
double gp0 = g0+(g1+g2*p0)*p0;

double gam0 = a1+2*a2*p0;
double gam1 = b1+2*b2*p0;
double gam2 = c1+2*c2*p0;
double gam3 = d1+2*d2*p0;
double gam4 = e1+2*e2*p0;
double gam5 = f1+2*f2*p0;
double gam6 = g1+2*g2*p0;

/* Number of elements and number of nodes */
int nElements = IfmGetNumberOfElements(pDoc);
int nNodes    = IfmGetNumberOfNodesPerElement(pDoc);

/* Loop through all elements */
for (e = 0; e < nElements; e++) {
  T = 0.;
  p = 0.;

  /* Loop locally through all nodes of this element */
  for (i = 0; i < nNodes; i++) {
    /* Get global node index */
    int indNode = IfmGetNode(pDoc, e, i);

    T += IfmGetResultsTransportHeatValue(pDoc,indNode);
    p += IfmGetResultsFlowPressureValue(pDoc,indNode);
  }

  /* Solving the average physical properties (T and p)
     of this element */
  T /= (double)nNodes;
  p /= (double)nNodes;

  if (p < 100) p = 100;
  A = gam0 +
      (gam1+(gam2+(gam3+(gam4+(gam5+gam6*T)*T)*T)*T)*T)*T;
      B = a2+(b2+(c2+(d2+(e2+(f2+g2*T)*T)*T)*T)*T)*T;

      /* Set BETA */
      beta = -(1/999.843633188666)*
              (bp0+(cp0+(dp0+(ep0+(fp0+gp0*T)*T)*T)*T)*T);

      /* Set GAMMA */
      gammastar = -(1/999.843633188666)*(A+B*(p-p0))/(T+.1);

      /* Set BETASTAR */
      BETASTAR = beta + gammastar;
      IfmSetMatFlowExpansionCoeff(pDoc, e, BETASTAR);
      /* IfmInfo(pDoc, "Beta and gamma: %g  \n", BETASTAR); */
    }
}
```

Validation of the code

We remind that the equations for $\bar{\beta}(T,p)$ (B2) and $\bar{\gamma}(T,p)$ (B3) are derived from the following polynomial fitting:

$$\rho^f(T,p) = a(p) + b(p)T + c(p)T^2 + d(p)T^3 + e(p)T^4 + f(p)T^5 + g(p)T^6 \quad \text{in } [\text{kg/m}^3] \qquad \text{(B7)}$$

$0 \le T \le 350\ ^\circ\text{C}$ and $p_{\text{Sat}} < p \le 100$ MPa

with the temperature T in °C and, the pressure p in kPa and the coefficients for freshwater are given in Tab. 2.1.

For validating the source code, we will run a vertical 2D coupled heat transport and fluid flow problem (i.e., $\bar{\alpha} = 0$) with the implemented `Beta_Gamma` module activated in order to compare the fluid density calculated from this simulation (B5) with the one provided by the polynomial fitting (B7). The code is valid only if these values coincide.

The conceptual model used for the coupled simulation is a 3.5 x 3.5 km square. The rectangular mesh is composed by 9 elements. At the top, a constant temperature of 10 °C and a head value of 0 m are set as bound-

ary conditions while at the bottom a constant temperature of 150 °C is set.

The temperature and the pressure are calculated by the simulation for each node of the considered element. The implemented `Beta_Gamma` module derives the elemental mean value of these physical parameters and uses them as input data for returning the value of BETASTAR of each element of the mesh by the use of (B6). Results are shown in Fig 2.3.

For simplicity, we will focus our attention on the central element of the mesh. The mean temperature and pressure for this element are 79.9 °C and 1.7 x 10^4 kPa respectively. The `Beta_Gamma` module returns a value of BETASTAR equal to 3.502 x 10^{-4} K^{-1} which leads to a fluid density equal to 979.245887 kg/m^3 (B5). On the other hand, by replacing the calculated mean temperature and pressure in the polynomial fitting (B7) we obtain a fluid density equal to 979.245892 kg/m^3 which validates our code.

Important remarks

(1) Since the module implements the EOS of the fluid density referred to freshwater condition (C_o = 0 g/l) at the atmospheric pressure (p_o = 100 kPa) and at triple point temperature (T_o = 0 °C), it is fundamental to set these reference values in the FEFLOW menu for running the simulations.

(2) Referring to (B6), users must be aware that the temperature involved in the simulations must differ from the reference temperature set in the source code, i.e., T_o = 0 °C, for avoiding numerical instabilities during the calculations.

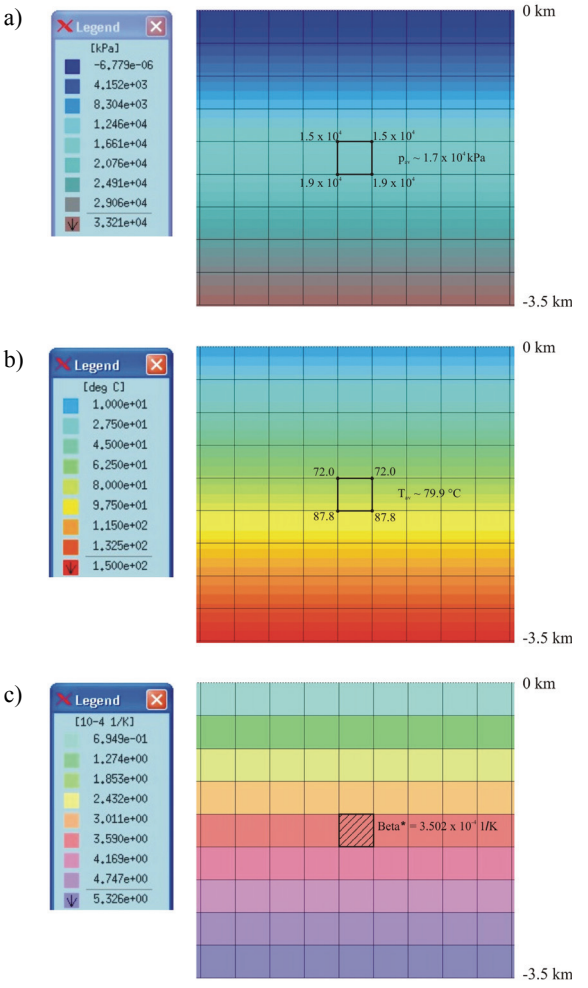

Figure 2.3 Calculated pressure (a) and temperature (a) input data and returned BETASTAR (c) by the use of the `Beta_Gamma` module.

2. Derivation of the coefficients of thermal expansion and compressibility for use in FEFLOW

Using and testing the algebraic multigrid equation solver SAMG in FEFLOW

3

H.-J. G. Diersch

WASY Institute for Water Resources Planning and Systems Research, Berlin, Germany

3.1 Motivation

For the solution of the sparse matrix systems resulting from the finite element discretization (grid, mesh) in 2D or 3D space either iterative or direct equation solvers must be applied. FEFLOW provides a family of different iterative solvers (Fig. 3.1), where the preconditioned conjugate gradient method (PCG) for symmetric matrices and the preconditioned Lanczos bi-conjugate gradient stable method (BiCGSTAB) for unsymmetric matrices are the standard options in solving large equation systems. These solvers show fast convergence and have proven efficient for typical problems over a wide range of applications in subsurface flow and transport problems. However, they also have some weaknesses and difficulties can occur, particularly if

- the spatial discretization is very heterogeneous, where the mesh is locally very dense while coarser parts exist elsewhere,
- the mesh refinement is generally very high,
- the mesh is distorted,
- the parameter contrast is high or very high,
- the angles of a triangular mesh are not optimal, e.g., high parameter contrasts occur at obtuse-angled triangles,
- poor initial estimations for the solution exist, e.g., for large timesteps or steady-state conditions.

Any of these complications can result in a significant increase of the required number of iterations. In other 'pathological' cases the iterative solvers can fail completely (no convergence) and direct Gaussian-based equation solvers would then be the only way out; unfortunately, they are rather improper for 3D problems.

As a consequence, a better and more efficient alternative should be available. This is now the case - the *Algebraic Multigrid (AMG) solution technique*, implemented and optimized in the SAMG solver[4,6], is incorporated in FEFLOW (starting with release 5.1). In the following the SAMG solution technique is briefly characterized and its usefulness and efficiency are shown in a number of typical applications. Comparisons with the standard solutions will be given.

Figure 3.1 Currently available FEFLOW equation solvers and default settings.

3.2 Algebraic Multigrid Technique and SAMG

3.2.1 Principle

A finite-element problem requires the solution of the following linear (or linearized) sparse algebraic equations

$$\boldsymbol{Ax} = \boldsymbol{b} \qquad (3\text{-}1)$$

where $\boldsymbol{A}$ is the sparse matrix, $\boldsymbol{x}$ the solution vector, and $\boldsymbol{b}$ the right-hand side. Standard iterative equation solvers, such as the PCG method, are so-called *one-level* strategies, where the matrix system (3-1) is solved for the given discretization grid as it is. In contrast, multigrid techniques[4] concern a family of efficient solution strategies that represent *hierarchical* multi-level algorithms. They combine the numerical information resulting from a hierarchy of increasingly coarse grids.

A multigrid method which operates on predefined grid hierarchies is called *geometric* multigrid, which is the traditional multi-level strategy. Unfortunately, the geometric multigrid method is restricted to hierarchically organized gridding. Commonly, meshes (grids) resulting from finite-element discretizations can be rather complex and thus are not generally suitable for geometric-hierarchical grid organization.

As a powerful alternative to the geometric multigrid method, there is the so-called *algebraic* multigrid method, where a reasonable hierarchy of grids is automatically constructed based on the algebraic information explicitly and implicitly contained in the discretization matrix $\boldsymbol{A}$. An algebraic multigrid requires only the matrix $\boldsymbol{A}$ and the right-hand side $\boldsymbol{b}$ as they result from the finite-element discretization; no specific geometric information is needed. This makes algebraic multigrid very attractive for finite elements.

An *algebraic multigrid* (AMG) is a hierarchical, matrix-based approach. Rather than on a hierarchy of grids, AMG operates on a hierarchy of *increasingly smaller linear systems of equations* which are con-

structed fully automatically. In particular, the construction of matrices used to transfer information between different levels (restriction of residuals and interpolation of corrections) is based on matrix entries, and matrices on coarser levels are computed based on the so-called Galerkin principle (Galerkin matrices). This automatism is the major reason for AMG's flexibility in adapting itself to specific requirements of the problem to be solved and for its robustness in solving large classes of problems despite using very simple smoothers[6].

AMG is a two-part process. The first part, a fully automatic *setup phase*, consists of recursively choosing the coarser levels and defining the transfer and coarse-grid operators. The second part, the *solution phase*, just uses the resulting components in order to perform normal multigrid cycling until a desired level of convergence is reached. It usually involves Gauss-Seidel relaxation for smoothing.

3.2.2 Coarsening

The coarsening strategy is the most tricky part in AMG. Because the matrix $\boldsymbol{A}$ is a result of a finite-element discretization of governing partial differential equations, AMG can take into account that the matrix entries are small stencils. A standard coarsening is based on a *direct coupling* of the matrix elements. A strong connectivity is here a *negative* coupling in $|A_{ij}|$. However, the direct connection can cause a relatively high complexity and an *aggressive coarsening* should be preferred which is based on the concept of long-range strong connections. Unfortunately, in finite element discretizations it cannot always be assumed that the strongest coupling is negative. Matrices $\boldsymbol{A}$ can occur which also contain strong *positive* entries.

3.2.3 Preconditioning

Recently, it has become very popular to use multigrid not as a stand-alone solver but rather combine it with acceleration methods such as conjugate gradient or BiCGSTAB. Experience has shown that AMG can also be a very good preconditioner, much better than standard (one-level) ILU-type preconditioners. This is mainly due to the fact that AMG, in contrast to any one-level preconditioner, efficiently operates on *all* error components, short-range as well as long-range.

3.2.4 SAMG solver package

Although the origin of AMG dates back to the early eighties[1,3] it still provides one of the most attractive algebraic approaches. Substantial progress has been achieved[5]. Today's best and most complete AMG solution strategies are available in the commercial software package SAMG[6]. SAMG has been developed by the AMG pioneer *K. Stüben* and its group at the Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), St. Augustin, Germany. SAMG is much more advanced than its academic, non-commercial forerunner, known as AMG1R5 in the scientific community. SAMG is a result of Stüben's long-term experiences and efforts in the AMG research. It is still being continuously developed, extended and improved. SAMG's usefulness and efficiency has been proven in many practical applications[4] in the field of solid and fluid mechanics.

3.2.5 FEFLOW-SAMG implementation

Starting with the FEFLOW release 5.1 the commercial SAMG solver is available as an additional option in the solver settings (Fig. 3.1). It requires an extra license in FEFLOW at a low additional cost. A property dialog allows editing of the SAMG solver options. The defaults are chosen to be suited for typical FEFLOW flow problems. In the dialog, the most important solver parameters and options can be edited. Their naming is consistent with the SAMG user's manual[6] to which the user is referred for a more detailed explanation of the options:

- iteration stop criterion, default is 10^{-8};
- types of cycling and acceleration (in the SAMG manual[6] termed as `ncyc`), default value is 12050, i.e., a V-cycle with preconditioner BiCGSTAB, at most 50 iterations;
- a general control switch for secondary parameters (in the SAMG manual[6] termed as `n_default`), by default the switch is not set and allows the specification of the coarsening strategy and matrix property via option menus (see below). Alternatively, if the switch is set the user can directly enter the `n_default` parameter as coded in SAMG's user's manual[6] to control the options manually;
- in case the control switch is not set the coarsening strategy can be chosen as *standard* (default) or *aggressive*;
- in case the control switch is not set the matrix property can be *ill-posed* (default) or *well-posed*, where the ill-posedness means that also positive strong matrix entries are allowed (see section 3.2.2);
- a *quiet* toggle, if set (default), suppresses all print output produced by SAMG; reset the toggle if information should be printed in FEFLOW's log window;
- a toggle can be set (by default it is unset) to *dump* the complete matrix system; this is only useful for analyzing SAMG via tools available to the developer.

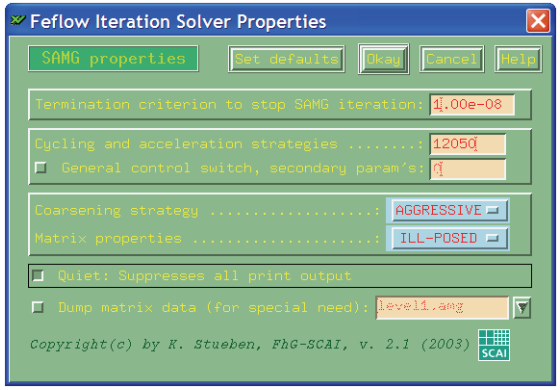

Figure 3.2 SAMG options available in FEFLOW's iteration-solver properties dialog.

3.3 Performance Tests

In the following we test the performance of SAMG against the standard PCG for selected flow problems in two and three dimensions. We start with a more systematic study for representative theoretical examples and will also consider more complex practical applications. The SAMG solver is normally used with its stan-

dard options as indicated in section 3.2.5. Different options settings will be indicated in the examples if useful.

For steady-state problems the iterations are always started with $\boldsymbol{x} = 0$. The stop criterion both for SAMG and for PCG is 10^{-8}. The measured CPU times are for a notebook with 1.8 GHz clock speed, 512 MB RAM and MS Windows XP (Version 2002, Service Pack 1). The performance measurements are obtained by using FEFLOW's time recording tool. The executable and libraries are compiler-optimized.

3.3.1 Durlofsky problem

Durlofsky[2] studied a square domain with a log-conductivity field and boundary conditions as shown in Fig. 3.3. The flow enters on the left and exits on the right side of the domain. The remaining boundaries are impervious.

The problem is steady-state and is characterized by a structured regular gridding with a heterogeneous conductivity distribution. The parameter contrast refers to a power of six. The domain is initially discretized by a 20 x 20 square quadrilateral mesh, which represents the starting (first) refinement Υ_o. We will test the matrix solution performances for a stepwise global refinement of the mesh according to

$$\Upsilon_l \qquad l = 0, 1, \ldots, L \tag{3-2}$$

where l is the refinement level and L is the maximum level (here, 6). In the global refinement of the mesh each quadrilateral is subdivided into four equally sized quadrilaterals. The number of quadrilaterals NE and number of nodes NP then increase according to the refinement level $l = 0, 1, \ldots$:

$$\begin{aligned} NE &= 20^2 \cdot 4^l \\ NP &= (\sqrt{NE} + 1)^2 = (20 \cdot 4^{l/2} + 1)^2 \end{aligned} \tag{3-3}$$

The obtained performance results are summarized in Tab. 3.1. It clearly indicates the superiority of the SAMG solver to the standard PCG for this problem once the matrices enlarge. For large problems, in the order of 10^5 or 10^6 equations, SAMG is more than ten times faster than PCG. It is important to note that SAMG does not increase the required number of iteration cycles if the problem enlarges, while the number of iterations required for the PCG solver significantly increases with the growing refinement level l.

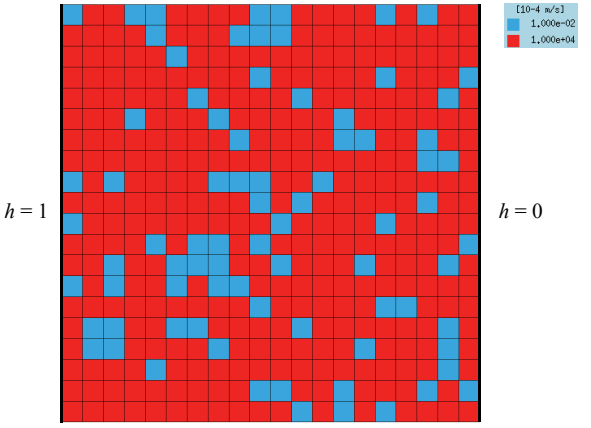

Figure 3.3 2D square test problem with a conductivity pattern (red = 1 m/s; blue = 10^{-6} m/s), basic 20 x 20 mesh Υ_o.

Table 3.1 Performance results for the Durlofsky problem refinement

l	*NE*	*NP*	PCG		SAMG		CPU-ratio PCG/SAMG
			number of iterations	CPU time[a] [sec]	number of cycles	CPU time[a] [sec]	
0	400	441	21	0.0	4	0.0	-
1	1,600	1,681	58	0.0	4	0.1	-
2	6,400	6,561	117	0.3	4	0.2	1.50
3	25,600	25,921	229	2.2	4	0.7	3.14
4	102,400	103,041	456	18.3	4	2.9	6.31
5	409,600	410,881	903	162.2	4	12.3	13.19
6[b]	1,638,400	1,640,961	1766	1320.9	4	54.2	24.37

a) encompasses matrix assembly, solution of the sparse equation system, and velocity computation
b) storage swapping partially occurred at this refinement level

3.3.2 Extreme unstructured meshing

Let us consider a triangular mesh, which is locally extremely dense as shown in Fig. 3.4. The domain has a constant conductivity. In the dense-mesh location there is a pumping well. The left and right boundaries are subjected to a constant hydraulic-head condition. The remaining boundaries are impervious. The problem is steady-state.

In the 2D case the mesh consists of 102,954 triangles (*NE*) and 51,595 nodes (*NP*). The problem is also extended to 3D by pentahedral prismatic elements, where 10 layers of constant thickness are used. It results in *NE* = 1,029,540 and *NP* = 567,545.

The obtained performances for SAMG and PCG are compared in Tab. 3.2. It reveals that SAMG is significantly faster in the 2D problem. However, for the 3D problem PCG results a similar performance. Note that SAMG becomes here somewhat faster when aggressive coarsening is used instead of the standard coarsening option.

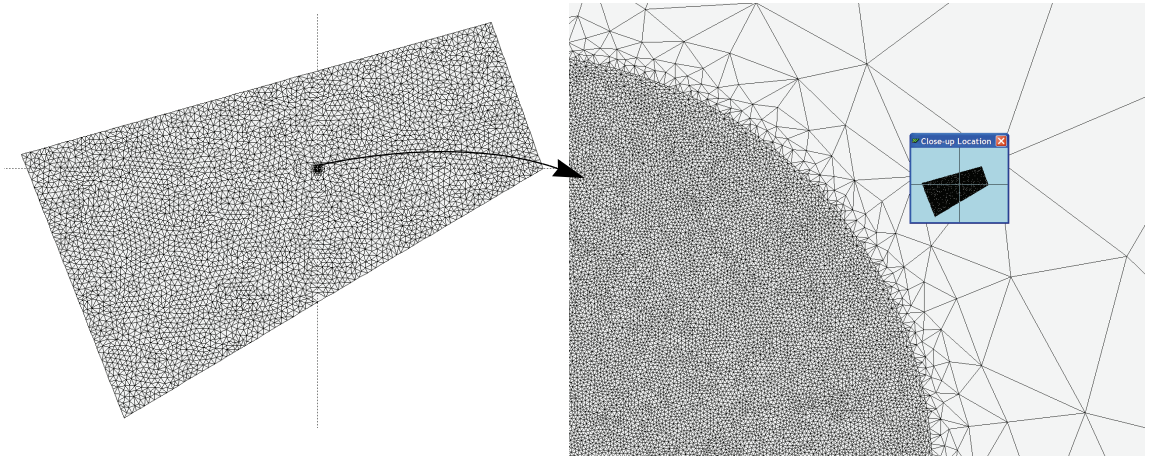

Figure 3.4 Unstructured triangular mesh with magnified view of the local dense part.

Table 3.2 Performance results for the extreme unstructured meshing problem

	PCG		SAMG		CPU-ratio PCG/SAMG
	number of iterations	CPU time[a)] [sec]	number of cycles	CPU time[a)] [sec]	
2D	356	9.5	5	2.2	4.32
3D	409	356.5	50[b)]	304.6	1.17
			50[c)]	211.2	1.69

a) encompasses matrix assembly, solution of the sparse equation system, and velocity computation
b) standard coarsening, termination after 50 cycles
c) aggressive coarsening, termination after 50 cycles

3.3.3 Cross-sectional problem with heterogeneous parameter distribution

A more complex cross-sectional 2D problem is shown in Fig. 3.5. The triangular mesh is fully unstructured and locally refined in a layered geometry. The problem models an aquifer-aquitard system with heterogeneous distribution of conductivity and storativity.

We consider both a steady-state and a transient situation (Tab. 3.3). In Fig. 3.6 the CPU times are shown for an adaptive time-stepping process starting from a constant initial solution. It reveals that PCG is superior to SAMG if the timesteps are small and the solution is not far from the distribution of the previous timestep (PCG needs here only a small number of iterations), which typically occurs at the beginning. However, as the time stepping progresses, SAMG becomes clearly superior to PCG because PCG requires significantly more iterations if the change in the solution over each timestep becomes larger. The overall CPU time for SAMG is about three times smaller than for the PCG method.

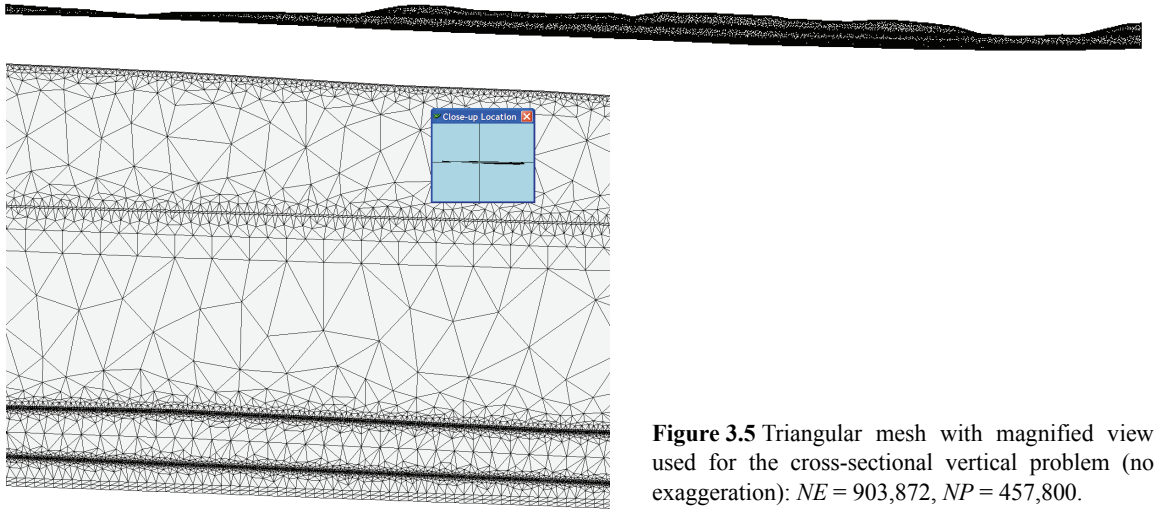

Figure 3.5 Triangular mesh with magnified view used for the cross-sectional vertical problem (no exaggeration): *NE* = 903,872, *NP* = 457,800.

Table 3.3 Performance results for the cross-sectional problem

	PCG		SAMG		CPU-ratio PCG/SAMG
	number of iterations	CPU time[a)] [sec]	number of cycles	CPU time[a)] [sec]	
steady-state	1000[b)]	262.9	5	22.2	11.84
transient (one timestep)[c)]	2	6.3	0	4.7	1.34
transient (series of timesteps)[d)]	-	739.5	-	209.2	3.53

a) encompasses matrix assembly, solution of the sparse equation system, and the velocity computation
b) stopped after 1000 iteration due to poor convergence
c) initial solution is very near the final solution
d) adaptive time stepping over 17 timesteps, starting with a constant initial distribution (see Fig. 3.6)

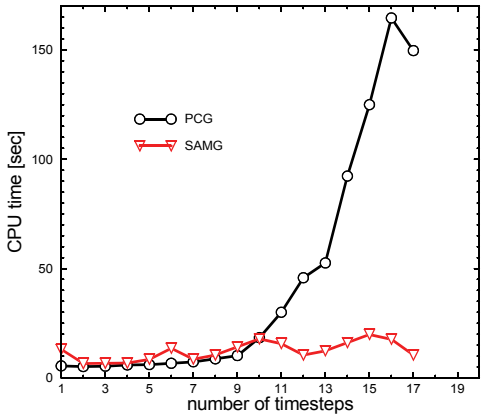

Figure 3.6 CPU times for the transient simulation of the cross-sectional problem required for the PCG and SAMG method.

3.3.4 Three-dimensional problem with highly transient boundary conditions

An unsteady 3D regional finite-element flow problem is tested as shown in Fig. 3.7. The mesh at a moderate resolution consisting of 221,210 pentahedral prismatic elements and 122,485 nodes is locally refined yet, well-formed and the parameter contrast remains moderate (conductivity ranges over five orders of magnitude). It can be considered as a typical 3D transient finite-element groundwater model used in practical water resources simulations. Of specific interest here are boundary conditions applied to rivers and pumping wells that possess a short-term dynamic (e.g., pumping capacity changes each day). The timesteps are automatically controlled by FEFLOW's adaptive predictor-corrector technique.

For such a standard simulation task the PCG

method remains clearly superior to the SAMG solvers as can be seen from Fig. 3.8 and Tab. 3.4. For this type of simulation SAMG needs more CPU time than the PCG solver. Overall, PCG is here about three times faster. Even for a higher resolution of the mesh (e.g., by using a global mesh refinement) the PCG solver is two to three times faster than the SAMG method.

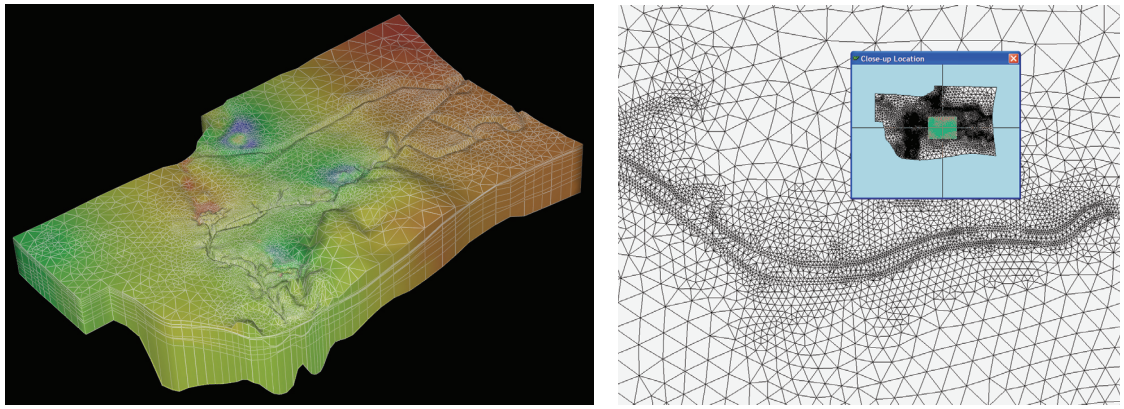

Figure 3.7 Large-scale 3D flow model: *NE* = 221,210, *NP* = 122,485.

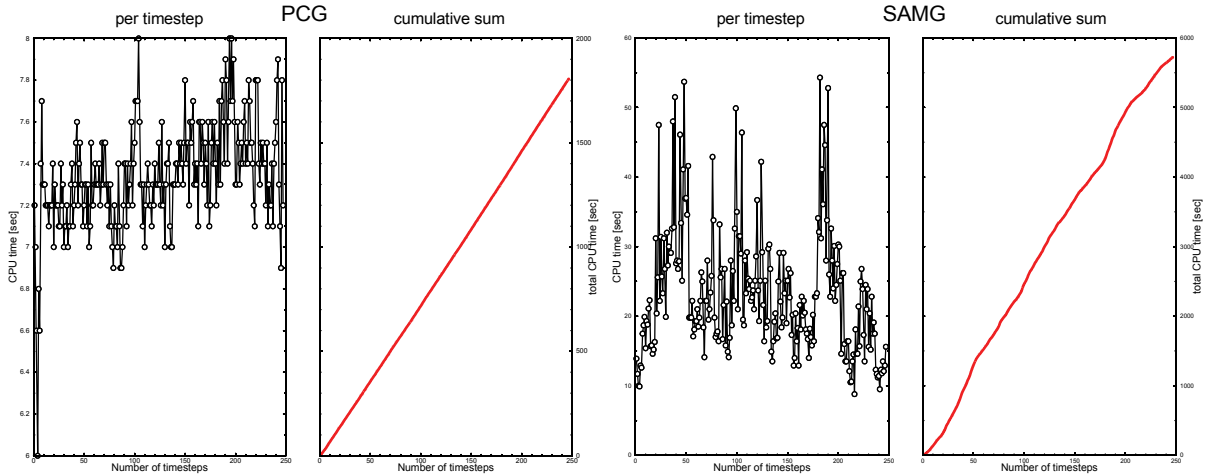

Figure 3.8 CPU times required for the PCG (left) and SAMG (right) solvers in simulating the transient the 3D flow problem (247 timesteps) at the mesh resolution *NE* = 221,210, *NP* = 122,485.

Table 3.4 Performance results for the 3D problem with highly transient boundary conditions

PCG total CPU time[a] [sec]	SAMG total CPU time[a] [sec]	CPU-ratio PCG/SAMG
1806	5724	0.32

a) encompasses matrix assembly, solution of the sparse equation system, and the velocity computation for 247 timesteps

3.3.5 Three-dimensional basin model with vertically distorted prisms at faulty zones

The final performance test refers to a 3D large-scale basin flow model. The pentahedral prismatic mesh with a moderate resolution has to incorporate a number of faulty zones, which leads to a vertical distortion of the prisms along these locations as exhibited in Fig. 3.9. The model is transient; fixed timesteps of 1 day are used. The parameter contrast is three orders of magnitude.

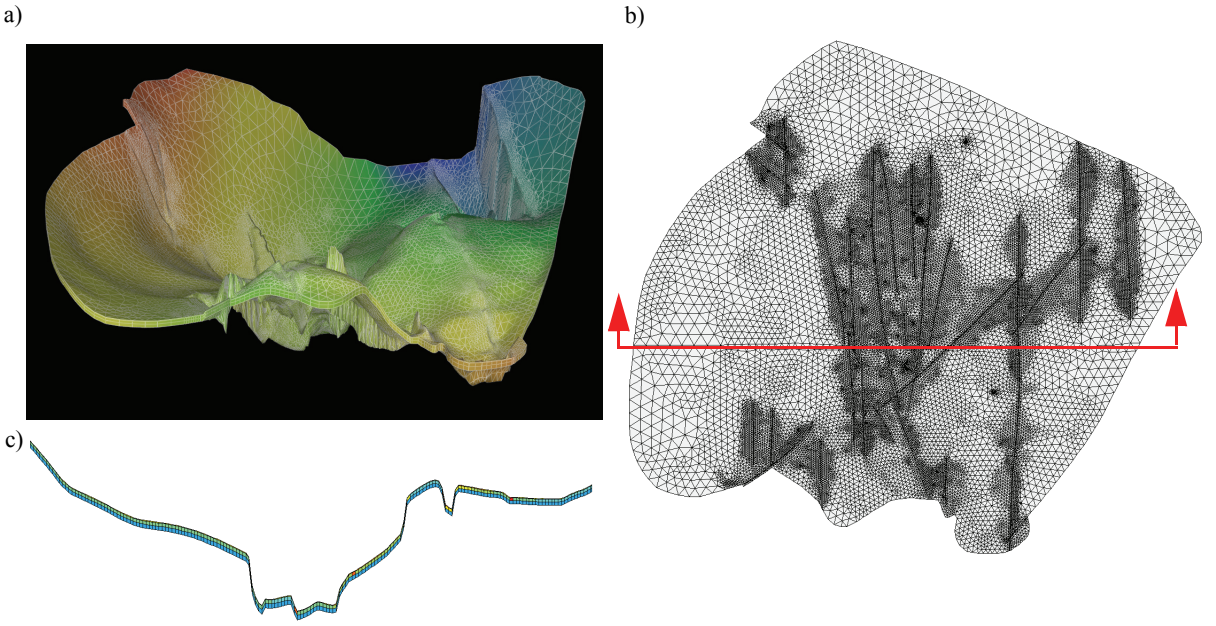

Figure 3.9 3D basin model: *NE* = 123,726, *NP* = 83,056: a) 3D view, b) horizontal view on the prismatic mesh, c) cross-section along indicated (red-colored) line (exaggeration 10 : 1).

For this problem the PCG solver completely failed. It was impossible to get convergent solutions below the residual error criterion of 10^{-8}. Continuing the simulation in time, after a small number of timesteps the solution became fully instable. In contrast to the PCG, the SAMG showed much better computational properties. SAMG was able to solve the problem. However, it should be noted that SAMG terminated after 50 cycles at each timestep. Although the residual error could not be reduced to the given error criteria of 10^{-8}, the simulation could be successfully continued in time with stable and sufficiently accurate solutions.

Table 3.5 Performance results for the basin model problem

	PCG		SAMG		CPU-ratio PCG/SAMG
	number of iterations	CPU time[a)] [sec]	number of cycles	CPU time[a)] [sec]	
at each timestep of 1 day	2000[b)] failed	131.2	50[c)] stable	23.0	5.7

a) encompasses matrix assembly, solution of the sparse equation system, and the velocity computation
b) terminated after 2000 (maximum) iterations due to poor convergence, solution continuation fails
c) terminated at 50 (maximum) cycles, solution remains stable

3.4 Conclusions

The algebraic multigrid solver SAMG is available in FEFLOW for release 5.1 and higher. SAMG has proven very powerful for difficult problems where the standard PCG solver takes a large number of iterations (poor convergence) or completely fails (divergence). This is often the case if the mesh is extremely unstructured and highly locally refined, the shape of the elements is not optimal (bad-formed elements), the parameter contrast is high, or the starting solution is far from the final solution. In steady-state solutions SAMG is generally the fastest and most robust solver. The speed-up can be a factor of ten or more. The power of SAMG increases with the number of equations (nodes). For large problems SAMG can often beat the performance of the standard PCG solver. However, for unsteady simulations, where the problem is well-posed and sufficiently smooth, PCG is usually faster due to its smaller computational overhead and numerical simplicity.

References

1. Brandt, A., McCormick, S., Ruge, J. Algebraic multigrid (AMG) for sparse matrix equations. In: *Sparsity and its Applications*, edited by D.J. Evans, Cambridge University Press, Cambridge, 1984, pp. 257-284.
2. Durlofsky, L.J. Accuracy of mixed and control volume finite element approximations to Darcy velocity and related quantities. *Water Resour. Res.* **30** (1994) 4, 965-973.
3. Stüben K. Algebraic multigrid (AMG): Experiences and comparisons. *Appl. Math. Comp.* **13** (1983), 23-56.

4. Stüben K. An introduction to Algebraic Multigrid. Appendix in the book *Multigrid* by U. Trottenberg *et al*., Academic Press, pp. 413-532, 2001 (also available as GMD-Report 70, St. Augustin, Germany, November 1999).
5. Stüben, K. A review of algebraic multigrid. *J. Comp. and Appl. Math. (JCAM)* **128** (2001), 281-309.
6. Stüben K. User's manual SAMG, Release 2.1, Fraunhofer Institute SCAI, St. Augustin, Germany, 2002.

Appendix A

Nomenclature

Latin symbols

$\boldsymbol{A}$	sparse system matrix;
$\boldsymbol{b}$	right-hand side of matrix system;
h	hydraulic head, potential;
L	maximum refinement level;
l	refinement level;
$\boldsymbol{x}$	solution vector;

Greek symbols

Υ_l	refinement at level l;

Abbreviations

AMG	algebraic multigrid;
CPU	central processing unit;
NE	number of elements;
NP	number of points (nodes);
PCG	preconditioned conjugate gradient;

Subject Index

Subject Index

Author Index

Author Index