The IPARSv2 air-water model *

Eleanor W. Jenkins †

July 19, 2002

Abstract

We document the two-phase air-water model that has been implemented in IPARSv2. We give the model equations, the discretization method, the initialization algorithm, and Jacobian formation. We also describe data structures that are included in the IPARSv2 model, including the data structure for storing derivatives and the data structure for Jacobian storage. We finish the paper with numerical examples for several problems, including a parallel example.

*Version as of July 19, 2002

†Center for Subsurface Modeling, Texas Institute for Computational and Applied Mathematics, University of Texas at Austin, ACES 5.340, 201 E. 24th Street, Austin, TX 78712 (lea@ticam.utexas.edu)

Contents

1 Introduction

This report describes version 2 of the air-water model (subdirectory airv2) that has been included in version 2 of the **I**ntegrated **Pa**rallel **R**eservoir **S**imulator, or IPARSv2, framework [2, 5, 6, 8, 10, 11, 12, 16]. The air-water code is intended to model the movement of groundwater in the unsaturated, or vadose, zone. Other software packages that model the movement of groundwater, such as FEMWATER or MODFLOW, use Richards' Equation, which only models water phase pressure; the air phase is assumed to be immobile. Thus Richards' Equation does not model the true physics as well as the two-phase system. In addition, our formulation follows a mass conservation principle, which we feel is important, especially in the case of reactive transport. We also include fully three-dimensional wells, which the other packages do not.

The IPARSv2 air-water model (subdirectory `air`) was originally developed by Wonsuck Lee and Myeong Noh [7]. They used the implicit black-oil model as their base model, and eliminated one of the three phases. In version 2 of the air-water model, the implicit hydrology model was used as the base model, with the major difference being that the non-wetting phase, i.e., air, is assumed to be compressible. The work of Lee and Noh in formulating the Jacobian and implementing the temporal integration scheme has been verified and corrected when necessary. The initial equilibrium condition for the reservoir has been added to the second version of the air-water model. The model has also been parallelized and multiblock keywords have been integrated, mimicing the capabilities of the implicit hydrology model. The boundary condition capability has been added, where the boundary condition capabilities of the IPARSv2 hydrology model were used as a basis [13].

While there are many similarities between the air-water model and the implicit hydrology model, there is a difference in the selection of the primary unknowns. For the air-water model, these unknowns are water pressure and water saturation, as opposed to oil concentration and oil pressure for the implicit hydrology model. Any differences in the models due to this choice of unknowns will be noted in this documentation. However, as in the implicit hydrology model, choosing these variables as primary unknowns does not affect the choice of variables that one can use for the boundary conditions, thus providing much-needed flexibility in the model.

This report is divided into 7 additional sections. The air-water model formulation is given in §2. This formulation is identical to the hydrology model with the added difficulty of compressibility; more details on the im-

plicit hydrology model equations may be found in [13]. In §3, the initialization procedure and the Jacobian contribution are discussed, and in §4, the boundary conditions that are available in the model are described. Numerical results are given in §5, and conclusions and future work are in §6. Information on the files that have been adjusted and written for version 2 of the air-water model is included in the appendix, labeled §8, as is more data on the numerical results.

2 Model Formulation

The air-water model equations are the same as for the two-phase implicit hydrology model, where the non-wetting phase, i.e., air, is assumed to be compressible. The compressibility assumption is included in the constitutive relationship for air density, and thus when formulating the Jacobian derivatives of air density with respect to water pressure cannot be ignored. In the case of the implicit hydrology model, these derivatives can be ignored as they are quite small due to the near incompressibility of oil and water and the Jacobians need only be approximate [4]. In the air-water model they are maintained and included when evaluating the Jacobian contributions.

2.1 Flow Equations

The mass balance equation for each of the α fluid phases is given by

$$\frac{\partial \left(\eta S_\alpha \rho_\alpha\right)}{\partial t} + \nabla \cdot \boldsymbol{u}_\alpha = q_\alpha, \tag{1}$$

where η is the porosity, S_α is the saturation of the α phase, ρ_α is the density of the α phase, P_α is the pressure of the α phase, and q_α is a source term.

Darcy's law for multi-phase flow is used to define the mass flux $\boldsymbol{u}_\alpha$ as

$$\boldsymbol{u}_\alpha = -\rho_\alpha K \frac{k_\alpha}{\mu_\alpha} \left(\nabla P_\alpha - \rho_\alpha g \nabla D\right). \tag{2}$$

Here, K is the permeability tensor, k_α is the relative permeability, μ_α is the viscosity, g is gravity, and $D = D(\boldsymbol{x})$ is the depth of the reservoir.

Substituting the Darcy velocity (2) into the mass balance equation (1) gives the governing equation for multi-phase flow of each of the α phases as

$$\frac{\partial \left(\eta S_\alpha \rho_\alpha\right)}{\partial t} - \nabla \cdot \left(\rho_\alpha K \frac{k_\alpha}{\mu_\alpha} (\nabla P_\alpha - \rho_\alpha g \nabla D)\right) = q_\alpha. \tag{3}$$

The system of two equations is closed using the saturation and capillary pressure relationships. The saturation relationship is

$$S_a + S_w = 1. \tag{4}$$

Capillary pressure is the difference in pressure across the interface between the non-wetting (air) and wetting (water) phase fluids. This is a known function of saturation, so we have that

$$P_c(S_w) = P_a - P_w. \tag{5}$$

The water phase is assumed to be slightly compressible, so that the density is given by

$$\rho_w = \rho_{w,ref} \exp^{c_w(P_w - P_{w,ref})}, \tag{6}$$

where c_w is the compressibility constant for water, $\rho_{w,ref}$ is a reference water density, and $P_{w,ref}$ is a reference water pressure. The density for the air phase is given by the real gas law as

$$\rho_a = \frac{P_a M}{Z(P_a)RT}, \tag{7}$$

where M is the molecular weight, R is the gas constant, T is the temperature, and $Z(P_a)$, the compressibility factor, is a function of air pressure.

2.2 Well Models

The well models in the air-water component are fully three-dimensional and are patterned after the well models in the implicit hydrology model. Currently, there are five well types: water injection, pressure specified; water injection, total mass rate specified; production well, pressure specified; production well, total mass rate specified; and air injection, pressure specified.

General information on the well equations, including the information given below, can be found in the IPARSv2 documentation provided with the code (subdirectory `doc`). The volumetic flow rate q_α, calculated for each element, is given by

$$q_\alpha = \frac{GLKk_\alpha \left(P_{wb} - \bar{P}_\alpha\right)}{\mu_\alpha}, \tag{8}$$

where G is a geometric factor, L is the length of the open wellbore penetrating the element, K is the permeability of the element normal to the

wellbore, k_α is the relative permeability of the α phase, P_{wb} is the wellbore pressure, and $\bar{P}_\alpha$ is the formation pressure of the α phase in the element. In defining problems with wells, the bottom hole pressure may be specified for the given well; the wellbore pressure may be obtained by using the bottom hole pressure as

$$P_{wb} = P_{bh} + \rho_{wb,\alpha} g \left(D_{wb} - D_{bh} \right),$$

where P_{bh} is the bottom hole pressure, $\rho_{wb,\alpha}$ is the average density of the α phase in the wellbore, g is gravity, D_{wb} is the depth of the wellbore in the element, and D_{bh} is the bottom hole depth.

The formation pressure, $\bar{P}_\alpha$, is determined by

$$\bar{P}_\alpha = P_\alpha + \rho_\alpha g \left(D_{wb} - D \right).$$

3 Implementation

This section contains details on the actual air-water model code. We describe the initialization procedure and the model discretization and include a description of the implicit temporal integration with the corresponding Jacobian matrix.

3.1 Initialization

Here we describe the equations that are implemented in `aivdat.df`. The reservoir is initialized using hydrostatic equilibrium conditions for water pressure P_w, air pressure P_a, and water saturation S_w. The equilibrium condition for each phase α is

$$\nabla P_\alpha - g\rho(P_\alpha)\nabla D = 0. \tag{9}$$

The density functions make this equation nonlinear; thus, the root is found using Newton's method. The user specifies a water pressure and water saturation at an initial depth. The code then loops through the elements and establishes the equilibrium condition based on the closest element to the current one. The discrete version of Equation (9), in the x coordinate direction, is

$$P_{\alpha,i+1,jk} - P_{\alpha,ijk} - g\frac{\rho_{\alpha,ijk} + \rho_{\alpha,i+1,jk}}{2} \left(D_{i+1,jk} - D_{ijk} \right) = 0, \tag{10}$$

where we use cell averages to evaluate the density.

Recall that the density for the water phase is $\rho_w = \rho_{w,ref} \exp^{c_w\left(P_w - P_{w,ref}\right)}$. We can use this to get an expression for $\rho_{w,i+1,jk}$ in terms of $\rho_{w,ijk}$ by noting that

$$\begin{aligned}
\rho_{w,i+1,jk} &= \rho_{w,ref} \exp^{c_w\left(P_{w,i+1,jk} - P_{w,ref}\right)} && (11)\\
&= \rho_{w,ref} \exp^{c_w\left(P_{w,i+1,jk} - P_{w,ijk} + P_{w,ijk} - P_{w,ref}\right)} && (12)\\
&= \rho_{w,ref} \exp^{c_w \triangle P_w} \exp^{c_w\left(P_{w,ijk} - P_{w,ref}\right)} && (13)\\
&= \rho_{w,ijk} \exp^{c_w \triangle P_w}, && (14)
\end{aligned}$$

where $\triangle P_w = P_{w,i+1,jk} - P_{w,ijk}$.

We can then formulate the discrete equation and hence the Newton function as a function of the change in water pressure ΔP_w:

$$F\left(\triangle P_w\right) = \triangle P_w - \frac{g}{2}\rho_{w,ijk}\left(1 + \exp^{c_w \triangle P_w}\right)\left(D_{i+1,jk} - D_{ijk}\right). \quad (15)$$

Thus the Jacobian for the Newton iteration is

$$F'\left(\triangle P_w\right) = 1 - \frac{g}{2} c_w \rho_{w,ijk} \exp^{c_w \triangle P_w} \triangle D, \quad (16)$$

where $\triangle D = D_{i+1,jk} - D_{ijk}$.

We obtain the initial guess for the Newton iteration by assuming that

$$\rho_{w,i+1,jk} = \rho_{w,ijk} + \left.\frac{\partial \rho_w}{\partial P_w}\right|_{P_{w,ijk}} \left(P^0_{w,i+1,jk} - P_{w,ijk}\right) \quad (17)$$

so that, using (10), we have

$$\left(P^0_{w,i+1,jk} - P_{w,ijk}\right) = \frac{g}{2} \triangle D \left(\rho_{w,ijk} + \rho_{w,ijk} + c_w \rho_{w,ijk}\left(P^0_{w,i+1,jk} - P_{w,ijk}\right)\right).$$

Thus we have that

$$\triangle P^0_w = \frac{g}{2} \triangle D \left(2\rho_{w,ijk} + c_w \rho_{w,ijk} \triangle P^0_w\right)$$

and

$$\left(1 - \frac{g}{2} \triangle D c_w \rho_{w,ijk}\right)\left(\triangle P^0_w\right) = g \triangle D \rho_{w,ijk},$$

which gives

$$\triangle P^0_w = \frac{g \triangle D \rho_{w,ijk}}{1 - \frac{g}{2} c_w \rho_{w,ijk} \triangle D} \quad (18)$$

as an initial guess for the equilibrium Newton iteration.

The next step in the initialization is the calculation of the equilibrium pressure for the air phase. Recall that the density for the air phase is given by the real gas law

$$\rho_a = \frac{MP_a}{Z(P_a)\,RT}, \tag{19}$$

so we cannot get an expression for $\rho_{a,ijk} + \rho_{a,i+1,jk}$ in terms of a pressure change $\triangle P_a$. Thus our Newton function for the air equilibrium is

$$F(P_{a,i+1,jk}) = P_{a,i+1,jk} - P_{a,ijk} - \frac{g}{2}(\rho_{a,ijk} + \rho_{a,i+1,jk})(D_{i+1,jk} - D_{ijk}).$$

Including the expression for air density gives

$$F(P_{a,i+1,jk}) = P_{a,i+1,jk} - P_{a,ijk} - \frac{g}{2} \triangle D \left(\frac{M}{RT}\right)\left(\frac{P_{a,ijk}}{Z_{ijk}} + \frac{P_{a,i+1,jk}}{Z_{i+1,jk}}\right)$$

where, as before, $\triangle D = D_{i+1,jk} - D_{ijk}$ and $Z_{ijk} = Z(P_{a,ijk})$. Thus the Jacobian is

$$F'(P_{a,i+1,jk}) = 1 - \frac{g}{2} \triangle D \left(\frac{M}{RT}\right)\left(\frac{P_{a,i+1,jk}}{Z_{i+1,jk}}\right)' \tag{20}$$

and

$$\left(\frac{P_{a,i+1,jk}}{Z_{i+1,jk}}\right)' = \frac{Z_{i+1,jk} - P_{a,i+1,jk}Z'_{i+1,jk}}{Z^2_{i+1,jk}}.$$

The initial guess for $P_{a,i+1,jk}$ is obtained similarly to the Newton guess for $P_{w,i+1,jk}$, where now

$$P^0_{a,i+1,jk} - P_{a,ijk} = \frac{g}{2} \triangle D \left(\frac{M}{RT}\right)\left(\frac{P_{a,ijk}}{Z_{ijk}} + \left(\frac{P_{a,ijk}}{Z_{ijk}}\right)'\left(P^0_{a,i+1,jk} - P_{a,ijk}\right)\right) \tag{21}$$

so that

$$\left(1 - \frac{g}{2} \triangle D \frac{M}{RT}\left(\frac{P_{a,ijk}}{Z_{ijk}}\right)'\right) P^0_{a,i+1,jk} = P_{a,ijk} + \frac{g}{2} \triangle D \frac{M}{RT}\left(\frac{P_{a,ijk}}{Z_{ijk}} - \left(\frac{P_{a,ijk}}{Z_{ijk}}\right)' P_{a,ijk}\right).$$

Hence

$$\left(1 - \frac{g}{2} \triangle D \frac{M}{RT} \left(\frac{P_{a,ijk}}{Z_{ijk}}\right)'\right) P^0_{a,i+1,jk} = \left(1 - \frac{g}{2} \triangle D \frac{M}{RT} \left(\frac{P_{a,ijk}}{Z_{ijk}}\right)'\right) P_{a,ijk} + \frac{g}{2} \triangle D \frac{M}{RT} \frac{P_{a,ijk}}{Z_{ijk}}$$

and the initial guess for the air phase equilibrium is

$$P^0_{a,i+1,jk} = P_{a,ijk} + \frac{\frac{g}{2} \triangle D \rho_{a,ijk}}{1 - \frac{g}{2} \triangle D \frac{M}{RT} \left(\frac{P_{a,ijk}}{Z_{ijk}}\right)'}. \tag{22}$$

3.2 Discretization

The model equations (3) are discretized using centered finite differences, which have been shown to be equivalent to the expanded mixed finite element method using lowest order Raviart-Thomas spaces on a rectangular grid and applying the appropriate integration rules [14]. Thus, these methods are locally mass conservative and hence appropriate for modeling groundwater flow with reactive transport. Many of the discretization details are outlined in [13] and will not be repeated here.

The mass balance equation gives three terms when discretized; an accumulation term $\mathcal{A}$, a transport term $\mathcal{T}$, and a source term from the wells $\mathcal{W}$.

The accumulation term is given as

$$\begin{aligned} \mathcal{A}_{\alpha,ijk} &=: V_{ijk} \eta_{ijk} \frac{\partial S_\alpha \rho_\alpha}{\partial t} \\ &= \phi_{ijk} \frac{\partial S_\alpha \rho_\alpha}{\partial t}, \end{aligned} \tag{23}$$

where V_{ijk} is the measure of cell Ω_{ijk} and η_{ijk} is the porosity of the cell. We define the pore volume of the cell as

$$\phi_{ijk} = V_{ijk} \eta_{ijk}.$$

In forming the Jacobian, the accumulation term should be in units of volume. When differentiating (23) with respect to water saturation S_w, we are left with units of density; thus the equations must be normalized by a reference density to maintain the units of volume. This problem does not occur in the implicity hydrology model because oil concentration, $N_o = \rho_o S_o$, is

chosen as a primary unknown instead of a saturation. Note that this normalization must occur throughout the model when formulating the Jacobian contributions.

The well term is given as

$$\mathcal{W}_{ijk} =: -\int_{\Omega} q_\alpha w_{ijk}\, dx = \int_{\Omega_{ijk}} q_\alpha\, dx, \tag{24}$$

where q_α is the source term (i.e., a volumetric flux) and w_{ijk} is piecewise constant across the cell. Thus the only contributions to $\mathcal{W}$ are from those cells that contain wells. The well terms in IPARSv2 are defined using the Peaceman formulation [9].

The transport term is given as

$$\mathcal{T}_{\alpha,ijk} \quad =: \quad \int_{\Omega} \nabla \cdot \boldsymbol{u}_\alpha w_{ijk} dx = \int_{\Omega_{ijk}} \nabla \cdot \boldsymbol{u}_\alpha dx,$$

where again w_{ijk} is piecewise constant across the cell. For transport in the x direction we use the divergence theorem to obtain

$$\begin{aligned} \mathcal{T}^{x+}_{\alpha,ijk} + \mathcal{T}^{x-}_{\alpha,ijk} \quad &=: \quad \int_{\partial\Omega_{i+1/2,jk}} \boldsymbol{u}_{\alpha,i+1/2,jk} \cdot \nu dx + \int_{\partial\Omega_{i-1/2,jk}} \boldsymbol{u}_{\alpha,i-1/2,jk} \cdot \nu dx \\ &= \quad \triangle y_j \triangle z_k (\boldsymbol{u}_{\alpha,i+1/2,jk} - \boldsymbol{u}_{\alpha,i-1/2,jk}), \end{aligned}$$

where $\partial\Omega_{i+1/2,jk}$ is the $x+$ part of $\partial\Omega_{ijk}$. Terms in the y, z direction are computed similarly.

The discrete velocity values $\boldsymbol{u}_{\alpha,i+1/2,jk}$ are defined using the transmissibility as

$$\begin{aligned} \triangle y_j \triangle z_k \boldsymbol{u}_{\alpha,i+1/2,jk} = T_{i+1/2,jk} \lambda_{\alpha,sjk} \big(&P_{\alpha,i+1,jk} - P_{\alpha,ijk} \\ &- g\rho_{\alpha,i+1/2,jk}(D_{i+1,jk} - D_{ijk})\big), \end{aligned} \tag{25}$$

where g is the gravity, D_{ijk} is the depth of cell Ω_{ijk}, $P_{\alpha,ijk}$ is the pressure of the α phase constituent in the cell, $\lambda_{\alpha,sjk}$ is the upwinded mobility, and $\rho_{\alpha,i+1/2,jk}$ is the density averaged across neighboring cells. The mobility term is defined as

$$\lambda_{\alpha,sjk} =: \frac{1}{\mu_\alpha} k_\alpha(S_{w,sjk}) \rho_\alpha(P_{\alpha,sjk}),$$

where s is the upwinded cell index. Velocities and mobilities are similarly defined for the y and z directions.

In order to make calculations for the Jacobian contribution easier, the mobilities are normalized by the appropriate reference density when they are computed. They are computed for interior cells in `aprop.df` and for boundary cells in the subroutine `abdprop`, found in `abdary.df`. Thus the transport contributions to the Jacobian need not be normalized when they are evaulated in `atran3.df`.

3.3 Temporal Integration and Jacobian

The model is advanced in time using the backward Euler method for implicit temporal integration. A possible extension to the current air-water model would be to use an IMPES (implicit pressure, explicit saturation) scheme to advance in time.

The backward Euler method applied to our system gives

$$\phi \frac{S_\alpha^{n+1}\rho_\alpha^{n+1} - S_\alpha^n \rho_\alpha^n}{\triangle t^{n+1}} - \nabla \cdot (\lambda_\alpha^{n+1}(\nabla P_\alpha^{n+1} - \rho_\alpha^{n+1} g \nabla D)) = q_\alpha^{n+1}.$$

Thus at each time step we solve a nonlinear, residual problem given as

$$\mathcal{R}_{\alpha,ijk}^{n+1} = \mathcal{A}_{\alpha,ijk}^{n+1} + \mathcal{T}_{\alpha,ijk}^{n+1} + \mathcal{W}_{\alpha,ijk}^{n+1} = 0. \tag{26}$$

The Jacobian is formulated by differentiating each of these terms with respect to the primary unknowns water pressure P_w and water saturation S_w. Again, one of the main differences in the air-water model and the implicit hydrology model lies here; Jacobian contributions and residuals must be normalized by the appropriate reference densities to maintain proper units in the accumulation term. Thus, given the residual formulation, the Jacobian contributions from each of $\mathcal{A}$, $\mathcal{T}$, and $\mathcal{W}$ must be normalized.

Note that in the air-water model, we solve for the Newton step s as

$$-F'(x)s = F(x)$$

whereas in the implicit hydrology model, the Newton equation is formulated as

$$F'(x)s = -F(x).$$

3.4 Data Structures

The Jacobian for the IPARSv2 modules is stored in the COF data structure, which is described here for completeness. The Jacobian calculations are based on a $7-point$ stencil. The COF data structure is formatted as

$$COF(I, J, K, s, e, p),$$

where s is the stencil identification, e is the equation number (1 for the water equation and 2 for the air equation), and p is the primary unknown (1 for P_w or 2 for S_w). The IJK index is for the discretization cell Ω_{ijk}. For more specific information on the COF data structure, please consult the *IPARS User's Manual* [1] or the documentation included with the code.

The air-water model contains a data structure to maintain the derivatives that are needed as a direct consequence of the compressibility of air. These derivatives are not needed in the implicit hydrology model, because of the small compressibilities of oil and water, but are needed for the implicit black-oil model. Thus the $TRNDAT$ data structure, which contains these derivatives, can be found in the black-oil model as well as the air-water model. The $TRNDAT$ data structure also contains the values for the mobilities. The components of $TRNDAT$ are evaluated at each of the interior and boundary cells in `aprop.df` and `abdary.df`, respectively. The derivatives that are stored in $TRNDAT$ are:

$$\begin{array}{ll}
\dfrac{\partial \rho_w}{\partial S_w} = 0 & \dfrac{\partial \rho_a}{\partial S_w} = \rho_a p_c' \left(\dfrac{1}{p_a} - \dfrac{Z'}{Z} \right) \\
\dfrac{\partial \rho_w}{\partial p_w} = c_w \rho_w & \dfrac{\partial \rho_a}{\partial p_w} = \rho_a \left(\dfrac{1}{p_a} - \dfrac{Z'}{Z} \right) \\
\dfrac{\partial p_w}{\partial S_w} = 0 & \dfrac{\partial p_a}{\partial S_w} = p_c' \\
\dfrac{\partial p_a}{\partial p_w} = 1 & \dfrac{\partial \mu_a}{\partial S_w} = \mu_a' p_c' \\
\dfrac{\partial \mu_a}{\partial p_w} = \mu_a' & \dfrac{\partial \eta}{\partial p_w} = \eta'
\end{array}$$

The actual components of $TRNDAT$ as given in the `.df` files are:

$$\begin{array}{ll}
TRNDAT(I,J,K,1) = \triangle t \dfrac{\lambda_{w,ijk}}{\rho_{w,ref}} & TRNDAT(I,J,K,8) = \dfrac{\partial \phi_{ijk}}{\partial P_{w,ijk}} \\
TRNDAT(I,J,K,2) = \triangle t \dfrac{\lambda_{a,ijk}}{\rho_{a,ref}} & TRNDAT(I,J,K,9) = \dfrac{\partial \mu_{a,ijk}}{\partial S_{w,ijk}} \\
TRNDAT(I,J,K,3) = \dfrac{\triangle t}{\rho_{w,ref}} \dfrac{\partial \lambda_{w,ijk}}{\partial S_{w,ijk}} & TRNDAT(I,J,K,10) = \dfrac{\partial \mu_{a,ijk}}{\partial P_{w,ijk}} \\
TRNDAT(I,J,K,4) = \dfrac{\triangle t}{\rho_{w,ref}} \dfrac{\partial \lambda_{w,ijk}}{\partial P_{w,ijk}} & TRNDAT(I,J,K,11) = \dfrac{\partial \rho_{a,ijk}}{\partial S_{w,ijk}} \\
TRNDAT(I,J,K,5) = \dfrac{\triangle t}{\rho_{a,ref}} \dfrac{\partial \lambda_{a,ijk}}{\partial S_{w,ijk}} & TRNDAT(I,J,K,12) = \dfrac{\partial \rho_{a,ijk}}{\partial P_{w,ijk}} \\
TRNDAT(I,J,K,6) = \dfrac{\triangle t}{\rho_{a,ref}} \dfrac{\partial \lambda_{a,ijk}}{\partial P_{w,ijk}} & TRNDAT(I,J,K,13) = \dfrac{\partial P_{a,ijk}}{\partial S_{w,ijk}} \\
TRNDAT(I,J,K,7) = \dfrac{\partial \phi_{ijk}}{\partial S_{w,ijk}} & TRNDAT(I,J,K,14) = \dfrac{\partial P_{a,ijk}}{\partial P_{w,ijk}}
\end{array}$$

Note that the first 6 elements of $TRNDAT$ are normalized by the appropriate reference densities. Making this computation and storing it here reduces the computational time needed for Jacobian evaluation. The derivatives of air viscosity and of capillary pressure, which are needed to construct certain elements of $TRNDAT$, are determined using table lookups.

3.5 Jacobian Transport Contributions

The transport contributions are evaluated at the half-points, separately for each of the coordinate directions. This calculation is done in `atran3.df`, which contains subroutines for evaluating the transport in each of the $x-$, $y-$, and $z-$ directions. Jacobian transport contributions are calculated for each of the water and air equations by taking derivatives, with respect to each of the primary unknwons, of the discretized flow equation (27)

$$f_w = \lambda_{\alpha,sjk}\left(P_{\alpha,ijk} - P_{\alpha,i-1,jk} - g\rho_{w,i-1/2,jk}\left(D_{ijk} - D_{i-1,jk}\right)\right), \qquad (27)$$

where $\rho_{w,i-1/2,jk} = \dfrac{\rho_{w,ijk} + \rho_{w,i-1,jk}}{2}$.

The mobility terms are upwinded, i.e., if $P_{\alpha,ijk} - P_{\alpha,i-1,jk} < 0$ then the mobility is evaluated at the $i-1,jk$ point; otherwise, it is evaluated at the ijk point. If the mobility is evaluated at the $i-1,jk$ point then the Jacobian contributions for the water equation with respect to water pressure are

$$\frac{\partial f_w}{\partial P_{w,i-1,jk}} = \frac{\partial \lambda_{w,i-1,jk}}{\partial P_{w,i-1,jk}}\left(P_{w,ijk} - P_{w,i-1,jk} - \frac{g}{2}\rho_{w,i-1/2,jk}\left(D_{ijk} - D_{i-1,jk}\right)\right) + \lambda_{w,i-1,jk}\left(-1 - \frac{g}{2}c_w\rho_{w,i-1,jk}\left(D_{ijk} - D_{i-1,jk}\right)\right), \qquad (28)$$

and

$$\frac{\partial f_w}{\partial P_{w,ijk}} = \lambda_{w,i-1,jk}\left(1 - \frac{g}{2}c_w\rho_{w,ijk}\left(D_{ijk} - D_{i-1,jk}\right)\right). \qquad (29)$$

The contribution from (28) is added to $COF(I,J,K,2,1,1)$ and $COF(I-1,J,K,1,1,1)$, and the contribution from (29) isadded to $COF(I,J,K,1,1,1)$ and $COF(I-1,J,K,3,1,1)$.

The Jacobian transport contribution for the water equation with respect to water saturation is, for mobility upwinded at the $i-1$ cell,

$$\frac{\partial f_w}{\partial S_{w,i-1,jk}} = \frac{\partial \lambda_{w,i-1,jk}}{\partial S_{w,i-1,jk}}\left(P_{w,ijk} - P_{w,i-1,jk} - g\rho_{w,i-1/2,jk}\left(D_{ijk} - D_{i-1,jk}\right)\right), \qquad (30)$$

and

$$\frac{\partial f_w}{\partial S_{w,ijk}} = 0. \tag{31}$$

The contribution from (30) is added to $COF(I, J, K, 2, 1, 2)$ and $COF(I-1, J, K, 1, 1, 2)$, while the contribution from (31) is added to $COF(I, J, K, 1, 1, 2)$ and $COF(I-1, J, K, 3, 1, 2)$.

3.6 Jacobian Well Contributions

Given the equation for the well flux, the Jacobian contribution from the well term with respect to water pressure is

$$\frac{\partial q_w}{\partial P_w} = GLK \left[\frac{\partial \rho_w}{\partial P_w} \frac{k_{rw}}{\mu_w} \left(P_{wb} - P_w - \rho_w g \nabla D \right) + \frac{k_{rw} \rho_w}{\mu_w} \left(-1 - \frac{\partial \rho_w}{\partial P_w} g \nabla D \right) \right], \tag{32}$$

where ∇D is the difference in depth. The contribution with respect to water saturation is

$$\frac{\partial q_w}{\partial S_w} = GLK \frac{\rho_w}{\mu_w} \frac{\partial k_{rw}}{\partial S_w} \left(P_{wb} - P_w - \rho_w g \nabla D \right). \tag{33}$$

For a water injection well, we assume that $k_{rw} = 1$. Thus, if the mobility is upwinded, we would have no contribution to the Jacobian.

In the file `abwell.df`, the contribution is given as

$$COF(I, J, K, 1, 1, 1) = COF(I, J, K, 1, 1, 1) - \text{cdt} * \text{elecons} \left(\frac{\text{rbw}}{\mu_w} - \frac{c_w * \text{rbw}}{\mu_w} * \text{dwp} \right), \tag{34}$$

where

cdt = cvf * $\triangle t$

elecons = elegeom * elelen * eleperm

rbw = $k_{rw,ijk}$

dwp = $P_{wb} - P_{w,ijk} - g\rho_{w,ijk}$ (eledep $- D_{ijk}$)

The wellbore pressure is calculated using the bottom hole pressure provided on input and is adjusted as

$$P_{wb} = BHP + \rho_{w,wb} g \left(\text{eledep} - \text{depbot} \right).$$

4 Boundary Conditions

The boundary conditions have been implemented similarly to the implicit hydrology model. Details on the boundary conditions for that model may

be found in [13]. The boundary calculations are found in `abdary.df`, which contains routines for reading input boundary data, evaluating properites of $TRNDAT$ on the boundary, and computing the transport Jacobian contributions.

Currently, the user can specify 6 types of boundary conditions:

1. water pressure, water saturation
2. water pressure, air saturation
3. air pressure, water saturation
4. water pressure, air pressure
5. water pressure, water saturation at a reference depth
6. water and air flux.

The boundary conditions can be temporally dependent. For instance, one may have a Dirichlet condition at time 0 and then change the boundary to Neumann at time t. As in the implicit hydrology model, no-flow boundary conditions are the default.

The input boundary information is translated to boundary data in the primary unknowns. That is, if the user specifies air pressure and water saturation on the boundary, then the capillary pressure is found using the water saturation and water pressure is obtained as a result. Once the given input data has been translated to the primary unknowns, the elements of $TRNDAT$ are updated for the boundary cells, and the transport contributions to the Jacobian are calculated.

The transport contributions are calculated for each of the $x+$, $x-$, $y+$, $y-$, $z+$, and $z-$ directions, depending on the location of the boundary. The mobilities are still upwinded as in the interior. When calculating the $x+$ contribution, the upwinding criterion is the difference in

$$P_{\alpha,ijk} - P_{\alpha,i-1,jk},$$

whereas for $x-$ the criterion is

$$P_{\alpha,i+1,jk} - P_{\alpha,ijk}.$$

The fluxes across the boundary are also computed and added to the total mass balance.

5 Numerical Results

We provide numerical results for several test problems in this section. The first numerical example is the Buckley-Leverett problem, a standard test problem for multiphase flow. The second problem is a two-dimensional test problem first presented in [15]. The third test problem is a flow problem around an oxbow bend, which is used to demonstrate the capabilities of the boundary conditions.

5.1 Buckley-Leverett Problem

The Buckley-Leverett problem given here was developed using the original hydrology data set file developed by Dr.Małgorzata Peszyńska. This is a standard one-dimensional test case for multiphase flow problems.

The model simulates density-driven, i.e., zero capillary pressure, flow across a reservoir of length $1000[ft]$ that is $10[ft]$ deep. Dirichlet water pressure and water saturation conditions are imposed on each end of the computational domain. The water pressure in the reservoir is initialized at $500[psi]$ at a depth of $5[ft]$, with water saturation at 0.2. The permeability is 500 in the direction of flow, and 1 in the other directions.

The water pressure is set to $550[psi]$ at the left end of the reservoir and $300[psi]$ at the right end, i.e., $1000[ft]$. The water saturations are set to 0.4 and 0.2, respectively.

Figures (1), (2), and (3) show the results for flow in each of the x, y, and z directions. None of the flows were gravity driven; i.e., the gravity direction was always orthogonal to the direction of flow. The capillary pressure and permeability curves used for this problem are given in Figures (4) and (5).

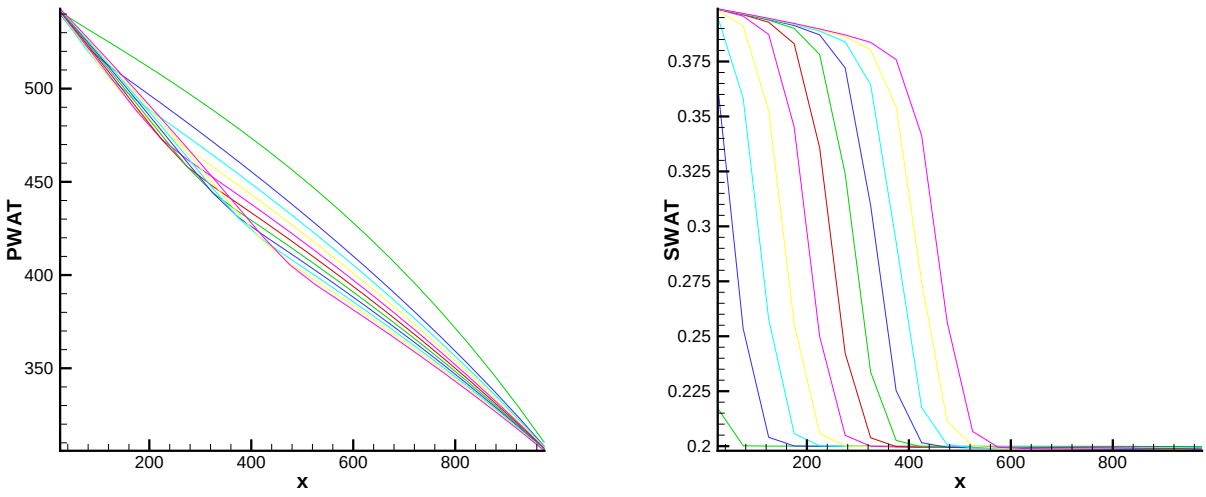

Figure 1: Flow in the x-direction (without gravity)

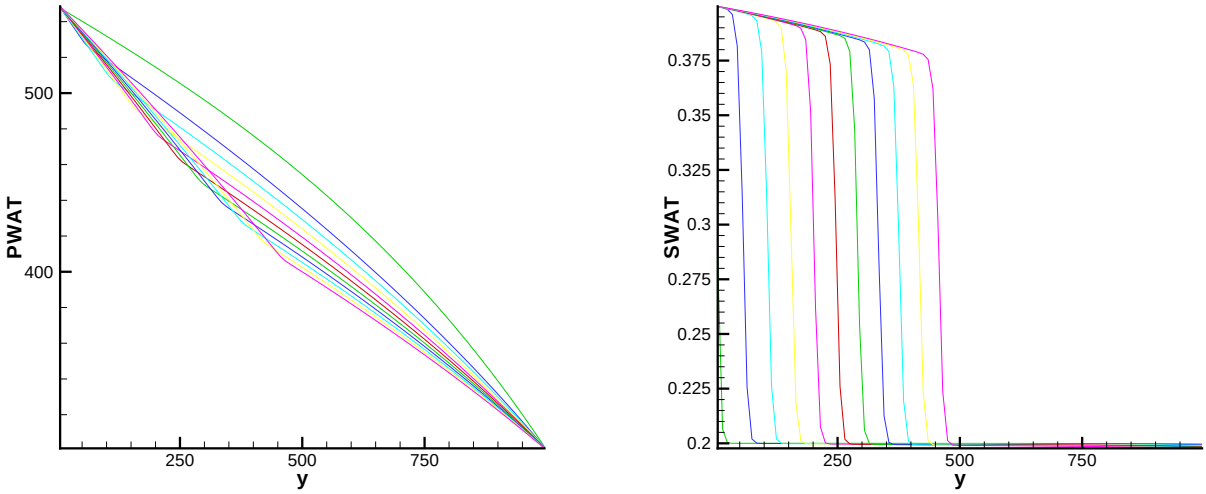

Figure 2: Flow in the y-direction

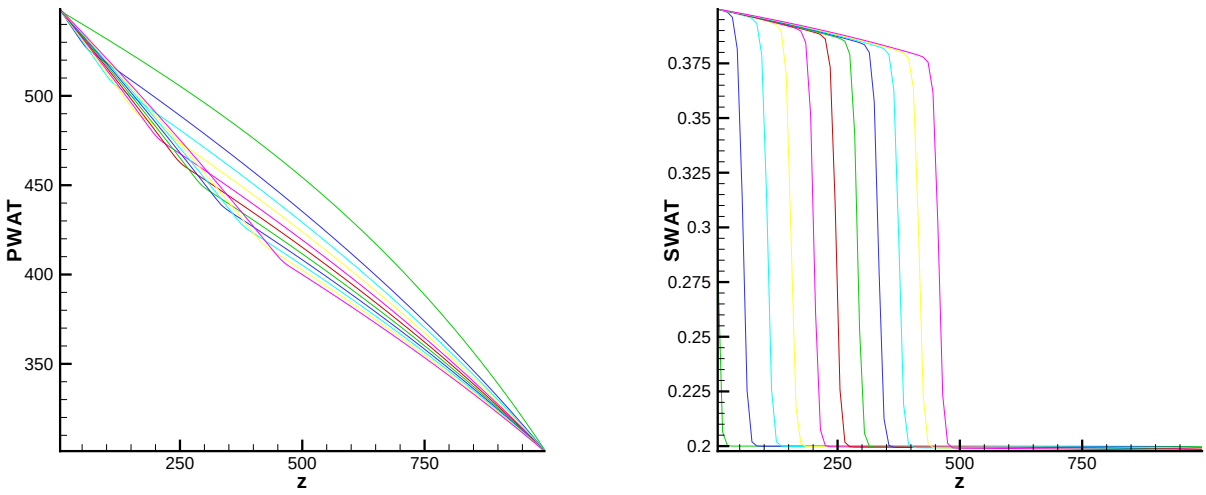

Figure 3: Flow in the z-direction

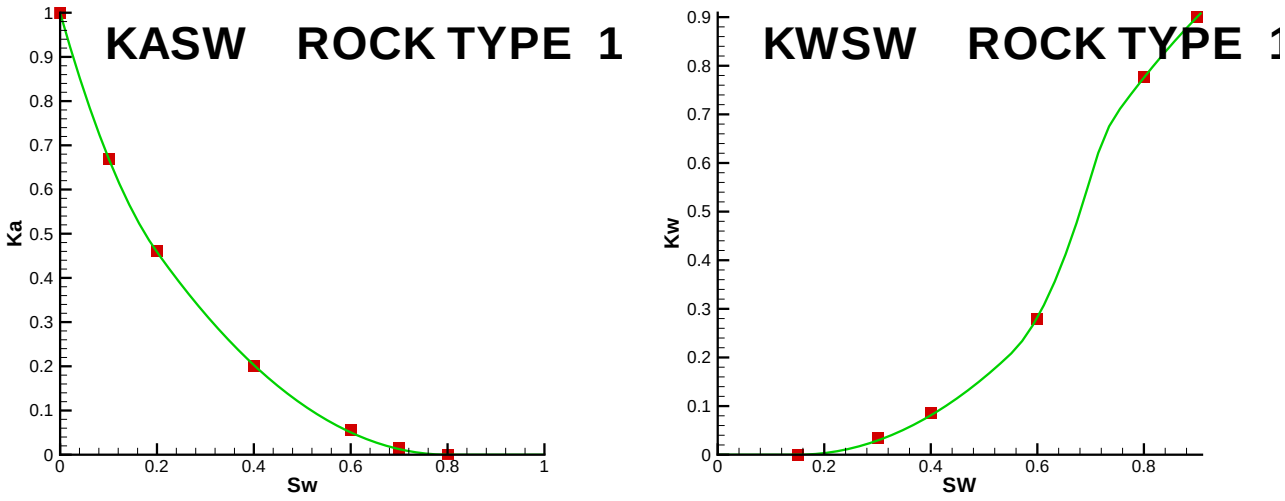

Figure 4: Permeability data

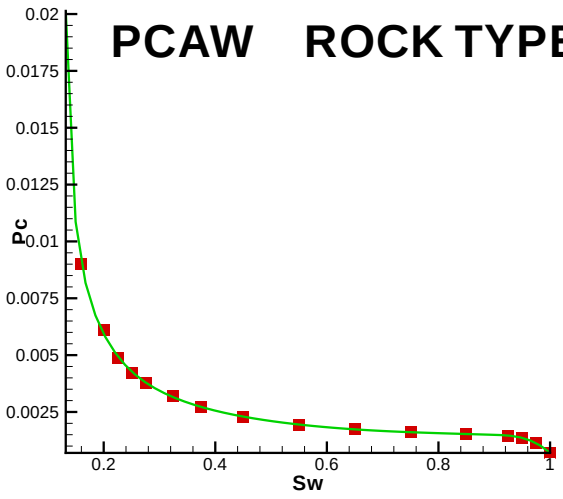

Figure 5: Capillary pressure data

5.2 Vauclin Problem

This test problem is used to validate two-dimensional results of the air-water model. It is based on an empirical study published in [15], and the numerical results in that study were performed using the single-phase Richards' equation model. This is a transient problem, modeling a vadose zone water table recharge. Note that the α parameter in [15] is reported incorrectly; it should be $\alpha = 40,000$.

The test domain is a sand filled tank with trenches on either end. The tank is $600[cm]$ wide, $180[cm]$ high, and $10[cm]$ thick. The tank is initially filled with water and allowed to drain to a height of $60[cm]$ of water. After the reservoir has stabilized, the tank is infiltrated with water through $100[cm]$ along the top edge. A pumping mechanism is used in the trench to maintain the $60[cm]$ of water. Thus after 8 hours of simulation, the water saturation profile resembles Figure (6). The tank is symmetric, however, so only one-half of the tank needs to be modeled. The porosity of the material in the tank is given as 0.30, and the data provided indicate that this is a coarse sand.

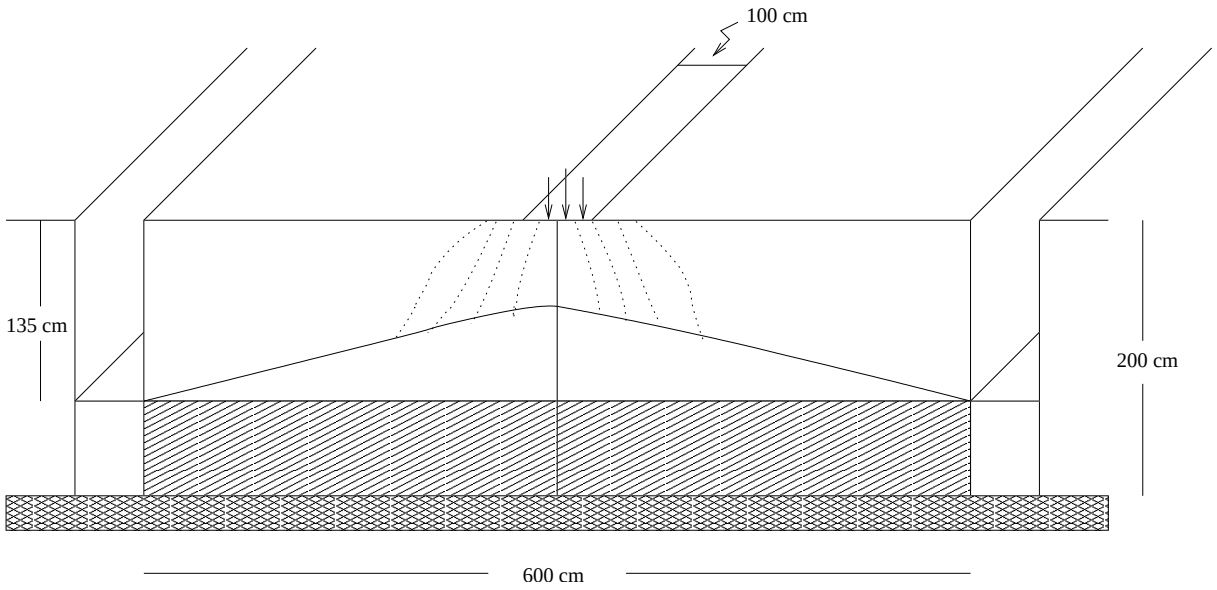

Figure 6: Experimental Schematic of Vauclin Problem

In order to formulate this problem in terms of our two-phase model, we needed to determine relative permeability and capillary pressure curves. The permeability curves were determined using the fitted conductivity data, and

the capillary pressure curves were determined by examining the pressure-saturation profile provided in the paper. Dr.J̃ohn A.W̃heeler developed the capillary pressure curve and the input datefile used for our simulation.

The dimensions of the problem were expanded for our simulation. Our computational domain is $400[ft]$ long by $20[ft]$ high and $20[ft]$ wide. We included a ditch in our model to simulate the trenches described in the paper. We reduce the capillary pressure by two orders of magnitude in the ditch to help maintain the constant water head. We initialize our problem with a water pressure of $503.94511[psi]$ and a water saturation of 0.5 at a depth of $11[ft]$.

We simulate the pumping and infiltration mechanisms by using wells in the trench and along the top $3[ft]$ of the tank. We actually use two production wells in the trench; one production well is located $3[ft]$ from the top, and the second production well is $3[ft]$ from the bottom. The top well produces air, and the bottom well produces water to maintain the head value in the trench. Both wells are bottom hole pressure specified wells, with the pressure for the top production well given as $501.499[psi]$ and the pressure at the bottom well given as $507.499[psi]$. The water injection well runs $3[ft]$ across the top of the reservoir, and the injection rate is given as $499.7[psi]$ initially, increasing linearly to $510[psi]$ after 10 days of simulation. Thereafter the pressure is held constant.

Figures (7) and (8) show the water pressure and saturation contours at day 1 of the simulation. The water table is clearly defined, and infiltration has begun in the top left corner of the domain.

Pressure and saturation contours after 301 days of simulation are given in Figures (9) and (10). The water table remains constant near the outflow boundary, which is consistent with the empirical results.

After 701 days of simulation, we see the recharge area beginning to fill in Figures (11) and (12).

Finally, after 1901 days of simulation, we see in Figures (13) and (14) that the water table has maintained its height at the outflow boundary, and that the recharge area is nearly linear.

There is more work to be done with the Vauclin problem. For instance, while the paper reports an almost constant porosity throughout the porous medium, their empirical results indicate this may not be the case. Thus, simulating the environment with constant porosity may give results that do not match well with their data. In addition, Fred Tracy of ERDC had mentioned running the problem with a steeper capillary pressure curve. He has had trouble using FEMWATER to simulate the Vauclin problem as the Picard iterations do not converge, and that is why he brought this problem

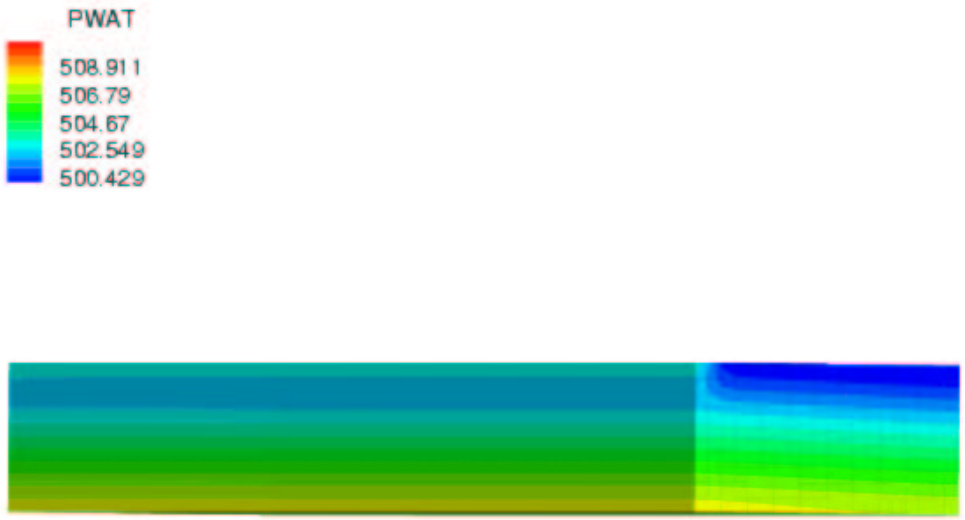

Figure 7: Water pressure contours for Vauclin problem, Day 1

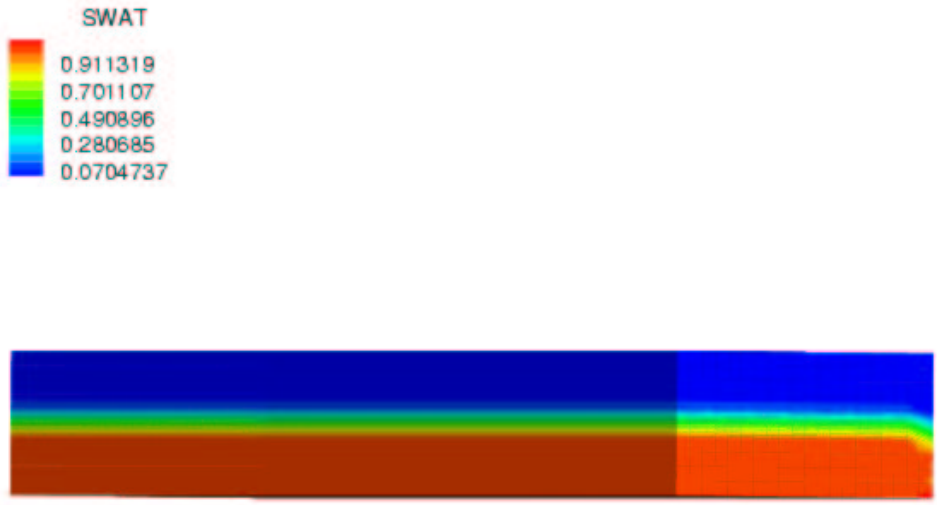

Figure 8: Water saturation contours for Vauclin problem, Day 1

to our attention. While we did observe some failures of the nonlinear solver

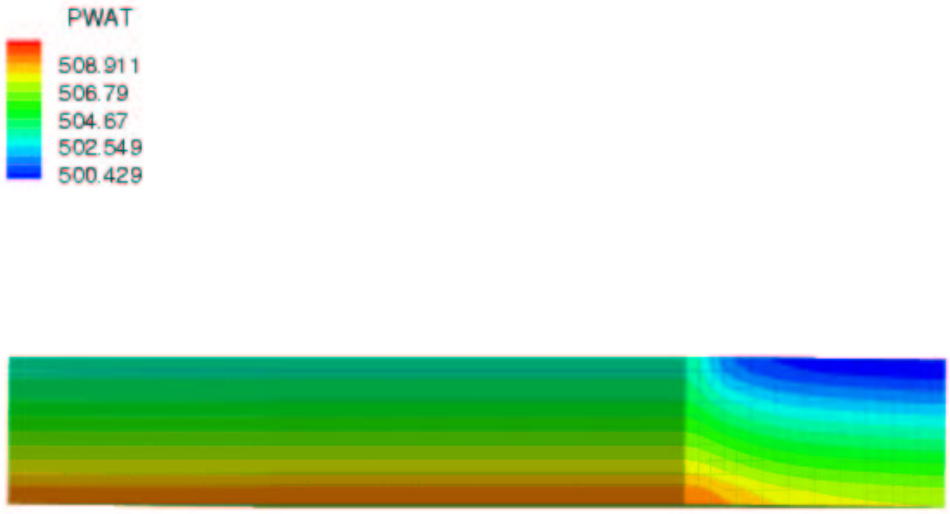

Figure 9: Water pressure contours for Vauclin problem, Day 301

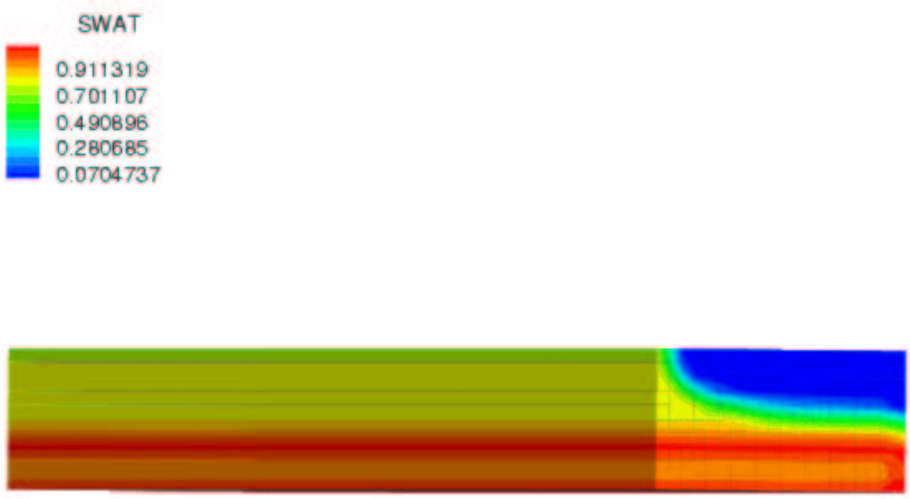

Figure 10: Water saturation contours for Vauclin problem, Day 301

to converge, this did not occur with enough frequency to cause the program

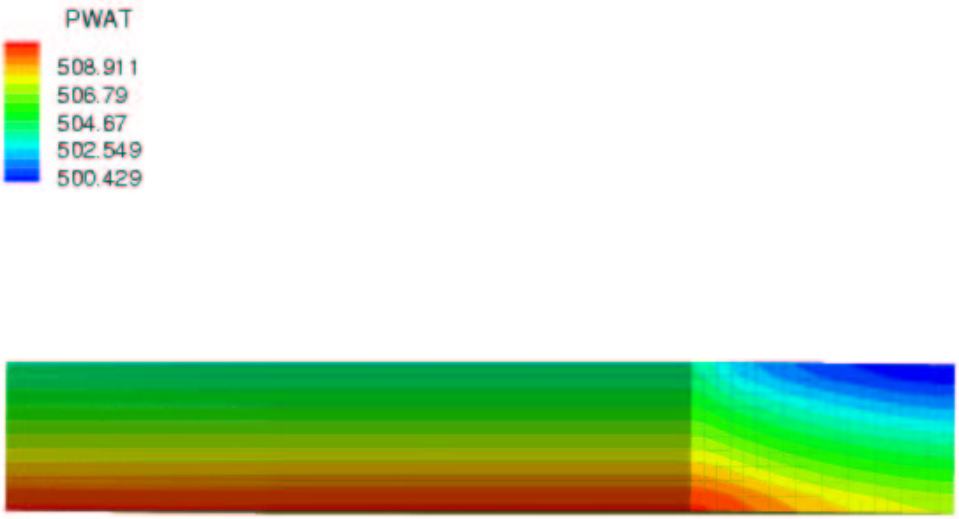

Figure 11: Water pressure contours for Vauclin problem, Day 701

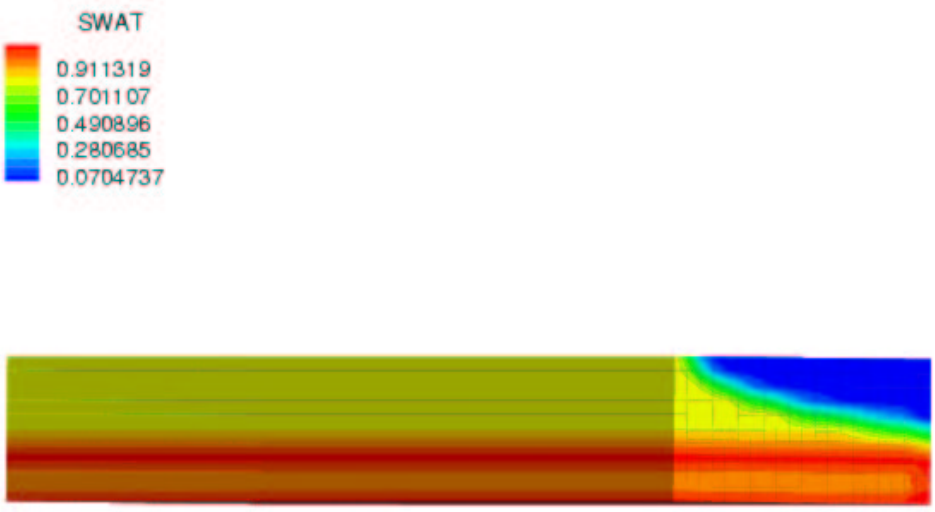

Figure 12: Water saturation contours for Vauclin problem, Day 701

to halt.

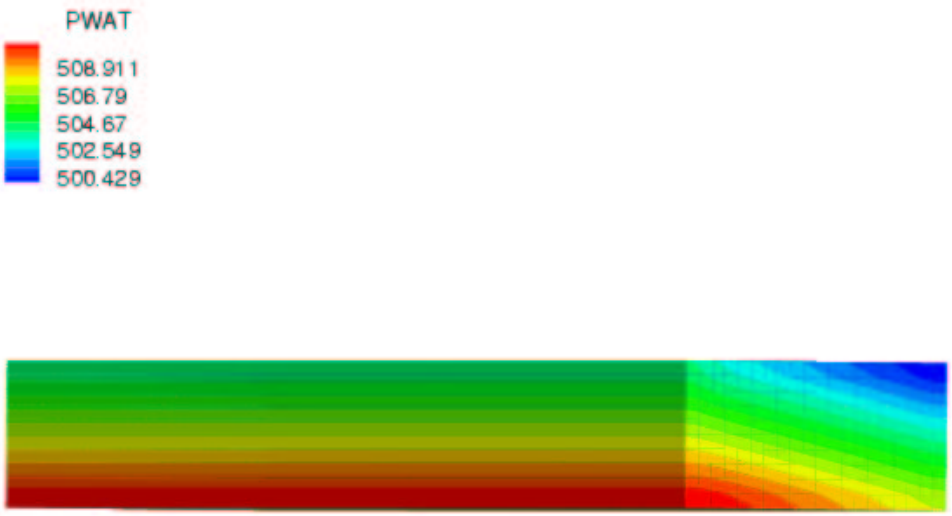

Figure 13: Water pressure contours for Vauclin problem, day 1901

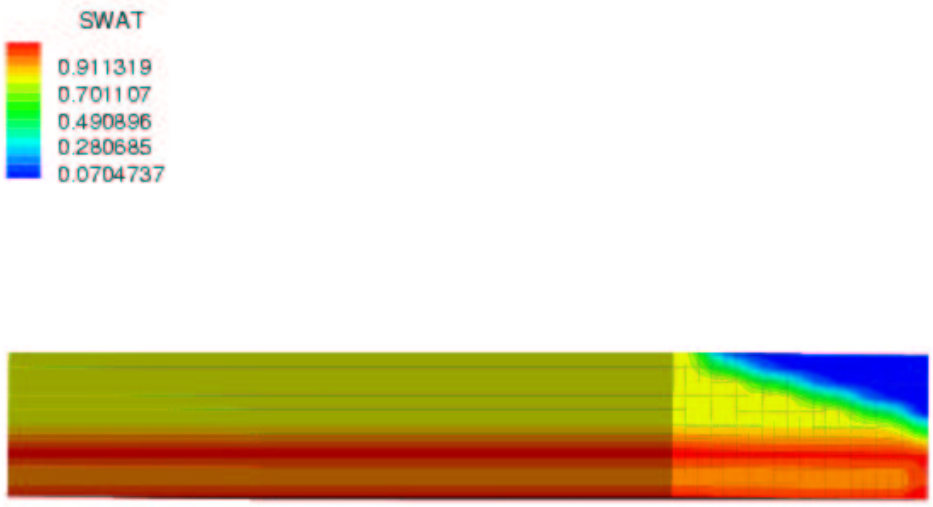

Figure 14: Water saturation contours for Vauclin problem, Day 1901

5.3 Oxbow Problem

The oxbow problem was developed by Dr.John A.Wheeler, with modifications for general boundary conditions by Dr.Małgorzata Peszyńska. The reservoir contains a horseshoe shaped bend, and Dirichlet conditions are initially imposed at the Γ_1, Γ_2, and Γ_3 boundaries, depicted in Figure (15). After 20 days of simulation, no-flow boundaries are imposed on Γ_1 and Γ_3, and after 200 days of simulation, another Dirichlet condition is imposed on Γ_4.

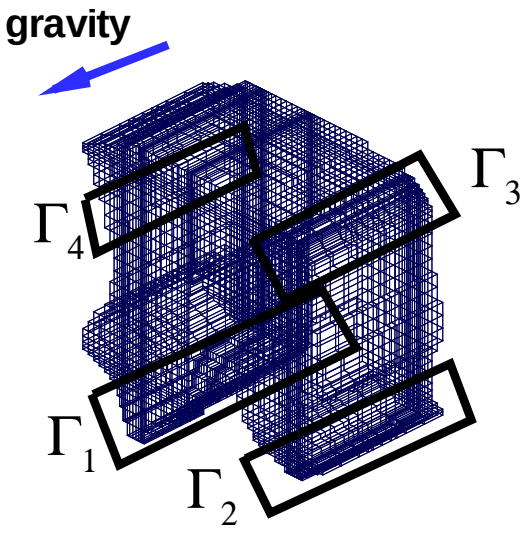

Figure 15: Computational domain for oxbow problem

The mesh contains 19793 active grid elements, and has dimensions $13.1[ft] \times 1759.9[ft] \times 919.9[ft]$. The simulation is run for 601 days, and the water pressure and water saturation contours are given in Figures (16) - (22). The permeability in the x direction is 25 everywhere, except in layers 4 and 7, where it is 5 and 3, respectively. Similarly, the permeability in the y direction is 200 everywhere, except in layers 4 and 7, where it is 30 and 40, respectively. The porosity is also significantly lower in these layers. The reservoir was initialized with a water pressure of $600[psi]$ and a water saturation of 0.35 at a depth of $1000[ft]$.

All of the Dirichlet conditions imposed on the boundary were water pressure / water saturation conditions. On Γ_1, the water pressure was initially set to $700[psi]$ and the water saturation to 0.7, and on Γ_2 and Γ_3, the water

pressure is 700[psi] with a water saturation of 0.6, creating an inflow at each of these boundaries. There is a no-flow boundary at Γ_4, which is changed to a water pressure of 500[psi] and water saturation of 0.25 after 200 days.

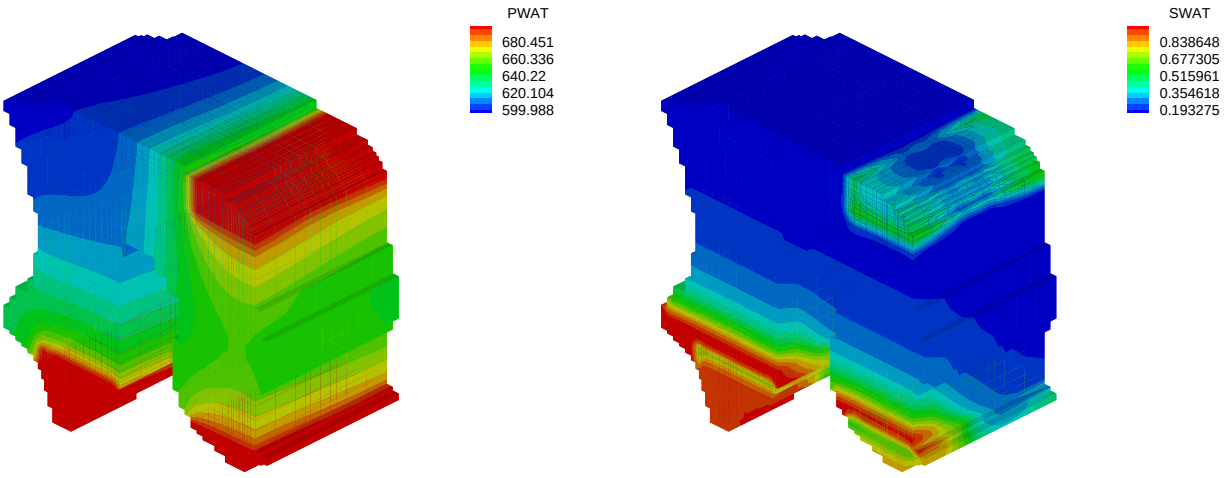

Figure 16: Water pressure and saturation contours for oxbow problem, day 1

Figures (18) and (19) show the pressure and saturation contours just before and after the Dirichlet condition on the left boundary is activated. This demonstrates the temporally dependent boundary capability.

5.4 Parallel Results

In this section, we provide numerical results for a parallel run, reusing the oxbow dataset. Note that both the linear and nonlinear residuals differ between sequential and parallel runs. The data for the sequential run shown in this section were obtained on one node of the Longhorn Beowulf cluster. Earlier sequential runs were made on a Linux workstation (`hubble.ticam.utexas.edu`), and the residuals did not differ between this and one node of the cluster. Parallel and sequential runs have also been made using the implicit hydrology model and a similar test file (i.e., the air phase pressures are made oil phase pressures, and the capillary pressure and relative permeability data are identical), with results much the same. That is, the parallel and sequential linear and nonlinear residuals differ. Results

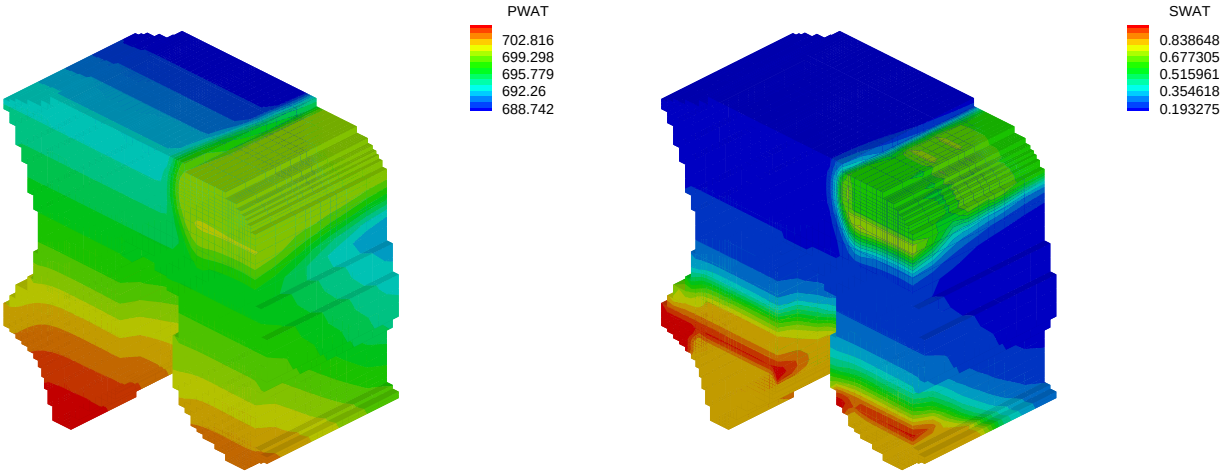

Figure 17: Water pressure and saturation contours for oxbow problem, day 121

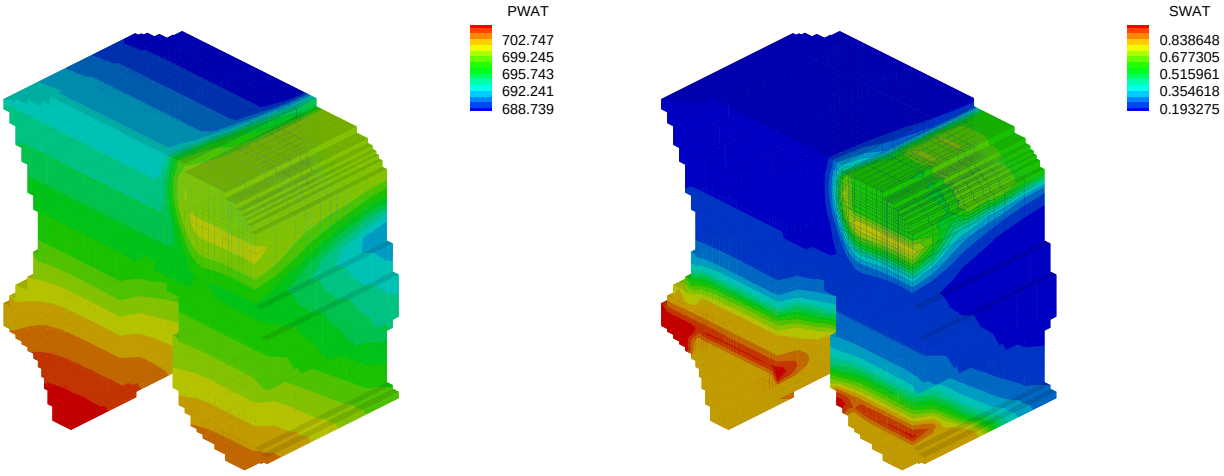

Figure 18: Water pressure and saturation contours for oxbow problem, day 181

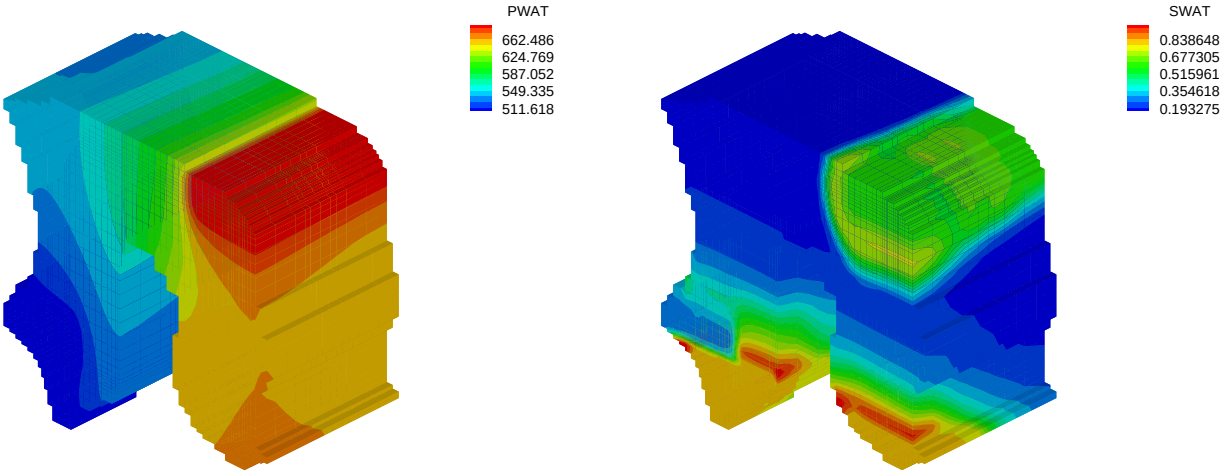

Figure 19: Water pressure and saturation contours for oxbow problem, day 221

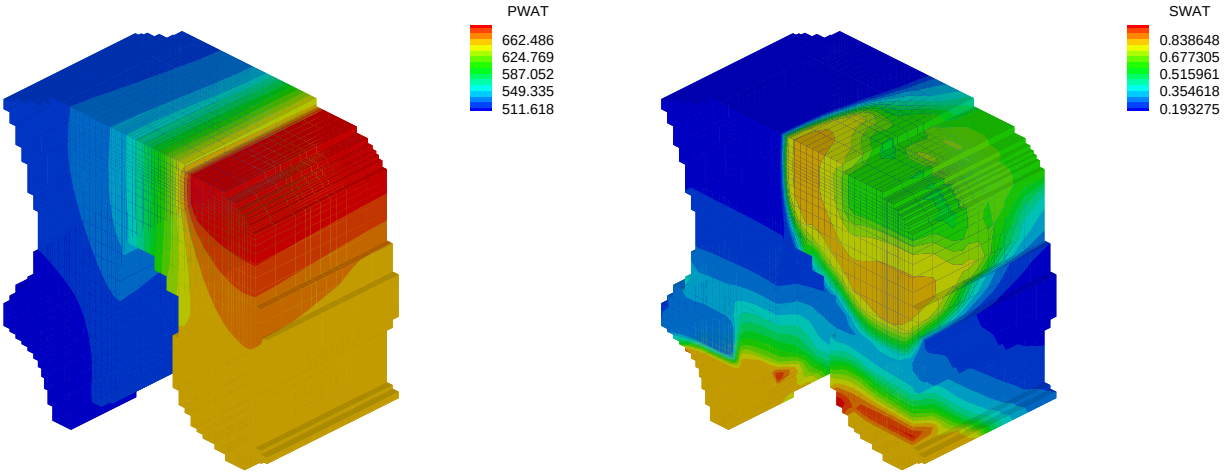

Figure 20: Water pressure and saturation contours for oxbow problem, day 301

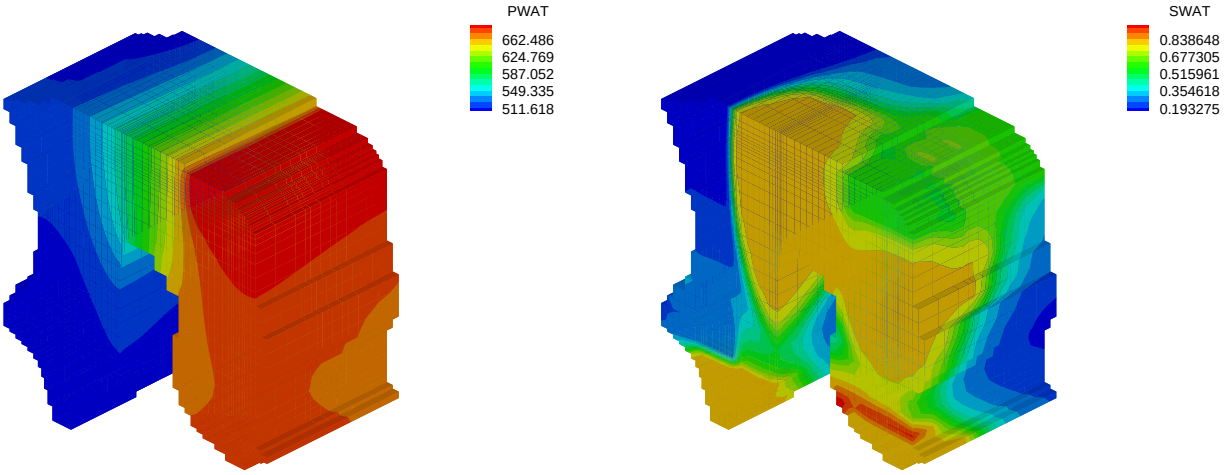

Figure 21: Water pressure and saturation contours for oxbow problem, day 461

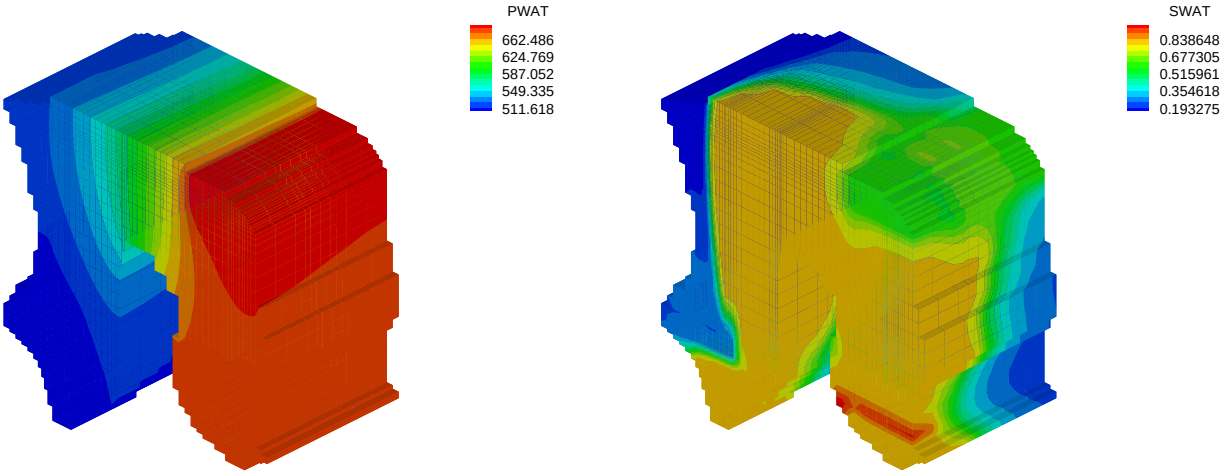

Figure 22: Water pressure and saturation contours for oxbow problem, day 601

from the first two time steps of a 50 day simulation using a single node of the longhorn cluster are given below.

```
 0 NewtRes26802.     33108.
# of GMRES itns =            14  res.err=    7.999146
 1 NewtRes262.40     8692.1
# of GMRES itns =            15  res.err=   8.9775123E-02
 2 NewtRes53.847     2806.8
# of GMRES itns =            16  res.err=   1.7494552E-02
 3 NewtRes29.662     748.21
# of GMRES itns =            18  res.err=   7.3507749E-03
 4 NewtRes12.817     169.43
# of GMRES itns =            18  res.err=   3.1941168E-03
 5 NewtRes2.3298     30.161
# of GMRES itns =            19  res.err=   2.4069681E-04
 6 NewtRes.12645     1.2769
# of GMRES itns =            18  res.err=   2.5556827E-05
 7 NewtRes.17735E-02 .97135E-02
# of GMRES itns =            18  res.err=   2.0137122E-07
STEP   1 TIME    1.20 WAT 1.0000000 AIR 1.0000000 NEWT  8 LINR 17.00
Final NewtRes: .15407E-06 .27126E-06

 0 NewtRes147.62     412.36
# of GMRES itns =            20  res.err=   6.9201112E-02
 1 NewtRes31.507     436.83
# of GMRES itns =            18  res.err=   1.3142820E-02
 2 NewtRes5.1775     226.63
# of GMRES itns =            17  res.err=   1.6179981E-03
 3 NewtRes1.1019     41.280
# of GMRES itns =            15  res.err=   3.1186911E-04
 4 NewtRes.11683     3.2787
# of GMRES itns =            15  res.err=   1.1699570E-05
 5 NewtRes.71025E-03 .16892E-01
# of GMRES itns =            16  res.err=   9.1500951E-08
 6 NewtRes.37841E-06 .29772E-05
# of GMRES itns =            11  res.err=   8.5738048E-09
STEP   2 TIME    2.64 WAT 1.0000000 AIR 1.0000000 NEWT  7 LINR 16.00
Final NewtRes: .10996E-09 .70953E-08
```

This second set of output data is from the same simulation, but run on 2 processors. Note that the residuals are close to the residuals from the

single-processor case, but they are not exact.

```
 DATA BROADCAST FROM PROCESSOR 0,         5502  BYTES
  0 NewtRes26802.     33108.
 # of GMRES itns =            20  res.err=    7.745908
  1 NewtRes262.46     8686.0
 # of GMRES itns =            22  res.err=   9.7680822E-02
  2 NewtRes53.847     2803.8
 # of GMRES itns =            22  res.err=   2.2004491E-02
  3 NewtRes29.651     747.24
 # of GMRES itns =            24  res.err=   6.8437341E-03
  4 NewtRes12.816     169.31
 # of GMRES itns =            23  res.err=   3.7993274E-03
  5 NewtRes2.3298     30.160
 # of GMRES itns =            25  res.err=   2.4731804E-04
  6 NewtRes.12643     1.2771
 # of GMRES itns =            23  res.err=   1.8971896E-05
  7 NewtRes.17741E-02 .97162E-02
 # of GMRES itns =            20  res.err=   2.3028251E-07
 STEP   1 TIME    1.20 WAT 1.0000000 AIR 1.0000000 NEWT  8 LINR 22.38
 Final NewtRes: .15390E-06 .27393E-06

  0 NewtRes147.62     412.36
 # of GMRES itns =            28  res.err=   6.8849027E-02
  1 NewtRes31.507     436.84
 # of GMRES itns =            23  res.err=   1.6458998E-02
  2 NewtRes5.1769     226.63
 # of GMRES itns =            23  res.err=   2.4060102E-03
  3 NewtRes1.1020     41.308
 # of GMRES itns =            23  res.err=   3.0282364E-04
  4 NewtRes.11696     3.2784
 # of GMRES itns =            20  res.err=   2.1121243E-05
  5 NewtRes.71005E-03 .16889E-01
 # of GMRES itns =            21  res.err=   1.2574947E-07
  6 NewtRes.37842E-06 .29761E-05
 # of GMRES itns =            15  res.err=   9.6606554E-09
 STEP   2 TIME    2.64 WAT 1.0000000 AIR 1.0000000 NEWT  7 LINR 21.86
 Final NewtRes: .10170E-09 .78553E-08
```

The parallel visualization capabilities have also been tested. The following images were obtained after a parallel run on 2 processors. These images

can be compared with the sequential images generated in the previous section for the oxbow test case. The figures give water pressure and saturation contours for the oxbow problem run in parallel on the same days as the previous images.

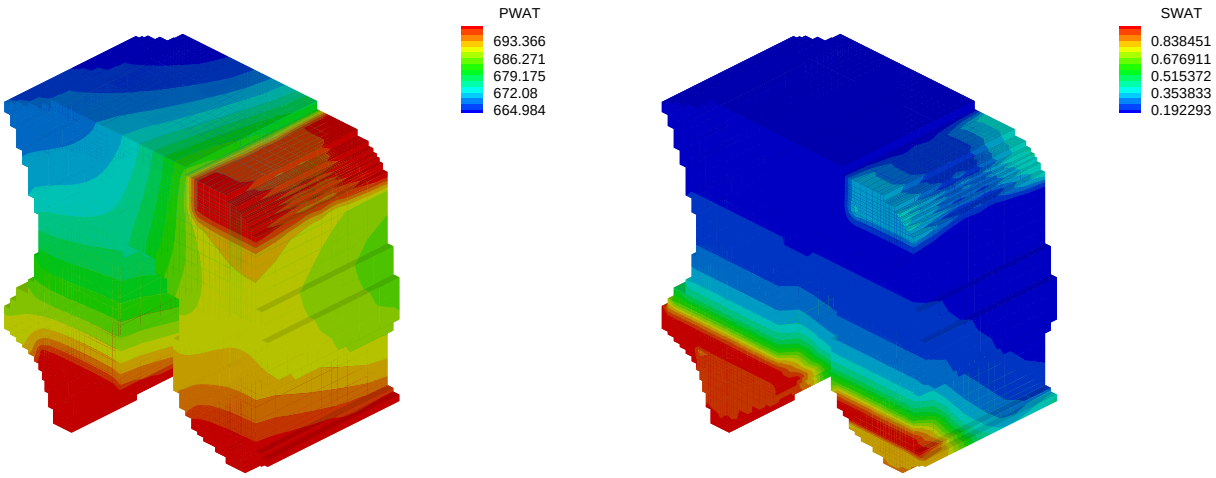

Figure 23: Water pressure and saturation contours for parallel oxbow problem, day 1

5.5 Multiblock Results

These results have not been completed. The MACE interface was not compiling at the time this was written. However, all multiblock keywords have been entered into the air-water files, in exactly the same manner as in the hydrology model.

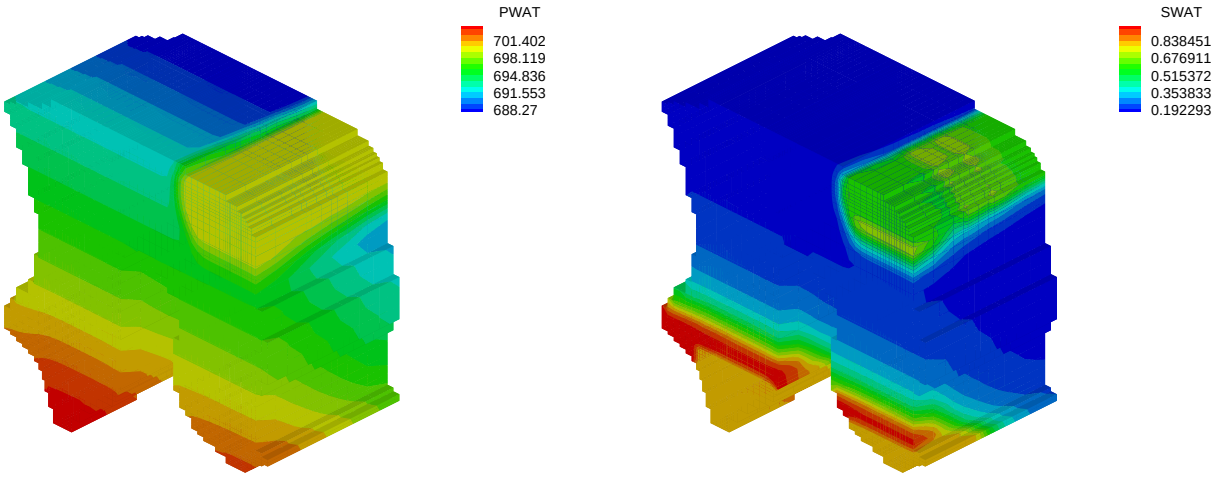

Figure 24: Water pressure and saturation contours for parallel oxbow problem, day 381

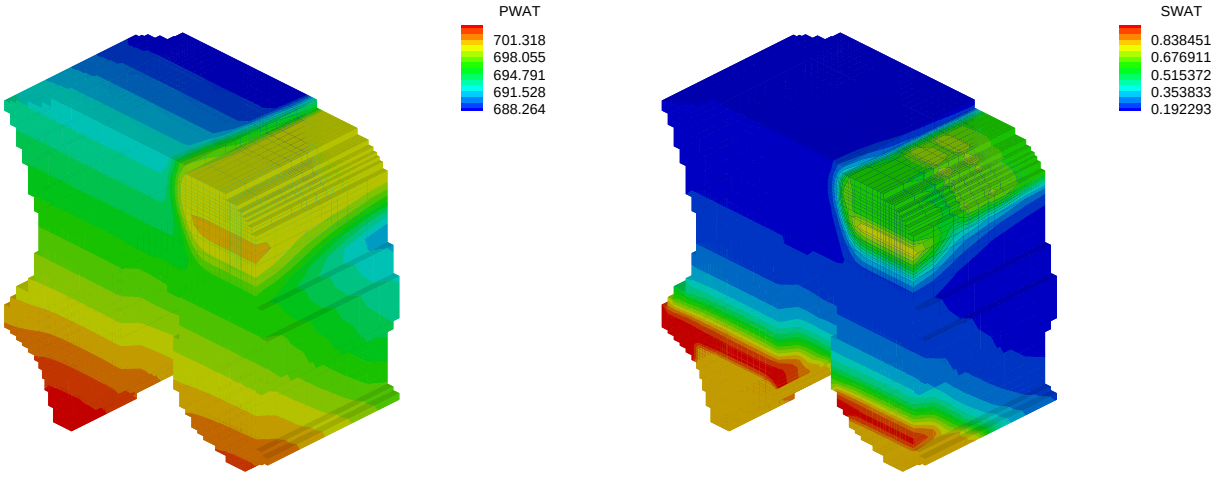

Figure 25: Water pressure and saturation contours for parallel oxbow problem, day 181

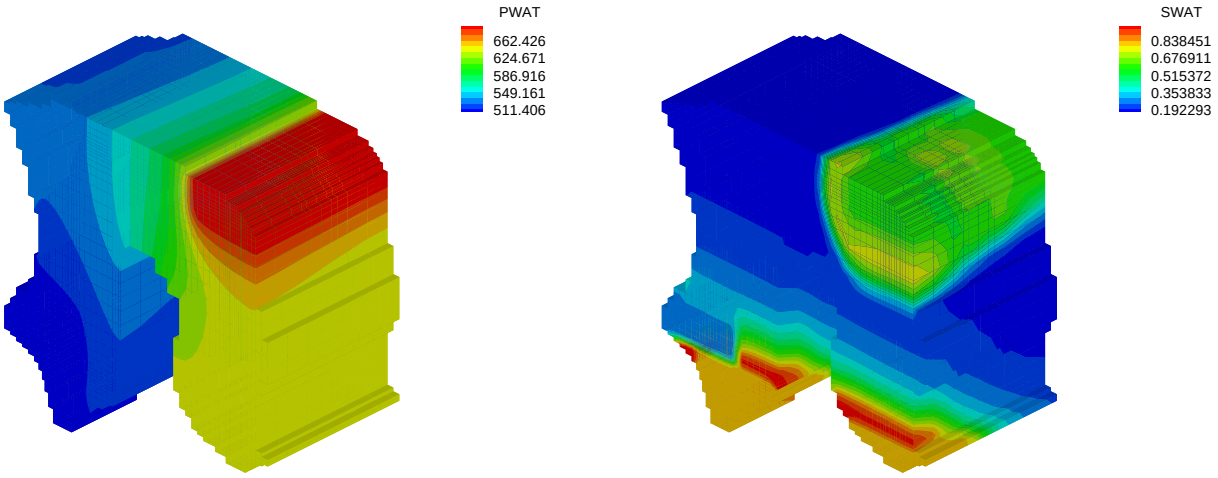

Figure 26: Water pressure and saturation contours for parallel oxbow problem, day 221

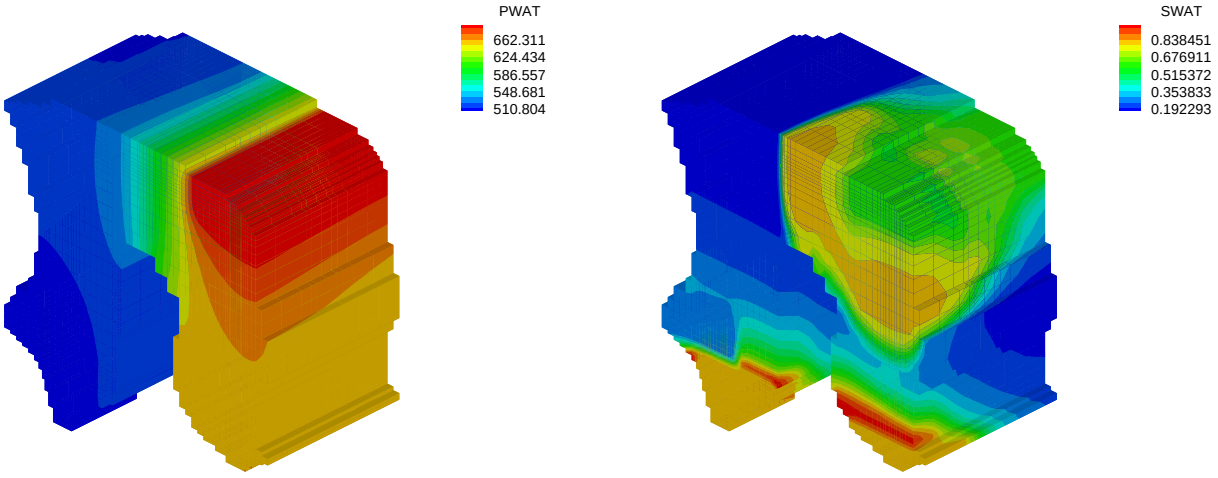

Figure 27: Water pressure and saturation contours for parallel oxbow problem, day 301

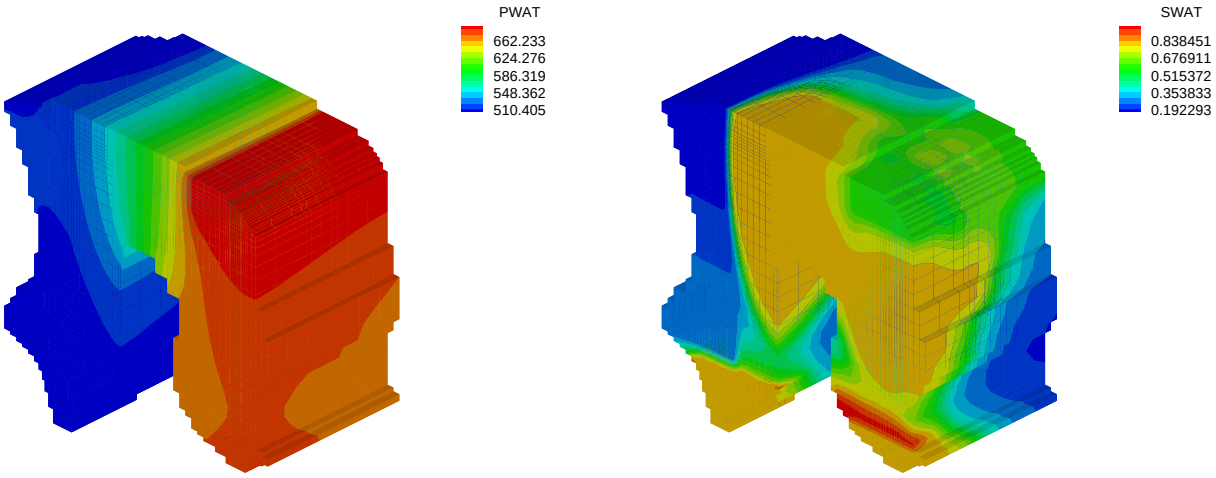

Figure 28: Water pressure and saturation contours for parallel oxbow problem, day 461

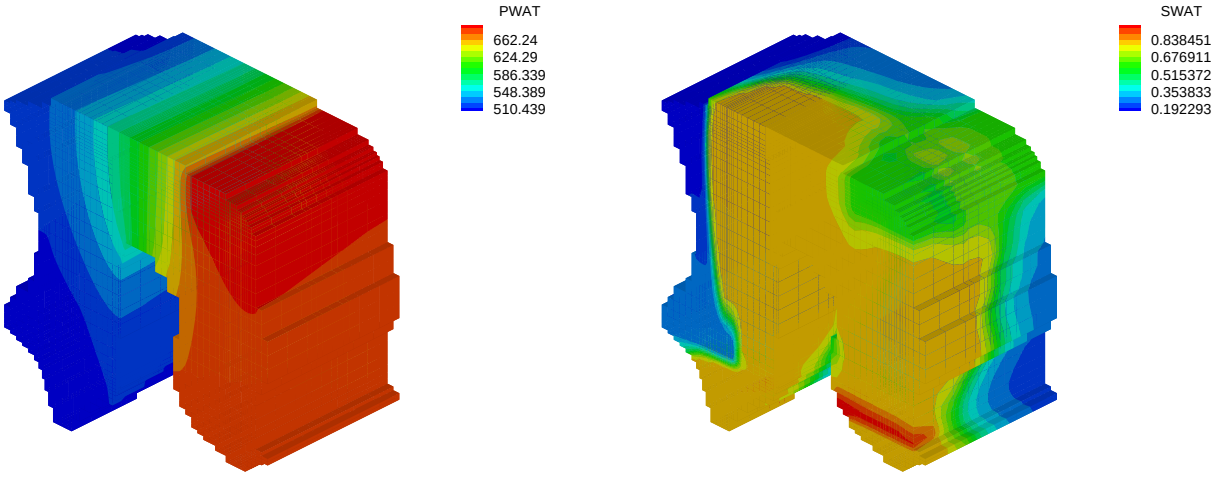

Figure 29: Water pressure and saturation contours for parallel oxbow problem, day 601

6 Conclusions and Future Work

Future work for the air-water model would include incorporating a reactive-transport component, reducing the model so that Richards' Equation is also incorporated into IPARS, and using different temporal integration schemes. As mentioned earlier, IMPES methods might work well for these problems, although there are concerns as noted in [14]. Kirchoff transformations have also been successfully applied to the two-phase hydrology model [3], and should be studied for the air-water model.

7 Acknowledgements

The author wishes to thank Dr. Małgorzata Peszyńska, Dr. John A. Wheeler, and Dr. Mary F. Wheeler for their help in writing and testing the air-water module.

8 Appendix

This appendix contains 6 sections. §8.1 contains information on adjusting the IPARS makefiles to run the air-water model. §8.2 and §8.3 have information on the files and subroutines in the model. The differences between the original air-water model and the second version are listed in §8.2, and §8.3 contains a list of all of the files in the `airv2` subdirectory along with a brief description of their function and the subroutines in the file. The input files for the numerical examples are provided in §8.4, §8.5, and §8.6, and the output file for the oxbow problem is given in §8.7.

8.1 Running the Code

A new makefile, `airv2.mak`, was written for AIRV2. This has been included in the `make/modular` subdirectory. However, a new `.siz` file has not been created; therefore, when adjusting the `ipars.siz` file to run AIRV2, use the existing `air.siz` file.

This version of the air-water model has only been run in parallel on longhorn, the 64-node beowulf cluster. To run this code in parallel, use either `x2.mak` or `x2-eth.mak`, depending on whether or not you wish to use the ethernet connection on longhorn. The user also needs to include the `beo.siz` file and `frame36.siz` file in `ipars.siz`. The longhorn directory is `/longhorn/<userid>`, and this directory is currently mounted automatically only from `x2` and `cowboy`. The user should compile the code on `x2` and copy the executable to the `/longhorn/<userid>` directory. To access the cluster, first log onto `cowboy`, then `ssh` to any of the cluster nodes. Interactive jobs using the ethernet connection are executed on the cluster node by using the following command:

```
/usr/mpich-eth/bin/mpirun -np <no. of processors> -machinefile
<hostfile> ipars
```

where the `<hostfile>` is a listing of nodes that may be used to run the job. For instance, if the base node is `cow49`, and the user wishes to make 8 nodes available for processing, the `<hostfile>` might look like

```
cow49
cow50
cow51
cow52
cow53
```

cow54
cow55
cow56
cow57

The user should check the availability of cluster nodes on the secure website

https://cowboy.ticam.utexas.edu/

before beginning any parallel job. This page contains node usage information for both longhorn and cronus.

8.2 List of Changes from AIR to AIRV2

Most of the files in the second version of the air-water model are alterations of the file from the initial version of the air-water model. As stated earlier, the alterations and additions have been based on the two-phase implicit hydrology model. In particular, the following list details which files have been altered, checked, and/or added.

1. Altered files:

   - astep.df
     Adjusted Newton iteration to more closely resemble the hydrology model; also eliminated working arrays that weren't needed. This was also edited to use the new boundary condition functions.
   - aivdat.df
     Added reservoir initialization steps and parallelization.
   - aprop.df
     The initialization for the Jacobian and residual were checked; there is a line that states:

     the following line is wrong, correct it!,

     but there is no indication as to who wrote this line and what is wrong. The line in question has been checked, and no changes were made.

     There was no instruction in the subroutine awritesol in the if loop that begins with IF (KEYOUT(I,J,K).EQ.1) THEN. A write statement was added here to correspond to the same subroutine, iwritesol, in iprop.df in the black oil model. A "writesol" subroutine is not found in the implicit hydrology model.

- `awell.df`
- `avisual.df`
  Added mortar and parallel components.

2. Checked files:

   - `aisdat.df`
     The value for temperature in `aisdat` was changed. It was previously set to 150, but was changed to be 519.67 degrees Kelvin, or 60 degrees Farenheit.
   - `atran3.df`
     Checked calculations of residual and transport contributions.
   - `afault.df`

3. Added files:

   - `abdary.df`

4. Removed files:

   - `abdry.df`

8.3 Descriptions of Files in AIRV2

All of the files names for version 2 of the air-water model are provided in this appendix, along with a brief description of their contents.

`aarray.df`
This routine creates air-water grid element arrays. It is the same as in the initial version of the air-water model, and contains only the `aarray` subroutine.

`abdary.df`
The subroutines for general boundary conditions are in this file. It is patterned after `hbdary.df` in the implicit hydrology model and contains the following subroutines:

1. `abdintab`
2. `abdprop`
3. `abdpropw`
4. `abdtran`

5. `abdtranw`
6. `abdbal`
7. `acomppreden`

The general boundary properties are evaluated in `abdpropw`. Given user-specified values, the appropriate calculations are done to obtain water pressure and water saturation on the boundary elements. Once this is done, the components of $TRNDAT$ for the boundary elements are evaluated. The transport contributions from the boundary elements for the Jacobian are found in `abdtranw`. Boundary fluxes are computed in `abdbal` and added to the mass balance on the interior. The routine `acomppreden` is used when the user specifies water pressure at a reference depth for the boundary condition. This routine initializes the remaining boundary elements relative to the pressure at the reference depth.

`abwell.df`
This file handles the well calculations and contains the subroutines

1. `abwell`
2. `abwelsums`

It follows the structure of `hwell.df`. Given the well type, the wellbore densities and pressures are calculated in `abwell`. Once this is done, the subroutine branches on well type, and Jacobian contributions and residuals are computed. The mass balances for the wells are found in `awelsums`.

`afault.df`
This file contains the multi-block code for the air-water model and is analogous to `hfault.df`. The subroutines contained in this file are

1. `abcini_ex`
2. `abcini`
3. `abcprop_ex`
4. `abc_prop`
5. `abcflux_ex`
6. `abcflux_comp`
7. `afluxcopy`

8. arep_bal
9. amrestbal

The routine abcini sets the boundary values for multiblock. The physical boundary conditions are set in abc_prop; the parameters that are set are air and water density, and air and water mobility. Velocity in the mortar layer are computed in abc_flux, and balances are initialized in amrestbal.

aiadat.df
This file contains only one subroutine. This routine either sets a dummy variable or reads initialization data if the areadin flag is set.

aisdat.df
This file is based on iisdat.df from the black-oil model. It contains only one routine, aisdat. The routine reads in the scalar data from the input file and sets default values if no value is given for the parameter.

aivdat.df
This file contains the initialization routines for the air-water model. It mimics the file hivdat.df from the implicit hydrology model and contains the subroutines

1. aivdat
2. ainit

The subroutine aivdat initializes the mass balances and calls ainit for the general initialization. The reservoir is required to be at equilibrium initially, and these calculations (as described in §3.1) are made in ainit.

aprop.df
This file follows hprop.df and contains the subroutines

1. aprop
2. aupsol
3. a_initrate
4. a_rate
5. a_zerdnk
6. amaxresid
7. asave

The subroutine `aprop` evaluates the parameters on the interior of the domain and populates the $TRNDAT$ data structure. Improved initial guesses for water pressure and water saturation for the Newton iteration are also determined, and relative permeabilities and porosities are calculated. The Jacobian and residuals are also initialized in this subroutine. The solution is updated in `aupsol`, which is called once the Newton iteration has converged. The variables `pwat_rate` and `swat_rate` are set to 0 in `a_initrate`, and these rates are updated in `a_rate`. `amaxresid` finds the local maximum Newton residuals and is called from `astep`. The values of the primary variables are saved from the previous Newton step in `asave` so that time step cuts can be made if needed. The `dunk(i,j,k,l)` parameter is zeroed in `azerdnk`. This was written to correspond to a similar function in the hydrology model, but it is not needed in the air-water model because `dunk(i,j,k,l)` is only used to store densities. In the implicit hydrology model, this data structure also stores the Newton step for the primary variables upon return from the linear algebra solver.

`aquit.df`
This file contains one subroutine, `aquit`, which is called upon successful termination of the air-water model.

`arest.df`
This file contains two subroutines,

1. `aresto`
2. `aresti`

The `aresto` subroutine outputs data from the model for restart purposes, and `aresti` reads the restart data.

`asprb3.df`
There are three subroutines in this file, which is used for transport evaluation for diagonal tensor permeabilities. The subroutines are

1. `callasprb3`
2. `asprb3x`
3. `asprb3y`
4. `asprb3z`

The subroutine `callasprb3` is used only in conjunction with certain preconditioners. The subroutines `asprb3x`, `asprb3y`, and `asprb3z` are used to evaulate x, y, and z transport contributions, respectively.

`astdout.df`
This file contains the subroutine `astdout`, and is used for air-water model output both to the screen and to the user-specified output file.

`astep.df`
This file contains the `astep` subroutine, which advances the air-water solution in time. The integration is done using backward Euler, and the resulting nonlinear equation is solved using Newton's method. This file is based on the `hstep.df` file, and contains the same termination criterion as given there. There are function calls to the `aprop`, `aupsol`, `awelsums`, `abdtran`, `abdprop`, and transport routines, among others.

`atdata.df`
This file contains only the `atdata` subroutine, which reads in the convergence tolerance parameter for the Newton iteration.

`atran3.df`
The transport subroutines for diagonal permeabilities are contained in this file, which follows `tran3.df` from the implicit hydrology model. These subroutines are

1. `atran3x`
2. `atran3y`
3. `atran3z`,

and they are used to evaluate the air and water equation transport contributions to the Jacobian and residual in the x, y, and z directions, respectively. As described in the documentation, the mobility terms are upwinded, and normalization by the standard air and water densities is incorporated into the mobilities in the $TRNDAT$ data structure.

`avisual.df`
This file is modeled after `hvisual.df`, and contains the subroutines

1. `avis_transl`
2. `avis_tecinit`
3. `avis_tecoutput`

4. avelcomp
5. avis_tecdebug
6. apscomp
7. abl_pscomp
8. acomp_awhead
9. acomp_awheadw

These subroutines are used to output air-water data for TECPLOT visualization. The variables requested by the user are input in avis_transl, and data is output in avis_tecoutput. avis_tecinit is currently a dummy routine. Velocities, or fluxes, across the cells are computed in avelcomp. In apscomp, air pressure and air saturation are found using water pressure and water saturation. The solution for the Buckley-Leverett problem is computed in abl_pscomp, and values for pressure head are found in acomp_awhead. The routines for the analytical solution of the Buckley-Leverett problem have not been fully debugged in the air-water model.

awdata.df
This file contains the subroutine awdata and is purported to input air-water model data. However, there is no evidence that anything is done by the subroutine.

8.4 Buckley-Leverett Input Datafile

The input file for the Buckley-Leverett problem is below and is also listed in the data subdirectory as `airBD.dat`. This example provides a one-dimensional test case with Dirichlet boundary conditions on water pressure and water saturation.

```
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ Original version hydroBD.dat by M. Peszynska, 1/01
$ Modified for use for use in air-water model, L. Jenkins, 10/01
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

TITLE(2)="1D example for Buckley-Leverett case"

DESCRIPTION()=""

AIR_WATER_MODEL

TIMEEND = 101.

$ I/O OPTIONS

OUTLEVEL = 2
PROCOUT
$N_GS_STEP = 5
$LSOL_TOL = 1.E-4
CVTOL = 1.E-6
$ TEMPERATURE = 60.
FACTOR_RATE = 0.

METHOD = 2

BLOCKNAME(1) = "BLOCK1"
DOWN(1 TO 3,1) = 1 0 0
$NX(1) = 1  NY(1) = 100  NZ(1) = 1
$DX(1 TO 1,1) = 10.  DY(,1) = 10.  DZ(1 TO 1,1) = 10.
$NX(1) = 1  NY(1) = 1  NZ(1) = 100
$DX(1 TO 1,1) = 10.  DY(1 TO 1,1) = 10.  DZ(,1) = 10.
NX(1) = 20  NY(1) = 1  NZ(1) = 1
DX(,1) = 50.  DY(1 TO 1,1) = 10.  DZ(1 TO 1,1) = 10.
```

```
XYZ111(,1) = 3*0.

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ boundary conditions: geometry

NBND_REG = 2
NBND_SUBREG = 2

$             nreg nblk  x1 y1 z1      x2 y2 z2
$BND_VOL(,1) = 1     1,   0. 0. 0.      100. 0. 20.      $ y = 0, y-
$BND_VOL(,2) = 2     1,   0. 1000. 0.   1000. 1000. 20.  $ y = 1000, y+

BND_VOL(,1) = 1     1,   0. 0. 0.      0. 100. 20.      $ x = 0, x-
BND_VOL(,2) = 2     1,   1000. 0. 0.   1000. 1000. 20.  $ x = 1000, x+

$BND_VOL(,1) = 1     1,   0. 0. 0.      100. 20. 0.      $ z = 0, z-
$BND_VOL(,2) = 2     1,   0. 0. 1000.   1000. 20. 1000.  $ z = 1000, z+

$ boundary conditions: type and conditions

BOUND_TYPE(1) = 1
ABOUND1(1) Block $ PWAT
  Interpolation Linear
  Extrapolation Constant
  Data  0.  700.
$  Data  0.  550.
EndBlock

ABOUND2(1) Block $ SWAT
  Interpolation Linear
  Extrapolation Constant
  Data  0.  .4
EndBlock

BOUND_TYPE(2) = 1
ABOUND1(2) Block $ PWAT
  Interpolation Linear
  Extrapolation Constant
  Data  0.  300.
EndBlock
```

```
ABOUND2(2) Block $ SWAT
  Interpolation Linear
  Extrapolation Constant
  Data  0.  0.8
$  Data  0.  .2
EndBlock

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ INITIAL CONDITIONS

DINIT = 25.0   PWINIT = 500.  SWINIT = .2

WATST = 62.3430
AIRST = 0.07631
$ROCKCMP = 0.0
$WCOMP = .0000031
WCOMP = 1.E-10
RGAS  = 10.732
MOLWT = 28.97
TEMPERATURE = 60.

$ POROSITY

POROSITY1() = .3

$ PERMEABILITIES

XPERM1() = 500
YPERM1() = 1
ZPERM1() = 1

KASW(1) Block        $ AIR RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Spline2
$  Interpolation Linear
  Extrapolation Constant
  Constraint  0 At .8
  Derivative  0 At .8
  Constraint  1 At  0
  Nodes  .2  .58
```

```
  Data  0.  1. , .1  .67 , .2  .46 , .4  .2 , .6  .055 , .7  .015 , .8  0
EndBlock

KWSW(1) Block        $ WATER RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Spline2
$  Interpolation Linear
  Extrapolation Constant
  Constraint  0 At .15
  Derivative  0 At .15
  Constraint  1 At   1
  Nodes  .55  .7  .75
  Data  .15  0 , .3  .035 , .4  .085 , .6  .28 , .8  .776 , .9 .9, 1  1
EndBlock

PCAW(1) Block        $ WATER-AIR CAPILLARY PRESSURE - ROCK TYPE 1
  Interpolation Spline3
  Extrapolation Same
  Nodes  .25  .4  .7  .9
  Pole   .12
  Data
  .16   .009 , .2   0.00612 , .225 0.00486 , .25  0.00422 ,
  .275 0.00378 , .325 0.0032 ,
  .375 0.00274 , .45  0.00228 , .55  0.00194 ,
  .65  0.00174 , .75  0.00161 , .85  0.00154
  .925 0.00144 , .95  0.00137 , .975 0.00114 , 1.0  .0007
EndBlock

ZFAC(1) Block
  Interpolation Linear
  Extrapolation Same
  Data
  0. 1.0 , 1470. 1.018
EndBlock

PHI(1) Block
  Interpolation Linear
  Extrapolation Same
  Data
  10.   0.2 , 4000.   0.2
EndBlock
```

```
AIRVIS(1) Block
  Interpolation Linear
  Extrapolation Same
  Data
  10.   0.02 , 4000.00   0.02
EndBlock

EndInitial

$ TRANSIENT DATA INPUT BLOCKS

BeginTime    0.
DELTIM = .1 DTIMMUL = 1.  DTIMMAX = 1.  TIMOUT = 1.  DTIMOUT = 100.
VISOUT = 1. DVISOUT = 10.
VIS_SCL = 4
VIS_SCL_NAMES(1) = "PWAT"
VIS_SCL_NAMES(2) = "SWAT"
VIS_SCL_NAMES(3) = "PAIR"
VIS_SCL_NAMES(4) = "SAIR"
VISFLAG = 1
VIS_FNAME = "BUCKLEV"
EndTime
```

8.5 Vauclin Input Datafile

The input file for the Vauclin problem is below and is listed in the data subdirectory as `vauclin.dat`. The description and title are self-explanatory. The line `AIR_WATER_MODEL` tells IPARSv2 to run the air-water code. This keyword is the same for either version of the air-water model; the version that is run depends on the executable that has been established using the `Makefile` and `ipars.siz` file.

Most of the information about IPARSv2 input files can be found in the User's Manual [1]. The variables specific to the air-water model are initialized in `aisdat.df`. In this example, we have specified water compressibility `WCOMP`, water stock tank density `WATST`, air stock tank density `AIRST`, the gas constant `RGAS`, and the molecular weight for air `MOLWT`.

The reservoir is initialized using values of water pressure `PWINIT` and water saturation `SWINIT` at a depth of `11[ft]`. We also specify that the initialization conditions holds at rock type 2 by giving a value for `ROCKINIT`. The default values for `ROCKINIT` is 1.

The curves that must be specified for the air-water model are capillary pressure `PCAW`, water relative permeability `KWSW`, air relative permeability `KASW`, compressibility factor for air `ZFAC`, air viscosity `AIRVIS`, and porosity `PHI`. Capillary pressure is a function of water saturation, as are the relative permeabilities. The compressibility factor and the air viscosity are functions of air pressure, and the porosity is a function of water pressure.

This example file shows how one may specify different material properties for different rock types. It also shows the input data for wells, both water injection and production.

```
TITLE(2)="AIR-WATER VERSION OF VAUCLIN TEST"

DESCRIPTION()=
"LENGTH (FT) : 400"
"WIDTH (FT) : 20"
"THICKNESS (FT) : 20"
"GRID BLOCKS : 10X20X20   4000 GRID ELEMENTS"
"DATE : 10/8/01"

AIR_WATER_MODEL

TIMEEND = 2000.0
```

```
$ I/O OPTIONS

OUTLEVEL = 2  SPLINEOUT  GEOMOUT  PROCOUT
WELLOUTKEY = 3   WELLFILE = "exp.wel"

VIS_TABOUTTYPE=1

WCOMP = 1.E-12
WATST = 62.428
AIRST = 0.07631
RGAS  = 10.732
MOLWT = 28.97
$TEMPERATURE = 60

$ FAULT BLOCK AND MESH DATA

DOWN() = 1 0 0
NX(1) = 10  NY(1) = 20  NZ(1) = 1
DX() = 2.   DY() = 20.  DZ() = 20.
XYZ111(,1)=3*0.

$ ROCK TYPES

ROCK1()=1
ROCK1(,20,)=2   $ DITCH

$ INITIAL CONDITIONS

PWINIT = 503.9451111  SWINIT = .500  DINIT = 11  ROCKINIT = 2   $ DITCH

$ POROSITY

POROSITY1() = .2

$ PERMEABILITIES

XPERM1() = 20  XPERM1(8,,) = 5
YPERM1() = 200
```

```
XPERM1(,20,) = 2000  YPERM1(,20,) = 2000      $ DITCH

KWSW(1) Block        $ WATER RELATIVE PERMEABILITY VS Sw - BED
  Interpolation Spline2
  Extrapolation Constant
  Constraint  1 At 1
  Nodes  .41016  .65609  .91390
  Data
  .0   3.35927E-02 , .05  3.65852E-02 , .1   4.00166E-02 , .15  4.39788E-02
  .2   4.85891E-02 , .25  5.39992E-02 , .3   6.04082E-02 , .35  6.80810E-02
  .4   7.73762E-02 , .45  8.87881E-02 , .5    .103012    , .55   .121048
  .6    .144369    , .65   .175196    , .7    .216961    , .75   .275057
  .8    .357924    , .85   .477919    , .9    .648100    , .95   .858851
  1.   1.
EndBlock

KWSW(2) Block        $ WATER RELATIVE PERMEABILITY VS Sw - DITCH
  Interpolation Spline2
  Extrapolation Constant
  Constraint  1 At 1
  Nodes  .41016  .65609  .91390
  Data
  .0   3.35927E-02 , .05  3.65852E-02 , .1   4.00166E-02 , .15  4.39788E-02
  .2   4.85891E-02 , .25  5.39992E-02 , .3   6.04082E-02 , .35  6.80810E-02
  .4   7.73762E-02 , .45  8.87881E-02 , .5    .103012    , .55   .121048
  .6    .144369    , .65   .175196    , .7    .216961    , .75   .275057
  .8    .357924    , .85   .477919    , .9    .648100    , .95   .858851
  1.   1.
EndBlock

KASW(1) Block        $ AIR RELATIVE PERMEABILITY VS Sw - BED
  Interpolation Spline2
  Extrapolation Constant
  Data  0.  1. , .3  .44444444 , .6  .11111111 , 1. 0.
EndBlock

KASW(2) Block        $ AIR RELATIVE PERMEABILITY VS Sw - DITCH
  Interpolation Spline2
  Extrapolation Constant
  Data  0.  1. , .3  .44444444 , .6  .11111111 , 1. 0.
```

```
EndBlock

PCAW(1) Block        $ AIR-WATER CAPILLARY PRESSURE - BED
  Interpolation Spline3
  Extrapolation Same
  Nodes  .30002 .90735
  Pole   .0
  Data
  .01  2.676916 , .05  1.515075 , .1   1.170937 , .15   .998281
  .2    .885302 , .25   .801695 , .3    .735145 , .35   .679500
  .4    .631256 , .45   .588215 , .5    .548889 , .55   .512192
  .6    .477269 , .65   .443383 , .7    .409822 , .75   .375802
  .8    .340312 , .85   .301798 , .9    .257297 , .95   .198854
  1.    .0
EndBlock

PCAW(2) Block        $ AIR-WATER CAPILLARY PRESSURE - DITCH
  Interpolation Spline3
  Extrapolation Same
  Nodes  .30002 .90735
  Pole   .0
  Data
  .01  .2676916 , .05  .1515075 , .1   .1170937 , .15   .0998281
  .2    .0885302 , .25   .0801695 , .3    .0735145 , .35   .0679500
  .4    .0631256 , .45   .0588215 , .5    .0548889 , .55   .0512192
  .6    .0477269 , .65   .0443383 , .7    .0409822 , .75   .0375802
  .8    .0340312 , .85   .0301798 , .9    .0257297 , .95   .0198854
  1.    .0
EndBlock

ZFAC(1) Block        $ AIR COMPRESSIBILITY VS AIR PRESSURE
  Interpolation Linear
  Extrapolation Constant
  Data
  0. 1., 1470. 1.0
EndBlock

AIRVIS(1) Block      $ AIR VISCOSITY VS AIR PRESSURE
  Interpolation Linear
  Extrapolation Constant
```

```
  Data
  10.   0.02 , 4000.00   0.02
EndBlock

PHI(1) Block         $ POROSITY VS WATER PRESSURE
  Interpolation Linear
  Extrapolation Constant
  Data
  10.   0.3 , 4000.   0.3
EndBlock

NUMWELL=3  TYPICAL=3

WELLNAME(1) = "INJECTION"
KINDWELL(1) = 1
WELLTOP(1 TO 3,1,1) = 0 2*10.
WELLBOTTOM(1 TO 3,1,1) = 2 2*10.
WELLPQ(1) Block
  Interpolation Linear
  Extrapolation Constant
$  Data  0.  499.7  10.  510.         $ 499.7 FROM TYPICAL OUTPUT
  Data  0.  505.00
EndBlock

WELLNAME(2) = "TOP PRODUCTION"
KINDWELL(2) = 3
WELLTOP(1 TO 3,1,2) = 0 390 10
WELLBOTTOM(1 TO 3,1,2) = 2 390 10
WELLPQ(2) Block
  Interpolation Linear
  Extrapolation Constant
$  Data  0.  503.15              $ 503.2 FROM TYPICAL OUTPUT
  Data  0.  503.82
EndBlock

WELLNAME(3) = "BOTTOM PRODUCTION"
KINDWELL(3) = 3
WELLTOP(1 TO 3,1,3) = 18 390 10
WELLBOTTOM(1 TO 3,1,3) = 20 390 10
WELLPQ(3) Block
```

```
  Interpolation Linear
  Extrapolation Constant
$  Data  0.  507.4               $ 507.4 FROM TYPICAL OUTPUT
  Data  0.  507.91
EndBlock

EndInitial

$ TRANSIENT DATA INPUT BLOCKS

BeginTime    0.
DELTIM = .2  DTIMMUL = 1.1  DTIMMAX = .34  TIMOUT = 0  DTIMOUT = 50
CVTOL = 2.E-5  TIMRES = 999999.
VISOUT = 1. DVISOUT = 100.
VIS_SCL = 2
VIS_SCL_NAMES(1) = "PWAT"
VIS_SCL_NAMES(2) = "SWAT"
VIS_FNAME = "VAUCLIN"
EndTime

BeginTime    100.
DTIMMAX = .5  TIMOUT = 365.25   DTIMOUT = 365.25
EndTime

BeginTime    200.
DTIMMAX = .67   CVTOL = 1.E-5
EndTime

BeginTime    365.25
DTIMMAX = 1.
EndTime
```

8.6 Oxbow Input Datafile

The input file for the oxbow problem is below and is listed in the data subdirectory as `air5BD.dat`. This example took advantage of the preconditioners available in the `ygmres/solve` directory with the `GMRES_PREC` option.

This example also demonstrates the use of boundary conditions in the air-water model. The boundary regions are specified the same as in other models. The boundary conditions are input using the `ABOUND` table input. The table information for boundary conditions has the same format as in the other IPARSv2 models.

```
TITLE(2)="OXBOW TEST CASE - MODIFIED FOR AIR-WATER"

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$ This test case was originally developed for the hydrology
$ model and has been modified for the air-water model.
$ L. Jenkins, 5/02
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

DESCRIPTION()=
"GRID ELEMENTS : 19793 ACTIVE (10 x 72 x 38)"
"DIMENSIONS : 13.1 FT x 1759.9 FT x 919.9 FT"
"DATE : 6/10/98"
TITLE(2)="AIR-WATER MODEL TEST 5 - OXBOW"

AIR_WATER_MODEL

TIMEEND=601.

$ I/O OPTIONS

OUTLEVEL = 2   GEOMOUT   PROCOUT
N_GS_STEP = 10
GMRES_PREC = 16
LSOL_TOL =1.E-3

VIS_TABOUTTYPE = 1         $ suggested: output for all tables is produced

$ FAULT BLOCK AND MESH DATA
```

```
DOWN() = .97   0.  -.03
NX(1) = 10  NY(1) = 72  NZ(1) = 38
DX() = 2*2 1 .6 1 1 .5 1 2*2

DY() = 4*40.0 39.2 30.1 23.1 17.7 13.6 10.4 8.0 10.4 13.6 17.7 2*23.1 17.7
13.6 10.4 8.0 10.4 13.6 17.7 23.1 30.1 39.2 2*40.0 39.2 30.1 23.1 17.7 13.6
10.4 8.0 10.4 13.6 17.7 23.1 30.1 39.2 11*40.0 39.2 30.1 23.1 17.7 13.6 10.4
8.0 10.4 13.6 17.7 13.6 10.4 8.0 10.4 13.6 17.7 23.1 30.1 39.2 40.0

DZ() = 40.0 39.2 30.1 23.1 17.7 13.6 10.4 8.0 10.4 8.0 10.4 13.6 17.7 23.1
30.1 39.2 9*40.0 39.2 30.1 23.1 17.7 13.6 10.4 8.0 10.4 8.0 10.4 13.6 17.7
23.1

GRIDMAP(1) Block
             00000000
          00000000000000
        000000000000000000
       0000000000000000000000                          00000000000
     000000000000000000000000                        00000000000000
    0000000000000000000000000                      0000000000000000
   00000000000000001000000000000                  00000000000000000000
  000000000000000000000000000000                 0000000000000000000000
 00000000000000000000000000000000              00000000000000020000000
0000000000000000000000000000000000             00000000000000000000
0000000000000000000000000000000000           0000000000000000000000
0000000000000000000000000000000000          00000000000000000000000
0000000000000000000000000000000000          00000000000000000000000
 00000000000000000000000000000000          000000000000000000000000
  000000000000000000000000000000          0000000000000000000000000
    0000000000000000000000000000          0000000000000000000000000
     00000000000000000000000000          00000000000000000000000000
     00000000000000000000000000           000000000000000000000000000
      0000000000000000000000000           000000000000000000000000000
      0000000000000000000000000           000000000000000000000000000
      00000000000000000000000000        0000000000000000000000000000
     000000000000000000000000000000   00000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000000
    0000000000000000000000000000000000000000000000000000000000000000000000
```

```
   0000000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000000
 00000000000000000000000000000000000000000000000000000000000000000000000
 00000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000300000000000000000000000000000000000000000000000050000000000
0000000000000000000000000000000000000000000000000000000000000000000000
 00000000000000000000000000000040000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
    000000000000000000000000000000000000000000000000000000000000000000
       0000000000000000000000000000000000000000000000000000
            00000000000000000000000000000000000000
EndBlock

KEYOUT1(1 TO 5,,1)=0    KEYOUT1(1 TO 4,,2)=0    KEYOUT1(1 TO 3,,3)=0
KEYOUT1(1 TO 2,,4)=0    KEYOUT1(1,,5)=0


NBND_SUBREG = 8
NBND_REG = 4

BND_VOL(,1) = 1 1 , 1000. 235. 0, 1015. 750. 196.
BND_VOL(,2) = 1 1 , 1500. 1000. 250, 1015. 1250. 420.

BND_VOL(,3) = 2 1 , 1500. 0. 800., 1015. 300. 1000.
BND_VOL(,4) = 2 1 , 1000. 1250. 800., 1015. 1800. 1000.

BND_VOL(,5) = 3 1 , 1000. 1000. 0., 1015. 2000. 200.
BND_VOL(,6) = 3 1 , 1500. 1250. 250., 1015. 1800. 420.

BND_VOL(,7) = 4 1 , 1000. 0. 200., 1015. 500. 300.
BND_VOL(,8) = 4 1 , 1500. 0. 750., 1015. 300. 300.

BOUND_TYPE(1) = 1
ABOUND1(1) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  700.
EndBlock
```

```
ABOUND2(1) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  0.7
EndBlock

BOUND_TYPE(2) = 1
ABOUND1(2) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  700.
EndBlock
ABOUND2(2) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  0.6
EndBlock

BOUND_TYPE(3) = 1
ABOUND1(3) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  700.
EndBlock
ABOUND2(3) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  0.6
EndBlock

$ FLUID PROPERTIES

WCOMP=0.0
WATST = 62.34
AIRST = 0.07631
RGAS  = 10.732
MOLWT = 28.97
WATVIS = .3

$ INITIAL CONDITIONS
```

```
XYZ111() = 1000.  0.  0.  DINIT = 1000.  PWINIT = 600.  SWINIT = .35

$ POROSITY

POROSITY1() = .22 POROSITY1(4,,) = .08 POROSITY1(7,,) = .09

$ PERMEABILITIES

XPERM1() = 25   XPERM1(4,,) = 5   XPERM1(7,,) = 3
YPERM1() = 200  YPERM1(4,,) = 30  YPERM1(7,,) = 40

KASW(1) Block        $ OIL RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Spline2
  Extrapolation Constant
  Constraint  0 At .8
  Derivative  0 At .8
  Constraint  1 At  0
  Nodes  .2  .58
  Data  0.  1. , .1  .67 , .2  .46 , .4  .2 , .6  .055 , .7  .015 , .8  0
EndBlock

KWSW(1) Block        $ WATER RELATIVE PERMEABILITY VS Sw - ROCK TYPE 1
  Interpolation Spline2
  Extrapolation Constant
  Constraint  0 At .15
  Derivative  0 At .15
  Constraint  1 At   1
  Nodes  .55  .7  .75
  Data  .15  0 , .3  .035 , .4  .085 , .6  .28 , .8  .776 , .9 .9, 1  1
EndBlock

PCAW(1) Block        $ WATER-OIL CAPILLARY PRESSURE - ROCK TYPE 1
  Interpolation Spline3
  Extrapolation Same
  Nodes  .25  .4  .7  .9
  Pole   .12
  Data
  .16   9. , .2   6.12 , .225 4.86 , .25  4.22 , .275 3.78 , .325 3.2
  .375 2.74 , .45  2.28 , .55  1.94 , .65  1.74 , .75  1.61 , .85  1.54
```

```
  .925 1.44 , .95  1.37 , .975 1.14 , 1.0  .7
EndBlock

ZFAC(1) Block        $ check - AIR COMPRESSIBILITY
  Interpolation Linear
  Extrapolation Same
  Data
  0. 1. , 400. 1.
EndBlock

AIRVIS(1) Block      $ check - AIR VISCOSITY
  Interpolation Linear
  Extrapolation Constant
  Data
  10.   0.02 , 4000.00   0.02
EndBlock

PHI(1) Block         $ check - POROSITY VS ?? PRESSURE ??
  Interpolation Linear
  Extrapolation Constant
  Data
  10.   0.3 , 4000.   0.3
EndBlock

EndInitial

$ TRANSIENT DATA INPUT BLOCKS

BeginTime    0.
DELTIM = 1.  DTIMMUL = 1.2  DTIMMAX = 1.5  TIMOUT = 30.
CVTOL = .3E-6  TIMRES = 30.
VISOUT = 1. DVISOUT = 10.
VIS_SCL = 2
VIS_SCL_NAMES(1) = "PWAT"
VIS_SCL_NAMES(2) = "SWAT"
VIS_FNAME = "HSHOE"
EndTime

BeginTime 20.
BOUND_TYPE(1) = 0
```

```
BOUND_TYPE(3) = 0
EndTime

BeginTime 200.
BOUND_TYPE(4) = 1
ABOUND1(4) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  500.
EndBlock
ABOUND2(4) Block
  Interpolation Linear
  Extrapolation Constant
  Data  0.  0.25
EndBlock
EndTime
```

8.7 Oxbow Output Datafile

The output for a 50 day simulation of the oxbow problem for a single node is given below. Note that the mass balances are maintained throughout the simulation.

```
 0 NewtRes26802.     33108.
# of GMRES itns =            14  res.err=    7.999146
 1 NewtRes262.40     8692.1
# of GMRES itns =            15  res.err=   8.9775123E-02
 2 NewtRes53.847     2806.8
# of GMRES itns =            16  res.err=   1.7494552E-02
 3 NewtRes29.662     748.21
# of GMRES itns =            18  res.err=   7.3507749E-03
 4 NewtRes12.817     169.43
# of GMRES itns =            18  res.err=   3.1941168E-03
 5 NewtRes2.3298     30.161
# of GMRES itns =            19  res.err=   2.4069681E-04
 6 NewtRes.12645     1.2769
# of GMRES itns =            18  res.err=   2.5556827E-05
 7 NewtRes.17735E-02 .97135E-02
# of GMRES itns =            18  res.err=   2.0137122E-07
STEP   1 TIME    1.20 WAT 1.0000000 AIR 1.0000000 NEWT  8 LINR 17.00
Final NewtRes: .15407E-06 .27126E-06

 0 NewtRes147.62     412.36
# of GMRES itns =            20  res.err=   6.9201112E-02
 1 NewtRes31.507     436.83
# of GMRES itns =            18  res.err=   1.3142820E-02
 2 NewtRes5.1775     226.63
# of GMRES itns =            17  res.err=   1.6179981E-03
 3 NewtRes1.1019     41.280
# of GMRES itns =            15  res.err=   3.1186911E-04
 4 NewtRes.11683     3.2787
# of GMRES itns =            15  res.err=   1.1699570E-05
 5 NewtRes.71025E-03 .16892E-01
# of GMRES itns =            16  res.err=   9.1500951E-08
 6 NewtRes.37841E-06 .29772E-05
# of GMRES itns =            11  res.err=   8.5738048E-09
STEP   2 TIME    2.64 WAT 1.0000000 AIR 1.0000000 NEWT  7 LINR 16.00
Final NewtRes: .10996E-09 .70953E-08
```

```
 0 NewtRes49.642     140.59
# of GMRES itns =            17  res.err=   4.2859267E-02
 1 NewtRes25.754     132.60
# of GMRES itns =            17  res.err=   5.6444202E-03
 2 NewtRes3.8924     29.387
# of GMRES itns =            14  res.err=   1.1003269E-03
 3 NewtRes.19659     3.0919
# of GMRES itns =            15  res.err=   1.9956371E-05
 4 NewtRes.20771E-02 .51608E-01
# of GMRES itns =            15  res.err=   1.7492172E-07
 5 NewtRes.15708E-05 .10754E-04
# of GMRES itns =            11  res.err=   6.0466934E-09
STEP   3 TIME    4.14 WAT 1.0000000 AIR 1.0000000 NEWT  6 LINR 14.83
Final NewtRes: .32390E-09 .11187E-07

 0 NewtRes40.267     114.04
# of GMRES itns =            16  res.err=   3.2969180E-02
 1 NewtRes18.805     53.789
# of GMRES itns =            14  res.err=   4.1285595E-03
 2 NewtRes.29415     3.7713
# of GMRES itns =            14  res.err=   8.7659697E-05
 3 NewtRes.38531E-02 .65089E-01
# of GMRES itns =            16  res.err=   2.7463062E-07
 4 NewtRes.49534E-05 .47984E-04
# of GMRES itns =            13  res.err=   4.2749577E-09
STEP   4 TIME    5.64 WAT 1.0000000 AIR 1.0000000 NEWT  5 LINR 14.60
Final NewtRes: .99926E-09 .81955E-08

 0 NewtRes34.359     97.312
# of GMRES itns =            15  res.err=   1.2469083E-02
 1 NewtRes11.957     51.062
# of GMRES itns =            13  res.err=   2.8089951E-03
 2 NewtRes.42453     3.2753
# of GMRES itns =            13  res.err=   9.3418777E-05
 3 NewtRes.17257E-01 .45570E-01
# of GMRES itns =            13  res.err=   2.0191558E-06
 4 NewtRes.95562E-05 .46350E-04
# of GMRES itns =            13  res.err=   5.5053331E-09
STEP   5 TIME    7.14 WAT 1.0000000 AIR 1.0000000 NEWT  5 LINR 13.40
```

```
Final NewtRes: .22179E-08 .76700E-08

 0 NewtRes27.317     77.364
# of GMRES itns =            15  res.err=   1.8417826E-02
 1 NewtRes7.5292     33.471
# of GMRES itns =            12  res.err=   2.0808335E-03
 2 NewtRes.15780     1.0523
# of GMRES itns =            12  res.err=   2.7482773E-05
 3 NewtRes.28215E-02 .13748E-01
# of GMRES itns =            11  res.err=   2.2830353E-07
 4 NewtRes.16819E-05 .99540E-05
# of GMRES itns =            10  res.err=   6.9059327E-09
STEP   6 TIME    8.64 WAT 1.0000000 AIR 1.0000000 NEWT  5 LINR 12.00
Final NewtRes: .47963E-09 .85540E-08

 0 NewtRes26.461     74.942
# of GMRES itns =            15  res.err=   8.7982388E-03
 1 NewtRes8.5285     17.380
# of GMRES itns =            12  res.err=   1.2489499E-03
 2 NewtRes.44759E-01 .56971
# of GMRES itns =            13  res.err=   5.5807263E-06
 3 NewtRes.33163E-03 .51461E-02
# of GMRES itns =            13  res.err=   3.7269938E-08
STEP   7 TIME   10.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 13.25
Final NewtRes: .26575E-06 .25974E-06

 0 NewtRes23.687     67.083
# of GMRES itns =            15  res.err=   6.1588585E-03
 1 NewtRes6.0351     16.351
# of GMRES itns =            12  res.err=   9.9523668E-04
 2 NewtRes.53202E-01 .34954
# of GMRES itns =            13  res.err=   4.5201318E-06
 3 NewtRes.18627E-03 .16201E-01
# of GMRES itns =            14  res.err=   1.8568478E-08
STEP   8 TIME   11.64 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 13.50
Final NewtRes: .38511E-06 .98747E-06

 0 NewtRes20.519     58.115
# of GMRES itns =            15  res.err=   5.2779205E-03
 1 NewtRes5.8791     8.5488
```

```
# of GMRES itns =            11  res.err=   7.1905693E-04
 2 NewtRes.26920E-01 .32730
# of GMRES itns =            12  res.err=   4.1733842E-06
 3 NewtRes.60338E-04 .62777E-02
# of GMRES itns =            14  res.err=   5.5411356E-09
STEP   9 TIME   13.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 13.00
Final NewtRes: .18708E-06 .52680E-06

 0 NewtRes18.718     53.013
# of GMRES itns =            15  res.err=   6.9358102E-03
 1 NewtRes3.7554     5.8557
# of GMRES itns =            10  res.err=   6.3320529E-04
 2 NewtRes.12686E-01 .63833E-01
# of GMRES itns =            13  res.err=   1.2277578E-06
 3 NewtRes.17343E-04 .26230E-04
# of GMRES itns =            12  res.err=   5.7261267E-09
STEP  10 TIME   14.64 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.50
Final NewtRes: .19037E-08 .86243E-08

 0 NewtRes17.875     50.626
# of GMRES itns =            15  res.err=   3.8641195E-03
 1 NewtRes3.8176     5.2828
# of GMRES itns =            11  res.err=   4.2167868E-04
 2 NewtRes.11229     .23669
# of GMRES itns =            10  res.err=   1.2915082E-05
 3 NewtRes.60600E-04 .40659E-03
# of GMRES itns =            12  res.err=   7.9470244E-09
STEP  11 TIME   16.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.00
Final NewtRes: .14706E-07 .40657E-07

 0 NewtRes15.587     44.147
# of GMRES itns =            16  res.err=   1.5634818E-03
 1 NewtRes4.8391     5.0677
# of GMRES itns =            11  res.err=   4.2523505E-04
 2 NewtRes.14978E-01 .68333E-01
# of GMRES itns =            12  res.err=   1.3138455E-06
 3 NewtRes.28267E-05 .13637E-04
# of GMRES itns =            12  res.err=   5.7378839E-09
STEP  12 TIME   17.64 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.75
Final NewtRes: .39818E-09 .82893E-08
```

```
 0 NewtRes13.356     37.828
# of GMRES itns =            15  res.err=   1.8146834E-03
 1 NewtRes3.1300     3.7639
# of GMRES itns =            11  res.err=   3.1937638E-04
 2 NewtRes.63791E-01 .63820E-01
# of GMRES itns =            11  res.err=   8.5471247E-06
 3 NewtRes.41459E-04 .22054E-03
# of GMRES itns =            13  res.err=   3.2352778E-09
STEP  13 TIME   19.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.50
Final NewtRes: .11324E-08 .88058E-08

 0 NewtRes11.443     32.412
# of GMRES itns =            16  res.err=   1.6073978E-03
 1 NewtRes2.3848     2.7573
# of GMRES itns =             9  res.err=   2.1154706E-04
 2 NewtRes.13377E-01 .38561E-01
# of GMRES itns =            12  res.err=   1.8208065E-06
 3 NewtRes.25075E-05 .19943E-04
# of GMRES itns =            10  res.err=   7.7772819E-09
STEP  14 TIME   20.64 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 11.75
Final NewtRes: .12976E-08 .73111E-08

 0 NewtRes10.175     28.819
# of GMRES itns =            16  res.err=   1.5003092E-03
 1 NewtRes4.7268     2.8644
# of GMRES itns =             9  res.err=   7.1764929E-04
 2 NewtRes.31879E-01 .15099
# of GMRES itns =            11  res.err=   4.0085051E-06
 3 NewtRes.12301E-03 .13132E-02
# of GMRES itns =            14  res.err=   6.8466384E-09
STEP  15 TIME   22.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.50
Final NewtRes: .77992E-07 .14345E-06

 0 NewtRes9.3731     26.548
# of GMRES itns =            15  res.err=   2.9935713E-03
 1 NewtRes3.0992     2.5592
# of GMRES itns =            10  res.err=   5.6310865E-04
 2 NewtRes.53190E-01 .11506
# of GMRES itns =            11  res.err=   7.1646523E-06
```

```
 3 NewtRes.34406E-04 .11318E-03
# of GMRES itns =            13  res.err=   8.0669302E-09
STEP  16 TIME   23.64 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.25
Final NewtRes: .12666E-07 .35970E-07

 0 NewtRes8.7347     24.740
# of GMRES itns =            16  res.err=   1.2013668E-03
 1 NewtRes2.1862     2.1976
# of GMRES itns =            10  res.err=   3.6372602E-04
 2 NewtRes.54233E-02 .49018E-01
# of GMRES itns =            13  res.err=   5.0195882E-07
 3 NewtRes.92774E-06 .56877E-05
# of GMRES itns =            12  res.err=   5.6363887E-09
STEP  17 TIME   25.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.75
Final NewtRes: .17848E-09 .88072E-08

 0 NewtRes7.8024     22.099
# of GMRES itns =            16  res.err=   8.1843470E-04
 1 NewtRes1.8814     1.5374
# of GMRES itns =            10  res.err=   2.9298404E-04
 2 NewtRes.90284E-02 .17060
# of GMRES itns =            12  res.err=   1.0625315E-06
 3 NewtRes.11865E-04 .14718E-03
# of GMRES itns =            12  res.err=   3.0095033E-09
STEP  18 TIME   26.64 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.50
Final NewtRes: .41533E-08 .11582E-07

 0 NewtRes6.6899     18.949
# of GMRES itns =            15  res.err=   1.8957384E-03
 1 NewtRes2.0419     1.3035
# of GMRES itns =            11  res.err=   2.1872562E-04
 2 NewtRes.41281E-01 .29714E-01
# of GMRES itns =            10  res.err=   3.5102714E-06
 3 NewtRes.32459E-05 .60369E-05
# of GMRES itns =            11  res.err=   5.3496909E-09
STEP  19 TIME   28.14 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 11.75
Final NewtRes: .18106E-09 .79984E-08

 0 NewtRes3.5235     9.9801
# of GMRES itns =            15  res.err=   5.4806593E-04
```

```
 1 NewtRes.15068     .44012
# of GMRES itns =            11  res.err=   2.7368642E-05
 2 NewtRes.60133E-03 .22541E-02
# of GMRES itns =            12  res.err=   5.3361386E-08
STEP  20 TIME   29.09 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 12.67
Final NewtRes: .44032E-07 .10252E-06

 0 NewtRes3.0859     8.7405
# of GMRES itns =            14  res.err=   4.7737648E-04
 1 NewtRes.16380     .23428
# of GMRES itns =            12  res.err=   9.5211881E-06
 2 NewtRes.94709E-04 .73635E-02
# of GMRES itns =            13  res.err=   9.0101819E-09
STEP  21 TIME   30.00 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 13.00
Final NewtRes: .97921E-07 .19033E-06

 0 NewtRes3.2822     9.2962
# of GMRES itns =            15  res.err=   4.3991167E-04
 1 NewtRes.20008     .29469
# of GMRES itns =            12  res.err=   1.8824076E-05
 2 NewtRes.36110E-04 .91121E-02
# of GMRES itns =            14  res.err=   5.4728075E-09
STEP  22 TIME   31.00 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 13.67
Final NewtRes: .21292E-06 .60977E-06

 0 NewtRes3.3630     9.5253
# of GMRES itns =            15  res.err=   5.9022865E-04
 1 NewtRes.48545     .61826
# of GMRES itns =            12  res.err=   3.1043997E-05
 2 NewtRes.10077E-02 .14207E-01
# of GMRES itns =            13  res.err=   1.3489621E-07
 3 NewtRes.65054E-06 .58842E-05
# of GMRES itns =             9  res.err=   6.4044339E-09
STEP  23 TIME   32.11 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 12.25
Final NewtRes: .36303E-09 .57050E-08

 0 NewtRes3.3242     9.4156
# of GMRES itns =            15  res.err=   5.6715508E-04
 1 NewtRes.59259     .45189
# of GMRES itns =            11  res.err=   6.4725042E-05
```

```
 2 NewtRes.30721E-02 .63853E-02
# of GMRES itns =            11  res.err=   3.3680436E-07
STEP  24 TIME   33.32 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 12.33
Final NewtRes: .11383E-06 .21626E-06

 0 NewtRes3.1750     8.9930
# of GMRES itns =            16  res.err=   5.8090268E-04
 1 NewtRes.22767     .56403
# of GMRES itns =            13  res.err=   2.4830450E-05
 2 NewtRes.21586E-03 .10424E-01
# of GMRES itns =            15  res.err=   1.7632054E-08
STEP  25 TIME   34.65 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 14.67
Final NewtRes: .11213E-06 .26189E-06

 0 NewtRes3.0555     8.6529
# of GMRES itns =            16  res.err=   1.4107567E-03
 1 NewtRes.41242     .73669
# of GMRES itns =            13  res.err=   3.4042190E-05
 2 NewtRes.35870E-03 .85134E-02
# of GMRES itns =            14  res.err=   4.9235670E-08
STEP  26 TIME   36.12 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 14.33
Final NewtRes: .28835E-06 .35683E-06

 0 NewtRes3.3748     9.5573
# of GMRES itns =            17  res.err=   1.4899180E-03
 1 NewtRes.35508     1.0011
# of GMRES itns =            14  res.err=   3.0085763E-05
 2 NewtRes.26239E-03 .10259E-01
# of GMRES itns =            15  res.err=   2.7027177E-08
STEP  27 TIME   37.74 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 15.33
Final NewtRes: .13785E-06 .20780E-06

 0 NewtRes3.6292     10.278
# of GMRES itns =            17  res.err=   6.8399380E-04
 1 NewtRes.26601     1.4087
# of GMRES itns =            14  res.err=   5.1067076E-05
 2 NewtRes.45871E-03 .11215E-01
# of GMRES itns =            16  res.err=   4.1396238E-08
STEP  28 TIME   39.51 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 15.67
Final NewtRes: .31115E-06 .88174E-06
```

```
 0 NewtRes3.8374     10.867
# of GMRES itns =            17  res.err=   7.2583632E-04
 1 NewtRes.35026     1.4203
# of GMRES itns =            13  res.err=   8.5539446E-05
 2 NewtRes.20846E-03 .14456E-01
# of GMRES itns =            16  res.err=   4.5901693E-08
STEP  29 TIME   41.46 WAT 1.0000000 AIR 1.0000000 NEWT  3 LINR 15.33
Final NewtRes: .34859E-06 .75492E-06

 0 NewtRes4.4447     12.587
# of GMRES itns =            18  res.err=   5.8144040E-04
 1 NewtRes.59123     1.7429
# of GMRES itns =            14  res.err=   9.8873243E-05
 2 NewtRes.28801E-02 .69428E-01
# of GMRES itns =            15  res.err=   4.0717299E-07
 3 NewtRes.85721E-06 .10185E-03
# of GMRES itns =            14  res.err=   4.0557935E-09
STEP  30 TIME   43.61 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 15.25
Final NewtRes: .77039E-08 .21813E-07

 0 NewtRes5.1461     14.574
# of GMRES itns =            18  res.err=   1.5374696E-03
 1 NewtRes.78937     2.8751
# of GMRES itns =            14  res.err=   1.2094323E-04
 2 NewtRes.11110E-01 .19046
# of GMRES itns =            15  res.err=   1.1204942E-06
 3 NewtRes.82079E-04 .10265E-02
# of GMRES itns =            15  res.err=   5.5933733E-09
STEP  31 TIME   45.98 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 15.50
Final NewtRes: .52180E-07 .12432E-06

 0 NewtRes4.6171     13.076
# of GMRES itns =            18  res.err=   6.8565959E-04
 1 NewtRes.56419     16.475
# of GMRES itns =            14  res.err=   7.8428748E-05
 2 NewtRes.24426E-02 .60273
# of GMRES itns =            16  res.err=   2.6086121E-07
 3 NewtRes.42820E-05 .88415E-03
# of GMRES itns =            15  res.err=   5.7960574E-09
```

```
STEP  32 TIME   48.03 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 15.75
Final NewtRes: .73547E-08 .19265E-07

 0 NewtRes4.3752     12.391
# of GMRES itns =            18  res.err=   6.9078378E-04
 1 NewtRes.51198     6.7958
# of GMRES itns =            14  res.err=   6.4515138E-05
 2 NewtRes.17458E-02 .54508
# of GMRES itns =            16  res.err=   2.9447457E-07
 3 NewtRes.88627E-04 .40267E-02
# of GMRES itns =            15  res.err=   7.5102795E-09
STEP  33 TIME   50.00 WAT 1.0000000 AIR 1.0000000 NEWT  4 LINR 15.75
Final NewtRes: .13959E-06 .18654E-06

Air-Water model successful termination
```

References

[1] IPARS User's Manual. Technical report, Center for Subsurface Modeling, Texas Institute for Computational and Applied Mathematics, University of Texas at Austin, 1998–2000.

[2] T. Arbogast, S. Bryant, J. Eaton, Q. Lu, M. Peszynska, M. F. Wheeler, and I. Yotov. A parallel multiblock/multidomain approach for reservoir simulation. under revision for Society of Petroleum Engineers Journal.

[3] F. J. Eaton. *A Multigrid Preconditioner for Two-Phase Flow in Porous Media*. PhD thesis, University of Texas at Austin, Austin, TX, 2001.

[4] C. T. Kelley. *Iterative methods for linear and nonlinear equations*. SIAM, Philadelphia, 1995.

[5] W. Lee, M. Noh, and M. F. Wheeler. Air-water flow simulation in unsaturated porous media. In L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, editors, *Computational Methods in Water Resources*, pages 93–100. A. A. Balkema, 2000.

[6] S. Minkoff, C. M. Stone, J. G. Arguello, S. Bryant, J. Eaton, M. Peszynska, and M. F. Wheeler. Staggered in time coupling of reservoir flow simulation and geomechanical deformation: Step 1 - one-way coupling. In *1999 SPE Symposium on Reservoir Simulation*, Houston, Texas, 1999. SPE 51920.

[7] Myeong Hwan Noh. Fully coupled air-water flow model in unsaturated porous media. Master's thesis, University of Texas at Austin, December 1999.

[8] M. Parashar, J. A. Wheeler, G. Pope, K. Wang, and P. Wang. A new generation EOS compositional reservoir simulator. part II: Framework and multiprocessing. In *Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas*, pages 31–38. Society of Petroleum Engineers, June 1997.

[9] D. W. Peaceman. Interpretation of well-block pressure in numerical reservior simulation with non-square grid blocks and anisotropic permeability. *Tran. AIME*, 275:10–22, 1983.

[10] M. Peszynska, Q. Lu, and M. F. Wheeler. Coupling different numerical algorithms for two phase fluid flow. In J. R. Whiteman, editor, *MAFE-*

LAP Proceedings of Mathematics of Finite Elements and Applications, pages 205–214, Uxbridge, U.K., 1999. Brunel University.

[11] M. Peszynska, Q. Lu, and M. F. Wheeler. Multiphysics coupling of codes. In L. R. Bentley, J. F. Sykes, C. A. Brebbia, W. G. Gray, and G. F. Pinder, editors, *Computational Methods in Water Resources*, pages 175–182. A. A. Balkema, 2000.

[12] M. Peszynska, M. F. Wheeler, and I. Yotov. Mortar upscaling for multiphase flow in porous media. Computational Geosciences, to appear.

[13] Małgorzata Peszyńska, Eleanor Jenkins, and Mary F. Wheeler. Boundary conditions for fully implicit two-phase flow model. *Contemporary Mathematics*, 2002.

[14] T. F. Russell and M. F. Wheeler. Finite element and finite difference methods for continuous flows in porous media. In R. E. Ewing, editor, *The Mathematics of Reservoir Simulation*, pages 35–106. SIAM, Philadelphia, 1983.

[15] M. Vauclin, D. Khanji, and G. Vachaud. Experimental and numerical study of a transient, two-dimensional unsaturated-saturated watwer table recharge problem. *Water Resources Research*, 15(5):1089–1101, 1979.

[16] P. Wang, I. Yotov, M. F. Wheeler, T. Arbogast, C. N. Dawson, M. Parashar, and K. Sepehrnoori. A new generation EOS compositional reservoir simulator. part I: Formulation and discretization. In *Fourteenth SPE Symposium on Reservoir Simulation, Dallas, Texas*, pages 55–64. Society of Petroleum Engineers, June 1997.