

# Adaptive numerical simulation of the remediation of soil contamination by in-situ gas venting

S.V. Pennington\*and P.K. Jimack  
Computational PDEs Unit, School of Computer Studies,  
University of Leeds, Leeds, LS2 9JT, UK.

K. McFarlane  
Shell Research Limited, Thornton Research Centre,  
PO Box 1, Chester, CH1 3SH, UK.

## Abstract

This paper describes a mathematical model and adaptive numerical simulation of the time-dependent multiphase, multicomponent flow which occurs when a gas venting process is used to remove a volatile contaminant from a porous medium. The numerical simulation is adaptive in both space and time and involves the use of a finite element spatial discretization and the SPRINT2D software ([6]) for time integration. Results are presented which demonstrate the high quality of the simulation, both in terms of the length scales of the features that are resolved and the efficiency of the solutions relative to those obtained on fixed grids.

## 1 Introduction

An important environmental problem faced in a number of industrial sectors today is that of how to deal with the contamination of soil and groundwater due to spills or seepage of unwanted substances, such as chemical waste or hydrocarbons. Frequently these contaminants are volatile and will evaporate freely given a sufficient supply of unsaturated air. In such situations a popular clean-up technique is to employ a venting process which extracts the contaminant-rich gas phase by means of vacuum aspiration of the ground in the vicinity of the spill. A number of authors have considered such a technique both experimentally and numerically, e.g. [3, 8, 14, 16, 20, 21, 25, 28].

A critical factor in the successful numerical simulation of venting processes is the accuracy and efficiency with which the numerical methods approximate the solutions of the underlying differential equations. For time-dependent problems in two and three dimensions the computational overheads associated with using uniform grids in space and time are enormous and so, in this paper, we propose an adaptive numerical scheme for such simulations.

The idea of mesh adaptivity is not a new one, with research into adaptive algorithms going back a number of years (see [1, 22] for example). Indeed, such algorithms have already been applied to certain groundwater transport problems with significant success (e.g. [18]). It is the purpose of this paper to demonstrate the significant advantages of using adaptivity in both space and time for the numerical simulation of gas venting in two space dimensions and to present some of the details behind our particular choice of adaptive algorithm.

It is not difficult to see why the use of mesh adaptivity is likely to be highly appropriate for this class of problem. The basic mechanism behind the venting process is to remove air from the

---

\*Current address: Manchester Computing, University of Manchester, Manchester, M13 9PL, UK.

contaminated ground by sucking it out at one or more wells. This air is replaced by fresh air entering the contaminated region either through its upper surface or from neighbouring, uncontaminated, regions. This in turn causes a flow of clean air through the contaminated ground which enhances the evaporation of the volatile contaminant. Once evaporated this contaminant is then removed at one of the wells where dirty air is extracted. Initially, a uniform spatial mesh will give a perfectly adequate representation of the concentration of the contaminant throughout the domain since it will be assumed that this concentration is uniform. Once venting is underway however, and clean air starts to be drawn in, a sharp transition layer will develop between this clean air and the contaminated region. Mesh refinement is ideal for accurately and efficiently capturing sharp fronts such as this since the mesh need only be fine in these transitional layers. Moreover, as the simulation proceeds and the layers evolve, the mesh can evolve too so that it always allows an accurate representation of the solution at the minimum computational cost.

As well as adaptivity of the spatial mesh, further significant efficiency and reliability gains may be achieved through the use of adaptivity in time. For example, as the rate of extraction at the vent is altered the dominant time-scale for the problem will also vary. In the extreme case where the vent is switched off completely then only diffusion and gravity will be left to drive the flow and so it is essential that very large time steps are permitted.

In the following section a quantitative model is presented for a quite general axisymmetric venting process. This model contains all of the features present in the qualitative description presented above but in common with most work in this area it makes use of a number of simplifying assumptions so as to prevent the complexity becoming excessive. Section 3 then presents a description of the adaptive finite element solution scheme that has been employed, including a detailed explanation of the spatial and temporal adaptivity algorithms used. Some typical computational results are presented in Section 4, where the advantages of mesh adaptivity are clearly demonstrated. The paper then finishes with some concluding remarks.

## 2 The model problem

The model problem considered here involves a multiphase porous media flow model, described in detail below. Corapcioglu and Panday [9] present a comprehensive overview of this type of model, along with numerical solution methods. Key publications relevant to this work include [10, 2, 13, 14, 15, 20, 21, 25].

### 2.1 Governing equations

In the model problem considered here it is assumed that a hydrocarbon liquid has been spilt on “dry” (not water saturated) ground, resulting in a volume distribution of this oleic liquid at or marginally above residual saturation throughout the vadose zone beneath the spill. At this stage we ignore variations in the saturation due to inhomogeneities or anisotropies in the porous media, as well as any accumulation of oil at the water table below the vadose zone. The residual saturation of oil is assumed to be immobile, and will be subject only to evaporative loss to other phases. Liquid water is present at or below residual saturation in the vadose zone and is therefore also immobile. In addition the water is assumed to be non-volatile.

Whilst in reality the spilt liquid will probably contain a number of different hydrocarbons, we will consider here a simplified model in which the liquid is a mixture of one *light* (volatile) and one *heavy* (non-volatile) component. This simplification will address most of the simulation issues whilst maintaining a manageable size for the resulting numerical system. Thus there are four components in this problem – light oil ( $l$ ), heavy oil ( $h$ ), water ( $w$ ) and atmospheric air ( $a$ ) – which can partition between the four phases present: a gaseous phase ( $g$ ) comprising air and light oil vapour; an oleic

phase ( $o$ ) comprising both light and heavy oils; an aqueous phase ( $w$ ) which contains dissolved light oil (the heavy oil being insoluble); and a sorbed phase ( $s$ ) consisting of both light and heavy oils sorbed to the soil surface. The sorbed phase is by definition immobile.

The remaining physical/chemical assumptions are that (i) the air and light oil vapour are ideal gases of fixed and constant molecular weight, (ii) the system is isothermal, (iii) the vadose zone is incompressible, (iv) Darcy's law governs the flow of air and light oil vapour, and (v) initially we have conditions of thermodynamic equilibrium between the phases and hydrostatic equilibrium in the gas phase. It is clear that some of these assumptions are more easily justified than others and we note that the final assumption, of equilibrium between the different phases, has begun to be questioned in a number of situations (see [21] for example).

A standard control volume analysis results in the following equations of motion for the light oil, air, water and heavy oil components respectively:

$$\frac{\partial}{\partial t}(\varepsilon\alpha_g\rho_g X_g^l + \varepsilon\alpha_w\rho_w X_w^l + \varepsilon\alpha_o\rho_o X_o^l + c_s X_s^l) + \nabla \cdot (\rho_g X_g^l \underline{q}) = \nabla \cdot (\varepsilon\alpha_g\rho_g \mathbf{D}_g^l \nabla X_g^l + \varepsilon\alpha_w\rho_w \mathbf{D}_w^l \nabla X_w^l + \varepsilon\alpha_o\rho_o \mathbf{D}_o^l \nabla X_o^l), \quad (1)$$

$$\frac{\partial}{\partial t}(\varepsilon\alpha_g\rho_g X_g^a) + \nabla \cdot (\rho_g X_g^a \underline{q}) = \nabla \cdot (\varepsilon\alpha_g\rho_g \mathbf{D}_g^a \nabla X_g^a), \quad (2)$$

$$\frac{\partial}{\partial t}(\varepsilon\alpha_w\rho_w X_w^w) = \nabla \cdot (\varepsilon\alpha_w\rho_w \mathbf{D}_w^w \nabla X_w^w), \quad (3)$$

$$\frac{\partial}{\partial t}(\varepsilon\alpha_o\rho_o X_o^h + c_s X_s^h) = \nabla \cdot (\varepsilon\alpha_o\rho_o \mathbf{D}_o^h \nabla X_o^h). \quad (4)$$

The full notation for these and later equations is listed in Table 2 in appendix A. Briefly however,  $\alpha_y$  represents the pore-space volume fraction of phase  $y$ ;  $\rho_y$  the density of phase  $y$ ;  $X_y^x$  represents the mass fraction of component  $x$  in phase  $y$ ; and  $c_s$  the total mass of light and heavy oil per unit volume of the porous medium that is sorbed to the soil. The Darcy velocity vector  $\underline{q}$  is given by

$$\underline{q} = -\frac{kK_g}{\mu}(\nabla P - \rho_g \underline{g}), \quad (5)$$

where the relative permeability  $K_g$  has a van Genuchten form

$$K_g(\alpha_g) = \alpha_g^{1/2} [1 - (1 - \alpha_g)^{1/m}]^{2m} \quad (6)$$

(with  $m = 1 - 1/n$  for given  $n$ ),  $\mu$  is the dynamic viscosity of the air and light oil vapour mixture (taken to be constant and equal to that of air),  $\underline{g}$  is the acceleration due to gravity and the gas phase pressure,  $P$ , is given by

$$P = \rho_g RT \left( \frac{X_g^l}{\lambda^l} + \frac{X_g^a}{\lambda^a} \right). \quad (7)$$

In addition, note that the following consistency relations apply:

$$\alpha_g + \alpha_w + \alpha_o = 1, \quad (8)$$

$$X_g^l + X_g^a = 1, \quad (9)$$

$$X_w^l + X_w^w = 1, \quad (10)$$

$$X_o^l + X_o^h = 1, \quad (11)$$

$$X_s^l + X_s^h = 1. \quad (12)$$

The dispersivity tensors  $\mathbf{D}_g^l$ ,  $\mathbf{D}_w^l$  and  $\mathbf{D}_o^l$  in equation (1) are given by:

$$(\mathbf{D}_g^l)_{ij} = D_g^l T_g \delta_{ij} + a_T u \delta_{ij} + (a_L - a_T) \frac{u_i u_j}{u}, \quad (13)$$

$$(\mathbf{D}_w^l)_{ij} = D_w^l T_w \delta_{ij}, \quad (14)$$

$$(\mathbf{D}_o^l)_{ij} = D_o^l T_o \delta_{ij}, \quad (15)$$

where  $u$  is the magnitude of the vector  $\underline{u} = \underline{q}/(\varepsilon\alpha_g)$ , and the tortuosities are given by  $T_g = \varepsilon^{4/3}\alpha_g^{10/3}$ ,  $T_w = \varepsilon^{4/3}\alpha_w^{10/3}$  and  $T_o = \varepsilon^{4/3}\alpha_o^{10/3}$ . The molecular diffusivities  $D_g^l$ ,  $D_w^l$  and  $D_o^l$  are in reality functions of pressure and the phase compositions, but as a first approximation we assume they are constants. The remaining phase dispersivities in the equations of motion are derived via symmetry relations as follows:

$$\mathbf{D}_g^a = \mathbf{D}_g^l, \quad (16)$$

$$\mathbf{D}_w^w = \mathbf{D}_w^l, \quad (17)$$

$$\mathbf{D}_o^h = \mathbf{D}_o^l. \quad (18)$$

In order to obtain a numerical solution using the finite element discretization described in Section 3 it is convenient to consider concentrations, rather than mass fractions, as the dependent variables in our governing equations. Hence we define the concentrations  $C_y^x$  of component  $x$  in phase  $y$  by:

$$C_g^l = \rho_g X_g^l, \quad C_g^a = \rho_g X_g^a, \quad (19)$$

$$C_w^l = \rho_w X_w^l, \quad C_w^w = \rho_w X_w^w, \quad (20)$$

$$C_o^l = \rho_o X_o^l, \quad C_o^h = \rho_o X_o^h, \quad (21)$$

and

$$C_s^l = c_s X_s^l, \quad C_s^h = c_s X_s^h. \quad (22)$$

Thus, equations (1) to (4) may be written as

$$\begin{aligned} \frac{\partial}{\partial t}(\varepsilon\alpha_g C_g^l + \varepsilon\alpha_w C_w^l + \varepsilon\alpha_o C_o^l + C_s^l) + \nabla \cdot (C_g^l \underline{q}) = \\ \nabla \cdot (\varepsilon\alpha_g \mathbf{D}_g^l \nabla C_g^l + \varepsilon\alpha_w \mathbf{D}_w^l \nabla C_w^l + \varepsilon\alpha_o \mathbf{D}_o^l \nabla C_o^l) - \\ \nabla \cdot (\varepsilon\alpha_g \frac{C_g^l}{\rho_g} \mathbf{D}_g^l \nabla \rho_g + \varepsilon\alpha_o \frac{C_o^l}{\rho_o} \mathbf{D}_o^l \nabla \rho_o + \varepsilon\alpha_w \frac{C_w^l}{\rho_w} \mathbf{D}_w^l \nabla \rho_w), \end{aligned} \quad (23)$$

$$\frac{\partial}{\partial t}(\varepsilon\alpha_g C_g^a) + \nabla \cdot (C_g^a \underline{q}) = \nabla \cdot (\varepsilon\alpha_g \mathbf{D}_g^a \nabla C_g^a) - \nabla \cdot (\varepsilon\alpha_g \frac{C_g^a}{\rho_g} \mathbf{D}_g^a \nabla \rho_g), \quad (24)$$

$$\frac{\partial}{\partial t}(\varepsilon\alpha_w C_w^w) = \nabla \cdot (\varepsilon\alpha_w \mathbf{D}_w^w \nabla C_w^w) - \nabla \cdot (\varepsilon\alpha_w \frac{C_w^w}{\rho_w} \mathbf{D}_w^w \nabla \rho_w), \quad (25)$$

$$\frac{\partial}{\partial t}(\varepsilon\alpha_o C_o^h + C_s^h) = \nabla \cdot (\varepsilon\alpha_o \mathbf{D}_o^h \nabla C_o^h) - \nabla \cdot (\varepsilon\alpha_o \frac{C_o^h}{\rho_o} \mathbf{D}_o^h \nabla \rho_o). \quad (26)$$

Observing that

$$\rho_g = C_g^l + C_g^a, \quad (27)$$

$$\rho_w = C_w^l + C_w^w, \quad (28)$$

$$\rho_o = C_o^l + C_o^h, \quad (29)$$

$$\text{and } c_s = C_s^l + C_s^h, \quad (30)$$

we see that equations (23) to (26) involve 11 unknowns:

$$\alpha_g, \alpha_w, \alpha_o, C_g^l, C_g^a, C_w^l, C_w^w, C_o^l, C_o^h, C_s^l, C_s^h. \quad (31)$$

We therefore require 7 additional equations in order to close the system. These are derived from three consistency constraints and four equilibrium relations as follows.

- The volume fractions of the phases sum to 1:

$$\alpha_g + \alpha_w + \alpha_o = 1. \quad (32)$$

- The densities of the aqueous and oleic phases are assumed to be constant ( $\rho_w$  and  $\rho_o$  respectively):

$$C_w^w + C_w^l = \rho_w, \quad (33)$$

$$C_o^h + C_o^l = \rho_o. \quad (34)$$

- The partitioning of the light oil between the oleic and gas phases is given by:

$$\frac{C_g^l RT}{\lambda^l} = P_{sat}^l \left( \frac{C_o^l}{\lambda^l} \right) / \left( \frac{C_o^l}{\lambda^l} + \frac{C_o^h}{\lambda^h} \right), \quad (35)$$

where  $P_{sat}^l$  is the saturation vapour pressure of a pure liquid phase containing only light oil.

- The partitioning of the light oil between the gas and aqueous phases is given by:

$$\frac{H_w^l C_w^l}{\lambda^l} = \frac{C_g^l RT}{\lambda^l} \quad (36)$$

where  $H_w^l$  is Henry's Law constant for light oil in water.

- The partitioning of the light oil between the aqueous and sorbed phases is given by

$$\frac{C_s^l}{\rho_b} = K_D^l C_w^l \quad (37)$$

where  $\rho_b$  is the soil bulk density and  $K_D^l$  is the soil sorption coefficient for light oil.

- Finally, the partitioning of the heavy oil between the oleic and sorbed phases is given by

$$\frac{C_s^h}{\rho_b} = (K_D^h S_w^h) \left( \frac{C_o^h}{\lambda^h} \right) / \left( \frac{C_o^l}{\lambda^l} + \frac{C_o^h}{\lambda^h} \right). \quad (38)$$

Note that the heavy oil is assumed insoluble in water, and hence  $S_w^h$  should be zero. However, to simplify this model, we allow the product ( $K_D^h S_w^h$ ) to be non-zero.

Equations (32) to (38) give us the additional relations required in order to solve the PDE system (23) to (26). It is not difficult to see that these relations allow all of the dependent variables present in the equations to be expressed in terms of the four variables  $C_g^l$ ,  $C_g^a$ ,  $\alpha_w$  and  $\alpha_o$ , which we will now refer to as the *primary* variables for our numerical simulation. Note that since we have assumed immobility of the liquid phases, we do not need to be concerned about phase appearance/disappearance, which can be an additional source of complexity (see e.g. [14] and Subsection 4.3 below). Furthermore, our specific choice for the four primary variables is certainly not unique and other formulations, such as ones which include pressure as a primary variable for example, have been considered. The precise choice appears to have little effect on the computational efficiency for the simulations that we have performed.

## 2.2 Spatial domain and boundary and initial conditions

The spatial domain,  $\Omega$ , that we consider in this paper is defined, in cylindrical polar coordinates, by  $R_V < r < R_D$  and  $0 < z < H$ , as illustrated in Figure 1. This is based upon the assumption of cylindrical symmetry about a vent of radius  $R_V$  centred on the line  $r = 0$ . This vent is modeled by an impermeable vertical pipe with a gap, through which gas is extracted, over the interval  $Z_L \leq z \leq Z_U$ . The whole of the lower and outer boundaries are assumed to be impermeable, as is the region  $R_V < r < R_C$  on the top boundary  $z = H$ . The rest of the top boundary is the ground surface which is open to the atmosphere.

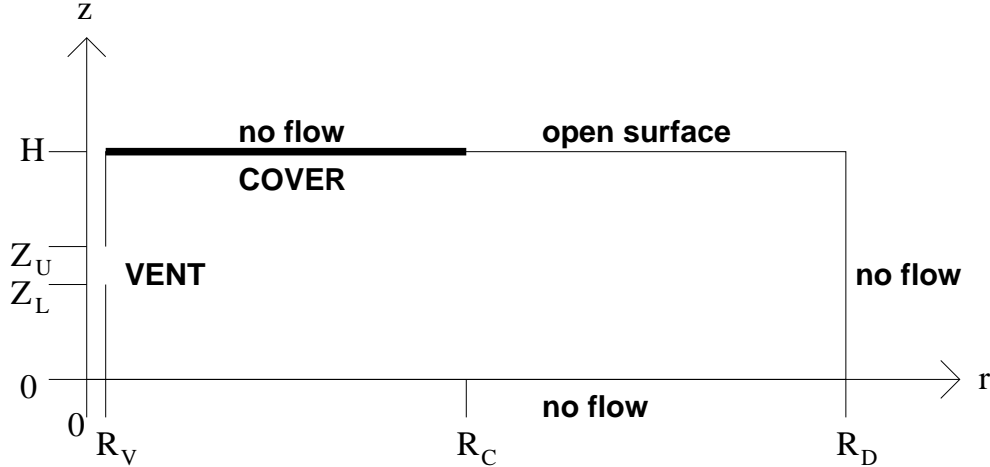


Figure 1: Spatial domain and boundary conditions.

For the initial conditions we assume that there has been a spillage of a heavy and light oil mixture (n-Dodecane and Toluene respectively in the simulations considered here), resulting in a residual saturation (10% by volume) of the liquid oil mixture, along with the already present 10% volume saturation of liquid water. It is assumed that the phases are in thermodynamic equilibrium, with (approximately) hydrostatic equilibrium in the gas phase. The gas phase pressure is assumed to be constant and equal to one atmosphere,  $P_\infty$ . (For strict hydrostatic equilibrium we require  $\partial P/\partial y = -\rho_g g$ .) The one exception to all of this is that towards the open surface part of the top boundary we force  $X_{lg}$  to tend to zero, the value in the atmosphere, in order to be consistent with the boundary conditions (see below). The gas extraction rate is zero initially (i.e. the pressure at the vent is  $P_\infty$ ) – it is increased (i.e. the pressure decreased) linearly to a steady value over an initial period of 1 second.

In order to obtain the bulk initial conditions it is assumed that the proportion (by weight) of each oil component in the unspilt oil is equal to that when partitioned in the porous medium. This implies that for the heavy oil say

$$X_h = \frac{\varepsilon \alpha_o C_o^h + C_s^h}{\varepsilon(\alpha_g C_g^l + \alpha_w C_w^l + \alpha_o C_o^l) + C_s^l + \varepsilon \alpha_o C_o^h + C_s^h}, \quad (39)$$

where  $X_h$  is the known mass fraction of heavy oil in the unspilt oil. Equation (39) is then solved along with the earlier algebraic relations and the assumed values for the volume fractions using a numerical nonlinear equation solver (routine C05NDF from the NAG Fortran Library [23]).

The boundary conditions are also summarized in Figure 1. The impermeability of the lower boundary gives zero normal flux conditions for all components:

$$\underline{n} \cdot [C_g^l \underline{q} + \Phi_g^l + \Phi_w^l + \Phi_o^l] = 0, \quad (40)$$

$$\underline{n} \cdot [C_g^a \underline{q} + \Phi_g^a] = 0, \quad (41)$$

$$\underline{n} \cdot [\Phi_w^w] = 0, \quad (42)$$

$$\underline{n} \cdot [\Phi_o^h] = 0, \quad (43)$$

where  $\underline{n}$  is the outward unit normal, and

$$\Phi_g^l = -\varepsilon \alpha_g \rho_g \mathbf{D}_g^l \underline{\nabla} X_g^l = -\varepsilon \alpha_g \mathbf{D}_g^l \underline{\nabla} C_g^l + \varepsilon \alpha_g \frac{C_g^l}{\rho_g} \mathbf{D}_g^l \underline{\nabla} \rho_g, \quad (44)$$

$$\Phi_g^a = -\varepsilon \alpha_g \rho_g \mathbf{D}_g^a \underline{\nabla} X_g^a = -\varepsilon \alpha_g \mathbf{D}_g^a \underline{\nabla} C_g^a + \varepsilon \alpha_g \frac{C_g^a}{\rho_g} \mathbf{D}_g^a \underline{\nabla} \rho_g, \quad (45)$$

$$\Phi_w^w = -\varepsilon \alpha_w \rho_w \mathbf{D}_w^w \underline{\nabla} X_w^w = -\varepsilon \alpha_w \mathbf{D}_w^w \underline{\nabla} C_w^w, \quad (46)$$

$$\Phi_o^h = -\varepsilon \alpha_o \rho_o \mathbf{D}_o^h \underline{\nabla} X_o^h = -\varepsilon \alpha_o \mathbf{D}_o^h \underline{\nabla} C_o^h. \quad (47)$$

At the outer boundary,  $r = R_D$ , it is assumed there is a ‘‘caisson’’ surface, again impermeable to all components, and hence the boundary conditions are exactly as given above for the lower boundary. These conditions also apply for those regions ( $0 \leq z < Z_L$  and  $Z_U < z \leq H$ ) of the inner boundary,  $r = R_V$ , where no extraction takes place. On the rest of this boundary,  $z_L \leq z \leq Z_U$ , whilst there is zero flux of water or heavy oil (conditions (46) and (47) respectively), there is a flow of light oil and air components in the gas phase. This is driven by a pressure difference and so we model this with a pressure boundary condition of the form

$$P = P_v; , \quad (48)$$

i.e., from (7) and (19),

$$\frac{C_g^l}{\lambda^l} + \frac{C_g^a}{\lambda^a} = \frac{P_v}{RT}. \quad (49)$$

In addition to this we add a fourth and final boundary condition at the vent which requires

$$\underline{n} \cdot \underline{\nabla} X_g^l = 0. \quad (50)$$

This condition corresponds to an assumption of zero diffusive flux at the vent.

Finally we consider the open surface part of the upper boundary ( $R_C \leq r \leq R_D$  and  $z = H$ ). Again, there can be no flow of water or heavy oil so conditions (46) and (47) are still appropriate. As with the vent we also impose a pressure boundary condition, but this time it takes the form  $P = P_\infty$ , i.e.

$$\frac{C_g^l}{\lambda^l} + \frac{C_g^a}{\lambda^a} = \frac{P_\infty}{RT}. \quad (51)$$

For the fourth condition we impose a value for  $X_g^l$  at the boundary:

$$X_g^l = 0. \quad (52)$$

This is based on the assumption that the mass fraction of light oil in the atmosphere is zero and that  $X_g^l$  is continuous at the boundary. Although this assumption is slightly questionable (and has the effect of creating a transition layer just under the surface rather than just above it, as in [15] for example) this simple condition appears to work well – and certainly permits the flow of gas in either direction through the boundary as is appropriate.

Details of the implementation of these boundary conditions are provided in the following section once details of the finite element discretization scheme and the time integration software have been covered. Some typical values of the constants that have been introduced in this section may be found in appendix B.

### 3 Adaptive numerical solution

For the numerical solution of the problem we employ the SPRINT2D adaptive partial differential equation (PDE) solver (see [6, 7]). This code uses the Method of Lines (MOL) whereby each of the PDEs are first discretized in space, resulting in a system of ordinary differential equations (ODEs), and the general purpose initial value ODE software is used to solve this system as efficiently as possible. Some details of SPRINT2D are presented in Subsection 3.1, including a description of how the user may provide their own spatial discretization via a residual routine and how adaptivity is allowed in both space and time. Subsection 3.2 then concentrates on the finite element algorithm that we use for the spatial discretization of this particular problem, and also presents a discussion of stability and the generation of appropriate error indicators.

#### 3.1 Method of Lines software

The SPRINT2D code is designed to provide a straightforward interface to a number of different algorithms for the adaptive solution of initial value ODEs when the MOL is used to solve initial value partial differential equations in two space dimensions. This is achieved by providing a triangulation of the spatial domain and then allowing the user to define their own discretization of the PDE problem that they are solving on this triangulation. This discretization will typically be based upon finite element or finite volume techniques and will yield a system of ODEs of the general form:

$$\underline{F}(\dot{\underline{y}}(t), \underline{y}(t), t) = \underline{0}; \quad \underline{y}(t_0) = \underline{y}_0, \quad (53)$$

where the vector  $\underline{y}(t)$  contains each of the spatial degrees of freedom, the dot above the  $\underline{y}$  denotes differentiation with respect to time,  $t$ , and  $\underline{y}_0$  comes from the initial conditions at time  $t = t_0$ .

As already indicated, the SPRINT2D software provides a number of different schemes for solving the ODE system (53): for example, a variable order BDF (backward differentiation formula) algorithm or an adaptive  $\theta$ -method code (i.e. in which the value of  $\theta$  may vary at each time-step) may be used. In either case the size of the time-step is allowed to vary automatically based upon a local error control strategy (this strategy is also used to select the order in the BDF algorithm). No matter which of the time integration algorithms is actually being used at any time, the user is only ever required to provide a single residual routine which is able to evaluate the function  $\underline{F}$  for given estimates of  $\dot{\underline{y}}$ ,  $\underline{y}$  and  $t$ . To illustrate this we consider the simplest of the BDF formulae (the first order scheme) by way of an example. (This also corresponds to the  $\theta$ -method with  $\theta = 1$ , usually known as the backward Euler method.)

The backward Euler method for solving (53) is based upon repeated application of the following formula:

$$\underline{F}\left(\frac{\underline{y}_n - \underline{y}_{n-1}}{k_n}, \underline{y}_n, t_n\right) = \underline{0}, \quad (54)$$

where  $\underline{y}_n \approx \underline{y}(t_n)$  and  $t_n = t_0 + \sum_{i=1}^n k_i$ . In practice this is achieved by maintaining estimates of both  $\underline{y}(t_n)$  and  $\dot{\underline{y}}(t_n)$  (as  $\underline{y}_n$  and  $\dot{\underline{y}}_n$  respectively) as follows.

1. Given  $\underline{y}_0$  and  $t_0$  solve  $\underline{F}(\dot{\underline{y}}_0, \underline{y}_0, t_0) = \underline{0}$  for  $\dot{\underline{y}}_0$ .
2. For  $n = 1, 2, 3, \dots$  until complete:
  - (a) set predictors  $\dot{\underline{y}}_n^P = \dot{\underline{y}}_{n-1}$  and  $\underline{y}_n^P = \underline{y}_{n-1} + k_n \dot{\underline{y}}_{n-1}$ ,
  - (b) solve  $\underline{F}(\dot{\underline{y}}_n^P + \underline{\alpha}, \underline{y}_n^P + k_n \underline{\alpha}, t_n) = \underline{0}$  for  $\underline{\alpha}$ ,
  - (c) set solutions  $\dot{\underline{y}}_n = \dot{\underline{y}}_n^P + \underline{\alpha}$  and  $\underline{y}_n = \underline{y}_n^P + k_n \underline{\alpha}$ .

The significant computational cost at each step of this algorithm is in the solution of the nonlinear system

$$\underline{F}(\underline{\dot{y}}_n^P + \underline{\alpha}, \underline{y}_n^P + k_n \underline{\alpha}, t_n) = \underline{0} \quad (55)$$

for  $\underline{\alpha}$ . This is achieved using a quasi-Newton algorithm which therefore requires an estimate of the Jacobian matrix

$$\frac{\partial \underline{F}}{\partial \underline{\dot{y}}} + k_n \frac{\partial \underline{F}}{\partial \underline{y}}. \quad (56)$$

If the user does not wish to provide a function for the evaluation of this Jacobian matrix it may be estimated from within the code using finite difference approximations. If the latter option is selected then a sparsity pattern for the Jacobian may be provided to increase the efficiency of both the approximation and the factorization stages.

When other time integration schemes are used similar solution algorithms may be derived (see [4, 5] for example) in which residual and optional Jacobian routines need to be provided. The results presented in Section 4 of this paper were all computed using the adaptive  $\theta$ -method of [5].

At the end of each time-step the SPRINT2D software also allows the user to adapt the spatial triangulation if desired. This is achieved by calling a monitor routine which has all of the latest solution details passed to it. Local error estimates or indicators may then be applied on each element and those elements for which the error is too large can be marked for refinement, whilst those for which the error is unnecessarily small can be marked for coarsening. The mesh adaptivity algorithm that is then called is based upon the hierarchical  $h$ -refinement approach first used by Löhner in [22] and illustrated in Figure 2. As may be observed, each triangle that is refined is divided into four children of a similar shape. In order to maintain the consistency of the mesh neighbouring triangles are not allowed to differ in refinement level by more than one, and some triangles are temporarily bisected in order to prevent nodes from lying on the midpoints of triangle edges. These bisected triangles must always be restored to their original state before they may be refined so as to prevent the mesh quality from deteriorating. It should be noted however that a potentially damaging side-effect of using this adaptive strategy is that the total mass of light oil is not guaranteed to be conserved when the mesh is either coarsened or refined. In order to monitor any errors that might arise from this non-conservation therefore, we suggest that it is worth incurring the small additional overhead of calculating the total mass in the domain both before and after each call to the mesh adaptivity routine. If the cumulative difference remains small then we can be confident that our results are not significantly affected by this side-effect.

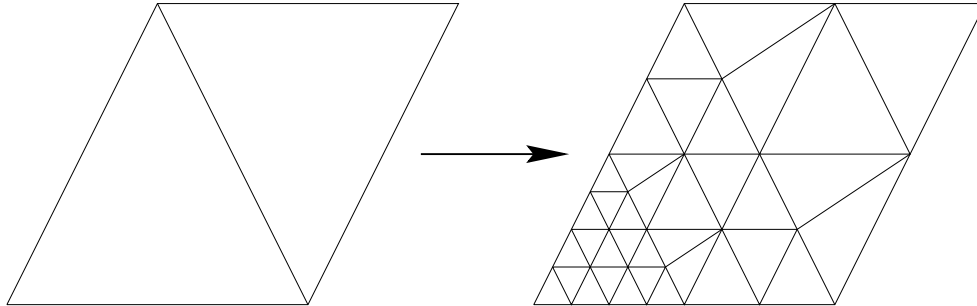


Figure 2: Illustration of the  $h$ -refinement algorithm used by SPRINT2D.

From this brief description it may be seen that SPRINT2D offers a flexible MOL solution environment which allows adaptivity in both space and time, and which is relatively straightforward to use. We are now in a position to present details of the adaptive finite element solution of the venting problem described in the previous section.

### 3.2 Spatial discretization and adaptivity

The spatial discretization that we use for the problem outlined in Section 2 is based upon an upwind modification of the Galerkin finite element method. Recall from Subsection 3.1 that the spatial discretization is defined by providing a residual routine which returns the value of

$$\underline{F}(\underline{\dot{y}}, \underline{y}, t) \quad (57)$$

for given values of  $\underline{\dot{y}}$ ,  $\underline{y}$  and  $t$ . In this particular work we choose to use a piecewise linear representation of each of the primary variables ( $C_g^l$ ,  $C_g^a$ ,  $\alpha_w$  and  $\alpha_o$ ) on the underlying triangulation of  $\Omega$ . This means that the vector  $\underline{y}$  corresponds to a set of values of each of these four variables at each vertex of the mesh, and  $\underline{\dot{y}}$  holds estimates of the temporal derivatives of these variables at the mesh points. Hence each of these vectors is of length  $4N$  where  $N$  is the total number of vertices in the mesh.  $\underline{F}$  is also a vector of length  $4N$  which is made up of four entries for each vertex.

To see how the residual vector  $\underline{F}$  is calculated we first derive a weak form of equations (23) to (26) on the region  $\Omega$  (with boundary  $\Gamma$ ):

$$\begin{aligned} 0 &= \int_{\Omega} \varepsilon(\alpha_g \dot{C}_g^l + \dot{\alpha}_g C_g^l) W d\Omega + \int_{\Omega} \varepsilon(\alpha_w \dot{C}_w^l + \dot{\alpha}_w C_w^l) W d\Omega \\ &+ \int_{\Omega} \varepsilon(\alpha_o \dot{C}_o^l + \dot{\alpha}_o C_o^l) W d\Omega + \int_{\Omega} \dot{C}_s^l W d\Omega - \int_{\Omega} C_g^l \underline{q} \cdot \underline{\nabla} W d\Omega + \oint_{\Gamma} C_g^l \underline{q} \cdot \underline{n} W d\Gamma \\ &+ \int_{\Omega} \varepsilon \alpha_g (\mathbf{D}_g^l \underline{\nabla} C_g^l - \frac{C_g^l}{\rho_g} \mathbf{D}_g^l \underline{\nabla} \rho_g) \cdot \underline{\nabla} W d\Omega - \oint_{\Gamma} \varepsilon \alpha_g (\mathbf{D}_g^l \underline{\nabla} C_g^l - \frac{C_g^l}{\rho_g} \mathbf{D}_g^l \underline{\nabla} \rho_g) \cdot \underline{n} W d\Gamma \\ &+ \int_{\Omega} \varepsilon \alpha_w (\mathbf{D}_w^l \underline{\nabla} C_w^l) \cdot \underline{\nabla} W d\Omega - \oint_{\Gamma} \varepsilon \alpha_w (\mathbf{D}_w^l \underline{\nabla} C_w^l) \cdot \underline{n} W d\Gamma \\ &+ \int_{\Omega} \varepsilon \alpha_o (\mathbf{D}_o^l \underline{\nabla} C_o^l) \cdot \underline{\nabla} W d\Omega - \oint_{\Gamma} \varepsilon \alpha_o (\mathbf{D}_o^l \underline{\nabla} C_o^l) \cdot \underline{n} W d\Gamma, \end{aligned} \quad (58)$$

$$\begin{aligned} 0 &= \int_{\Omega} \varepsilon(\alpha_g \dot{C}_g^a + \dot{\alpha}_g C_g^a) W d\Omega - \int_{\Omega} C_g^a \underline{q} \cdot \underline{\nabla} W d\Omega + \oint_{\Gamma} C_g^a \underline{q} \cdot \underline{n} W d\Gamma \\ &+ \int_{\Omega} \varepsilon \alpha_g (\mathbf{D}_g^a \underline{\nabla} C_g^a - \frac{C_g^a}{\rho_g} \mathbf{D}_g^a \underline{\nabla} \rho_g) \cdot \underline{\nabla} W d\Omega - \oint_{\Gamma} \varepsilon \alpha_g (\mathbf{D}_g^a \underline{\nabla} C_g^a - \frac{C_g^a}{\rho_g} \mathbf{D}_g^a \underline{\nabla} \rho_g) \cdot \underline{n} W d\Gamma, \end{aligned} \quad (59)$$

$$\begin{aligned} 0 &= \int_{\Omega} \varepsilon(\alpha_w \dot{C}_w^w + \dot{\alpha}_w C_w^w) W d\Omega \\ &+ \int_{\Omega} \varepsilon \alpha_w (\mathbf{D}_w^w \underline{\nabla} C_w^w) \cdot \underline{\nabla} W d\Omega - \oint_{\Gamma} \varepsilon \alpha_w (\mathbf{D}_w^w \underline{\nabla} C_w^w) \cdot \underline{n} W d\Gamma, \end{aligned} \quad (60)$$

$$\begin{aligned} 0 &= \int_{\Omega} \varepsilon(\alpha_o \dot{C}_o^h + \dot{\alpha}_o C_o^h) W d\Omega + \int_{\Omega} \dot{C}_s^h W d\Omega \\ &+ \int_{\Omega} \varepsilon \alpha_o (\mathbf{D}_o^h \underline{\nabla} C_o^h) \cdot \underline{\nabla} W d\Omega - \oint_{\Gamma} \varepsilon \alpha_o (\mathbf{D}_o^h \underline{\nabla} C_o^h) \cdot \underline{n} W d\Gamma, \end{aligned} \quad (61)$$

where use has been made of (33) and (34) and the dot above a variable again denotes differentiation with respect to  $t$ . The vector  $\underline{n}$  is the unit outward normal, the test functions  $W(r, z) \in H^1(\Omega)$  and the second order terms have been integrated by parts using Green's theorem.

In the Galerkin finite element approximation we replace each of the primary variables by a linear combination of the usual linear ‘‘hat’’ basis functions  $L_i$  on the current mesh. (Here  $L_i$  is linear on each triangular element and  $L_i(\underline{x}_j) = \delta_{ij}$  for  $i, j \in \{1, \dots, N\}$ , where  $\delta_{ij}$  is the Kronecker delta and  $\underline{x}_j$  is the position vector of node  $j$ .) Hence we obtain the following approximations:

$$C_g^l \approx \tilde{C}_g^l = \sum_{i=1}^n \tilde{C}_{g,i}^l L_i, \quad (62)$$

$$C_g^a \approx \tilde{C}_g^a = \sum_{i=1}^n \tilde{C}_{g,i}^a L_i, \quad (63)$$

$$\alpha_w \approx \tilde{\alpha}_w = \sum_{i=1}^n \tilde{\alpha}_{w,i} L_i, \quad (64)$$

$$\alpha_o \approx \tilde{\alpha}_o = \sum_{i=1}^n \tilde{\alpha}_{o,i} L_i. \quad (65)$$

Note that the coefficients,  $\tilde{C}_{g,i}^l$ ,  $\tilde{C}_{g,i}^a$ ,  $\tilde{\alpha}_{w,i}$  and  $\tilde{\alpha}_{o,i}$ , in these four expansions form the vector  $\underline{y}$  in the residual (57). Also observe that conditions (32) to (38) allow for all of the variables in (58) to (61) to be expressed in terms of these coefficients. Finally, the test functions  $W$  in (58) and (59) are replaced by  $L_j$  for those  $j$  which correspond to vertices *not* on the open surface or at the vent, and the test functions  $W$  in (60) to (61) are replaced by  $L_j$  for  $j = 1, \dots, N$ . This yields  $4N - 2M$  residual equations (where  $M$  is the number of mesh vertices on the open surface or at the vent), with the remaining  $2M$  residual equations being determined by Dirichlet boundary conditions of the form

$$\tilde{P} = \text{prescribed value} \quad (66)$$

$$\text{and} \quad \tilde{X}_g^l = \text{prescribed value} \quad (67)$$

at these  $M$  nodes (see Subsection 2.2 for details of the prescribed values). From (7), (19) and (27) these last two conditions easily reduce to simple constraints on  $\tilde{C}_g^l$  and  $\tilde{C}_g^a$ .

It should be noted that all of the boundary integral terms in the residual equations are zero due either to the zero flux boundary conditions (again refer to Subsection 2.2 for details) or the fact that in (58) and (59)  $W = L_j$  is always zero on the parts of the boundary where there are non-zero fluxes of light oil or air (i.e. at the open surface or the vent). Hence the four residual equations which correspond to each vertex,  $j$ , not on the open surface or at the vent, have the following form (where, as in (66) and (67), the tilde above each of the secondary variables indicates that these are derived from the given piecewise linear approximations to the primary variables and their temporal derivatives (i.e. from  $\underline{y}$  and  $\dot{\underline{y}}$  respectively)) using (32) to (38):

$$\begin{aligned} r_1(j) = & \sum_{\epsilon=1}^E \left[ \int_{\Omega_\epsilon} \epsilon (\tilde{\alpha}_g \dot{\tilde{C}}_g^l + \dot{\tilde{\alpha}}_g \tilde{C}_g^l) L_j d\Omega + \int_{\Omega_\epsilon} \epsilon (\tilde{\alpha}_w \dot{\tilde{C}}_w^l + \dot{\tilde{\alpha}}_w \tilde{C}_w^l) L_j d\Omega \right. \\ & + \int_{\Omega_\epsilon} \epsilon (\tilde{\alpha}_o \dot{\tilde{C}}_o^l + \dot{\tilde{\alpha}}_o \tilde{C}_o^l) L_j d\Omega + \int_{\Omega_\epsilon} \dot{\tilde{C}}_s^l L_j d\Omega - \int_{\Omega_\epsilon} \tilde{C}_g^l \underline{\tilde{q}} \cdot \underline{\nabla} L_j d\Omega \\ & + \int_{\Omega_\epsilon} \epsilon \tilde{\alpha}_g (\tilde{\mathbf{D}}_g^l \underline{\nabla} \tilde{C}_g^l - \frac{\tilde{C}_g^l}{\tilde{\rho}_g} \tilde{\mathbf{D}}_g^l \underline{\nabla} \tilde{\rho}_g) \cdot \underline{\nabla} L_j d\Omega \\ & \left. + \int_{\Omega_\epsilon} \epsilon \tilde{\alpha}_w (\tilde{\mathbf{D}}_w^l \underline{\nabla} \tilde{C}_w^l) \cdot \underline{\nabla} L_j d\Omega + \int_{\Omega_\epsilon} \epsilon \tilde{\alpha}_o (\tilde{\mathbf{D}}_o^l \underline{\nabla} \tilde{C}_o^l) \cdot \underline{\nabla} L_j d\Omega \right], \quad (68) \end{aligned}$$

$$\begin{aligned} r_2(j) = & \sum_{\epsilon=1}^E \left[ \int_{\Omega_\epsilon} \epsilon (\tilde{\alpha}_g \dot{\tilde{C}}_g^a + \dot{\tilde{\alpha}}_g \tilde{C}_g^a) L_j d\Omega - \int_{\Omega_\epsilon} \tilde{C}_g^a \underline{\tilde{q}} \cdot \underline{\nabla} L_j d\Omega \right. \\ & \left. + \int_{\Omega_\epsilon} \epsilon \tilde{\alpha}_g (\tilde{\mathbf{D}}_g^a \underline{\nabla} \tilde{C}_g^a - \frac{\tilde{C}_g^a}{\tilde{\rho}_g} \tilde{\mathbf{D}}_g^a \underline{\nabla} \tilde{\rho}_g) \cdot \underline{\nabla} L_j d\Omega \right], \quad (69) \end{aligned}$$

$$r_3(j) = \sum_{\epsilon=1}^E \left[ \int_{\Omega_\epsilon} \epsilon (\tilde{\alpha}_w \dot{\tilde{C}}_w^w + \dot{\tilde{\alpha}}_w \tilde{C}_w^w) L_j d\Omega + \int_{\Omega_\epsilon} \epsilon \tilde{\alpha}_w (\tilde{\mathbf{D}}_w^w \underline{\nabla} \tilde{C}_w^w) \cdot \underline{\nabla} L_j d\Omega \right], \quad (70)$$

$$r_4(j) = \sum_{\epsilon=1}^E \left[ \int_{\Omega_\epsilon} \epsilon (\tilde{\alpha}_o \dot{\tilde{C}}_o^h + \dot{\tilde{\alpha}}_o \tilde{C}_o^h) L_j d\Omega + \int_{\Omega_\epsilon} \dot{\tilde{C}}_s^h L_j d\Omega + \int_{\Omega_\epsilon} \epsilon \tilde{\alpha}_o (\tilde{\mathbf{D}}_o^h \underline{\nabla} \tilde{C}_o^h) \cdot \underline{\nabla} L_j d\Omega \right]. \quad (71)$$

In the above equations it has been assumed that there are a total of  $E$  triangular elements,  $\Omega_1$  to  $\Omega_E$ , and, in practice, the integrals on each of these elements are approximated using a quadrature rule (e.g. see [12] for some examples of Gaussian quadrature rules on triangles).

## Numerical stabilization

As mentioned previously, the strong convective element in this venting problem is known to give rise to numerical instabilities when using standard discretization methods such as the Galerkin finite element method described above (see e.g. [14, 19]). As such we require some form of stabilization scheme. Various methods exist for improving stability in finite element methods, such as Petrov-Galerkin methods, e.g. SUPG (Streamline Upwinded Petrov-Galerkin) [17], or Galerkin Least Squares [24]. A method which has been successfully used for this type of problem (see e.g. [14, 19]) is an upstream (or upwind) weighting technique, in which greater emphasis is placed on information coming from the upstream direction. We use a similar upwind scheme to that used by Forsyth and Shao [14] in their “control volume finite element” method. In this scheme the direction of upwinding is determined by looking at the pressure gradient along each edge of the element.

In the standard Galerkin scheme described above the integrals in the residual equations (68) to (71) are calculated using a standard quadrature rule on each triangle (see [12] for example). This is achieved by first calculating the four primary variables at each quadrature point, using the given nodal values and the fact that the trial functions are linear on each element, and then using (32) to (38) in order to calculate the values of all of the secondary variables at these points.

In the upstream formulation we no longer use a linear interpolant for calculating the quadrature point values that are used for the integrals of the convective terms (i.e.  $\int_{\Omega_e} \tilde{C}_g^l \tilde{q} \cdot \nabla L_j d\Omega$  in (68) and  $\int_{\Omega_e} \tilde{C}_g^a \tilde{q} \cdot \nabla L_j d\Omega$  in (69)), but instead we weight the interpolant in the upstream direction (see Figure 3 for example). Following Forsyth and Shao [14] we apply this upstream shift only in the evaluation of  $\tilde{C}_g^l$  and  $\tilde{C}_g^a$  appearing explicitly in these two convective terms. The Darcy velocity components are calculated from (5) using the unweighted linearly interpolated values of  $\tilde{C}_g^l$  and  $\tilde{C}_g^a$  at the quadrature points. The required modifications to the residual calculations in the code are therefore minor, and it is found in practice that this scheme has a significant stabilizing effect. Furthermore, our use of mesh refinement ensures that elements are always small in regions where solution gradients are large and so the smearing that is often associated with simple stabilization schemes such as this does not appear to degrade the quality of our results (but see, for example, [29] for a scheme designed to reduce the smearing effects of the stabilization still further).

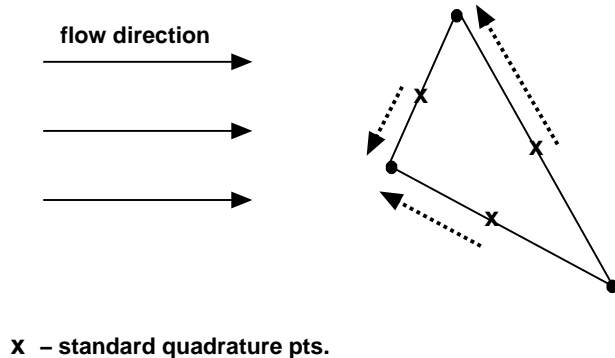


Figure 3: Illustration of the weighting of the quadrature point values used in the upstream method.

## Spatial mesh adaptivity

As outlined in Subsection 3.1 the form of adaptivity used by SPRINT2D is one of static rezoning in which the mesh is adapted at discrete times, and the solution is interpolated onto the new mesh before the next time step is taken. The decision about when and where to refine or derefine (coarsen) the mesh is made on the basis of spatial error estimates. Further details can be found in [7].

For the scheme described here the spatial error estimate is based on a simple weighted average of the spatial derivatives of the primary variables, i.e.

$$err(e) = \text{area}(\Omega_e) \left[ w_1 \left( \left| \frac{\partial \tilde{C}_g^l}{\partial r} \right| + \left| \frac{\partial \tilde{C}_g^l}{\partial z} \right| \right) + w_2 \left( \left| \frac{\partial \tilde{C}_g^a}{\partial r} \right| + \left| \frac{\partial \tilde{C}_g^a}{\partial z} \right| \right) + w_3 \left( \left| \frac{\partial \tilde{\alpha}_w}{\partial r} \right| + \left| \frac{\partial \tilde{\alpha}_w}{\partial z} \right| \right) + w_4 \left( \left| \frac{\partial \tilde{\alpha}_o}{\partial r} \right| + \left| \frac{\partial \tilde{\alpha}_o}{\partial z} \right| \right) \right], \quad (72)$$

for each element  $\Omega_e$ . The value  $err(e)$  is then compared against an error ‘target’ supplied by the user, and the element is flagged for refinement or derefinement accordingly. An appropriate value for the target error is in practice found by trial and error, although we have found that once a suitable value has been obtained it is generally robust enough to be applied for a wide range of problems. In the example results presented in the next section a value of 0.025 has been used for the target, with weight vector  $\underline{w} = (2.5, 2.5, 0.0, 0.0)$ . Hence the error estimate is biased towards the derivatives of  $\tilde{C}_g^l$  and  $\tilde{C}_g^a$ . More sophisticated error estimation schemes could be implemented, however, we believe that this simple scheme is perfectly adequate for this problem over a wide range of parameter values.

The user must specify an initial coarse mesh and a maximum level of refinement. SPRINT2D will adapt the initial mesh as appropriate before proceeding with the integration. Other options related to the adaptivity include the parameters for the so-called “safety layer”, being the extra layers of refinement placed around elements when they are refined. This safety layer ensures that the refined area is large enough to accommodate the possible movement of the high error part of the solution before the next remesh. This is particularly important for problems with steep moving fronts, as in this problem. A “safety skip” parameter determines how often to update this safety layer, it being inefficient to update at every remesh.

## 4 Results and Discussion

In this final section of the paper we present some typical computational results obtained using the models and numerical methods described above. The use of adaptivity in both space and time is shown to be highly beneficial in terms of delivering results of the same accuracy as on fixed meshes but at significantly less computational cost.

### 4.1 Adaptive simulations

For the numerical simulations considered here the parameters for the model problem are chosen as in appendix B. Two different situations are considered. In the first there is an impermeable cover of radius 4.0m present on the upper boundary ( $R_c = 4.0m$ ), whereas this cover is not present in the second case (so  $R_c = R_v = 0.3m$ ). Both runs are undertaken with the initial conditions described in Subsection 2.2 and so begin with a transition layer near the open part of the upper boundary where the mass fraction of light oil in the gas phase increases from zero to its bulk value in the interior of the domain. The runs continue for a simulated time of 350000 seconds (almost 100 hours) by which time the mass of the light oil contaminant left in the region is beginning to level out.

Figure 4 shows a series of contour plots of the calculated value of the mass fraction of light oil in the gas phase (i.e.  $X_g^l$ ) for the first case (i.e. with a cover present) at increasing times (from top to

bottom). The initial plot clearly shows the starting condition described above. As time progresses we then see clean air (i.e.  $X_g^l = 0.0$ ) being pulled in through the open surface towards the vent at the inner boundary. Throughout the simulation there is a sharp front which represents the change in concentration from zero to the bulk value of 0.015. This transition region is represented well using our adaptive meshing algorithm as shown in Figure 5. This shows the meshes which correspond to each of the solutions in Figure 4. The initial mesh is coarse throughout (25 by 5 elements), other than in the regions of the boundary layer and the vent (where we have imposed some initial refinement). The final mesh clearly shows the benefit of allowing coarsening of the mesh as well as refinement since this has reduced the number of elements required in the region behind the front, where  $X_g^l = 0.0$ . It is apparent in Figure 4 (and reflected in the meshes in Figure 5) that there is an isolated peak in the solution at the top left hand corner. This is due to a stagnation point (zero velocity) at the centre of the cover. There is also a transient peak in solution at the singular point at the top corner of the vent. Neither of these features cause any long term degradation of the solution.

A similar transition front for  $X_g^l$  is present when the simulation is repeated without the presence of a cover on the upper boundary and so the same advantages of adaptivity may be observed. Indeed in the case of no cover the mesh coarsens in a more extensive region behind the contamination front since there is no stagnation point at the top left hand corner in this case.

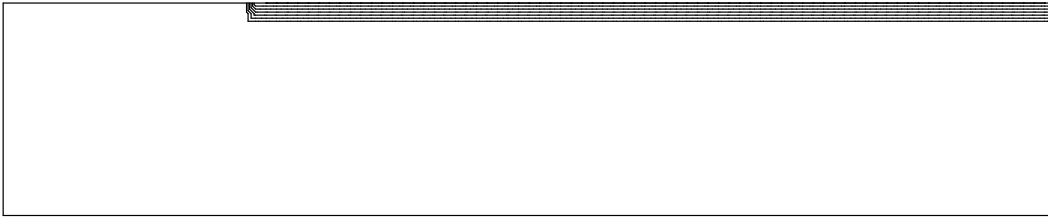
The reason that a cover is likely to be present in practice however may be seen from Figure 6 which shows plots of the total mass of light oil in the domain versus time for each of these simulations. The use of a cover can clearly be seen to increase the total amount of contaminant extracted, at least in the case where a single vent is used. The small difference in the initial mass of light oil in each simulation is a result of the different initial conditions used in each case – the light oil concentration falls to zero in the open surface region only. Gamliel and Abdul [16] consider optimal surface sealing and related factors in detail. This figure also shows that despite the fact that the gas being extracted has quite a low fraction of light oil present after almost 100 hours of venting (as indicated by the gradient of each curve), the contaminated region is far from clean. This suggests that a single vent will not generally be adequate (see Subsection 4.3 below).

## 4.2 Performance

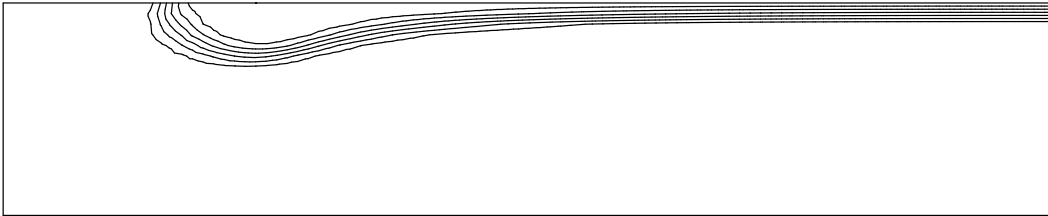
In order to assess the performance of our adaptive algorithm we compared the solutions obtained above against similar solutions obtained on a fixed spatial mesh with the same resolution as the finest mesh used in the adaptive simulations. In order to keep the run-times reasonable the maximum level of refinement permitted was set to just two (see Figure 5 for example) and so the fixed mesh used in these comparisons consisted of 4000 elements and 2121 nodes (this compares with the 619 to 920 nodes that were present at different times in the adaptive simulation). Even for the fixed spatial mesh it was decided to use adaptive time-stepping since undertaking simulations such as these with fixed time steps would be likely to lead to unnecessary temporal errors or excessively small step sizes (or both).

In terms of accuracy, Figure 7 suggests that, as one would expect, there is little to choose either qualitatively or quantitatively between the adaptive and the fixed mesh solutions. This figure shows the total mass of light oil present in the domain against time (for the run with the cover present) for both of these simulations. The same similarities are present when contour plots of adaptive and fixed-mesh solutions are contrasted. The use of spatial refinement has ensured that when the resolution of a fine grid is required to maintain accuracy it is provided. Furthermore, as explained in subsection 3.1, throughout the adaptive run we have monitored the effects of non-conservation of mass when the mesh is refined and/or coarsened. For this (typical) simulation the sum of the mass differences before and after each adaptation is just 0.0056% of the total mass of light oil initially present, and the sum of the absolute values of these differences is only 0.0090% of the total mass. These figures provide further evidence of the high accuracy of the adaptive simulation.

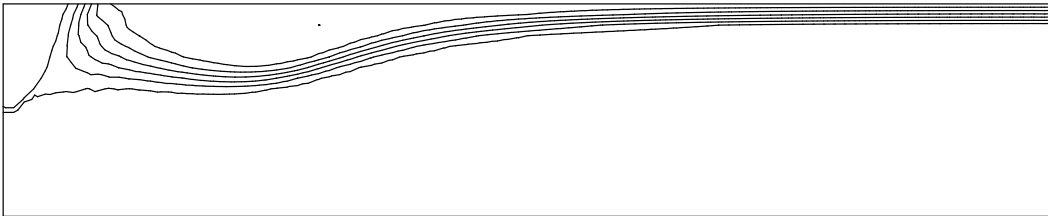
time = 0.0 s



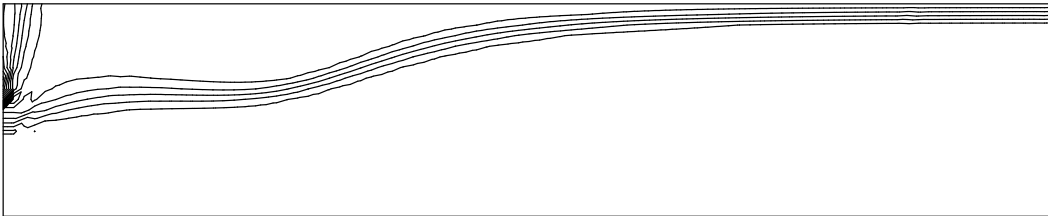
time = 50000 s



time = 100000 s



time = 150000 s



time = 350000 s

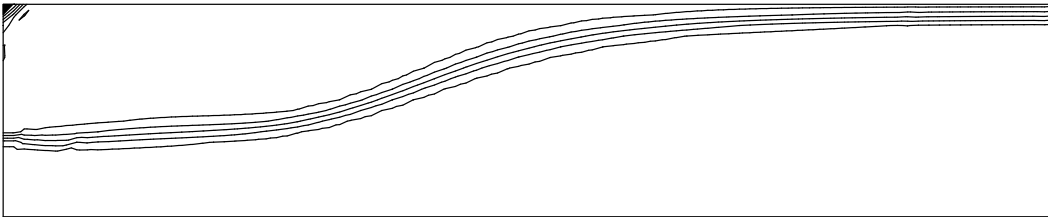
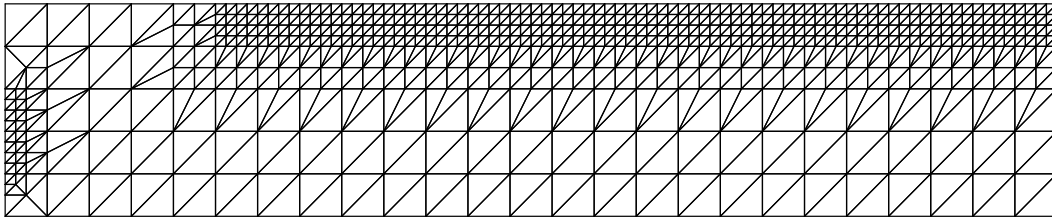
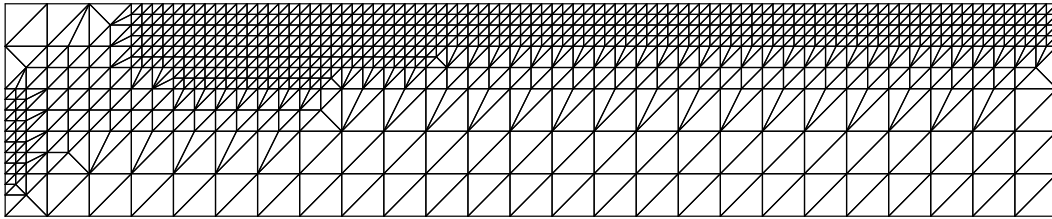


Figure 4: Contour plots showing the calculated  $X_g^l$  solution to the problem with the cover present at various times.

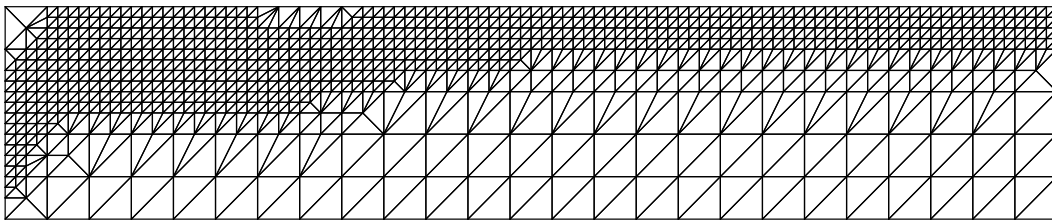
time = 0.0 s



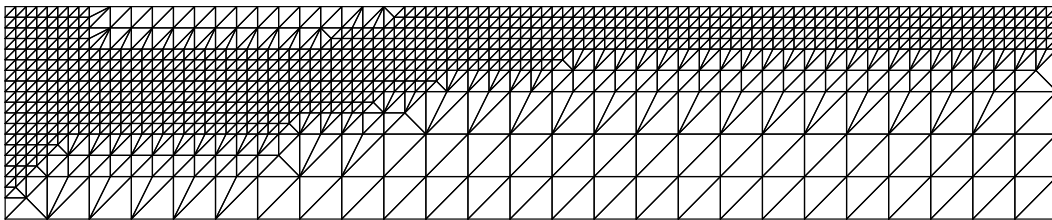
time = 50000 s



time = 100000 s



time = 150000 s



time = 350000 s

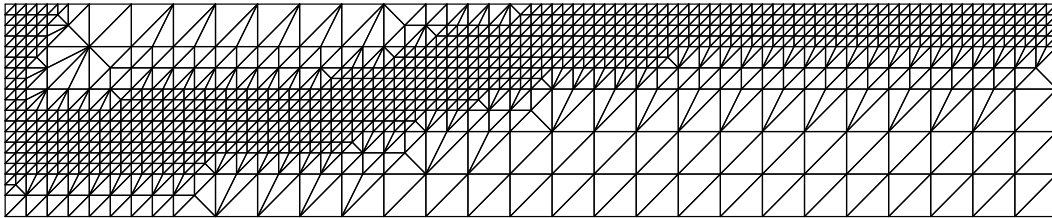


Figure 5: Meshes corresponding to the contour solutions illustrated in Figure 4.

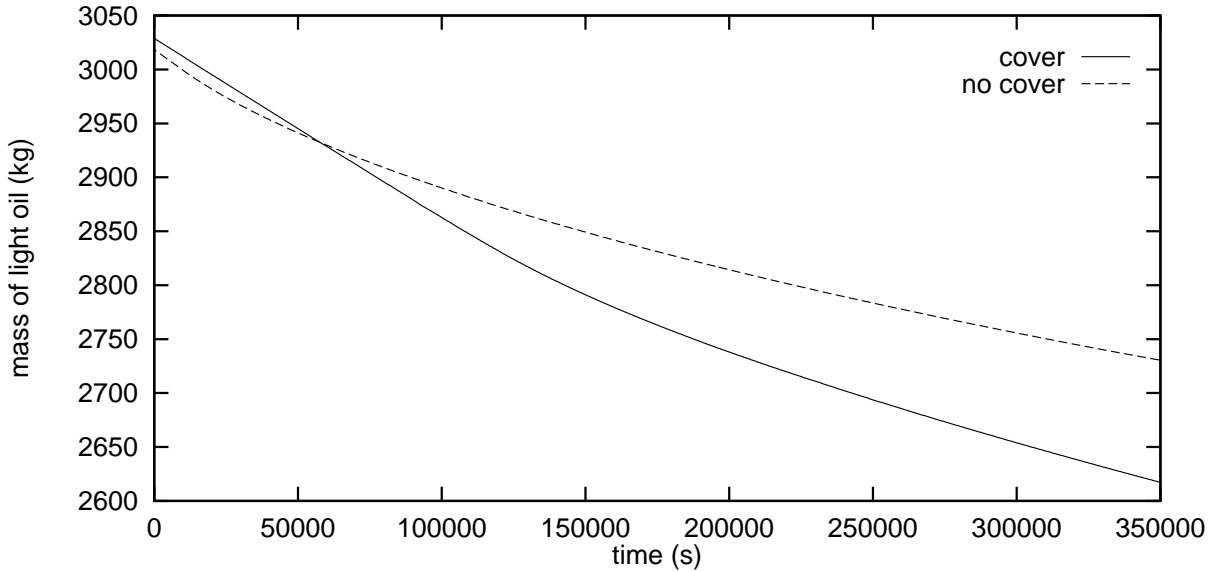


Figure 6: A comparison of the total mass of light oil in the domain against time for the adaptive simulations with and without the cover on the top boundary.

We now consider the efficiency of the adaptive solution. Table 4.2 shows some run statistics for the adaptive and fixed mesh simulations (again for the problem with the cover present). Recall that the same adaptive time-stepping strategy has been used in each case with identical error tolerances and an identical upper limit on the step size. The cpu times quoted are for an SGI O2 workstation with a 180MHz IP32 processor and 256Mb of RAM. It is clear that the number of time steps taken is not particularly affected by the use of adaptivity. For reasons of efficiency we seek to apply the adaptivity algorithm only when a Jacobian evaluation is due to take place anyway (since altering the mesh will clearly force a Jacobian evaluation) and this strategy can be seen to be quite effective. Hence the only statistic that is significantly different between the two runs is the cpu time; the substantial saving offered by the adaptive scheme coming as a result of the smaller linear systems that are present due to there being fewer vertices in the mesh at each time. Of course there is an overhead associated with implementing the adaptive mesh algorithm but this cost is clearly covered by the savings in the time-stepping stages. The smaller linear system associated with the adaptive mesh algorithm also leads to a reduction in memory requirements.

	No. vertices	No. time steps	No. Jac. evals.	Time (cpu secs.)
Adaptive	619-920	1431	86	11414
Fixed	2121	1436	83	23750

Table 1: Run statistics for a typical adaptive and fixed mesh simulation.

### 4.3 Conclusions and future work

The adaptive numerical solution of a particular porous media gas venting problem has been described and the significant computational advantages of using an automatic mesh adaptivity algorithm have been demonstrated. Typical simulations, such as those outlined above, suggest that for the problem sizes encountered here, the total cpu time required by the adaptive solver is approximately proportional to the average number of vertices present in the meshes throughout the simulation. This may

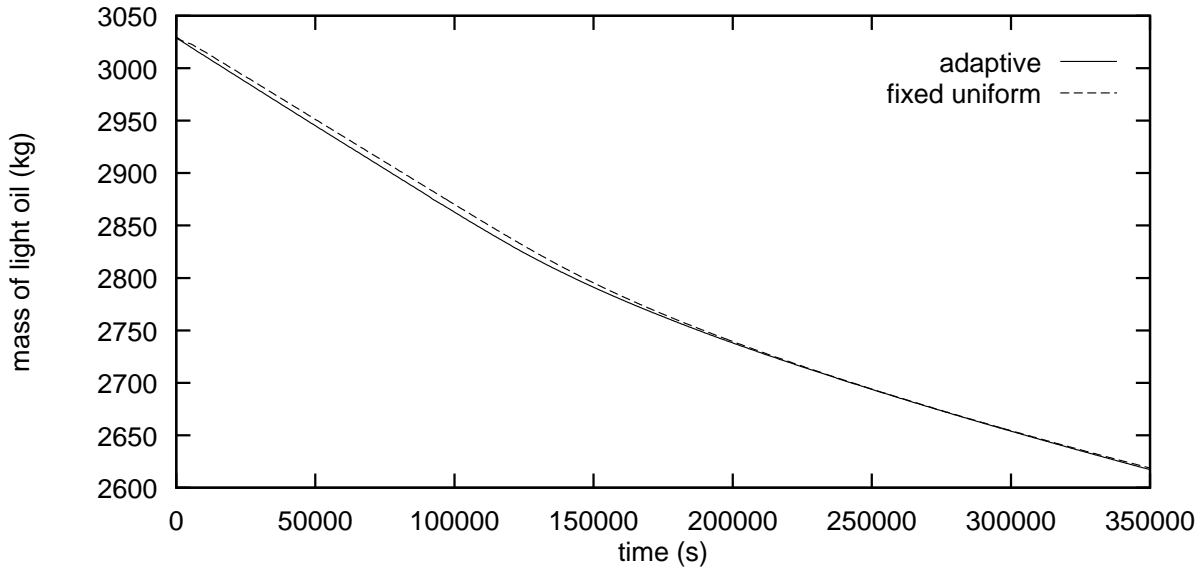


Figure 7: A comparison of the total mass of light oil in the domain against time for the adaptive and the fixed mesh simulations (with the cover on the top boundary).

be explained by noting that although the iterative linear algebra routines used by SPRINT ([5]) have a complexity that is slightly worse than linear, the relative overhead associated with the spatial mesh adaptivity actually decreases as the level of refinement (and therefore the mesh sizes) increases.

There is no reason to believe that similar savings due to the use of mesh adaptivity will not also be achieved when the particular model described in Section 2 is modified. For example, there are clearly situations in which one would also wish to allow mobility of the liquid and/or oleic phases, or the volatility of the liquid water. These possibilities have been excluded from the current model purely to simplify the discretization by ensuring that phase appearance and disappearance is not possible. Nevertheless, the potential advantages of mesh adaptivity are clear in these situations too. A further generalization in 2-d would be to allow more components in both the gas and oleic phases and to include reaction terms in the equations of motion so as to introduce of biodegradation into the model (see for example [10, 2, 11, 30, 3]). Such reaction terms will frequently introduce different time-scales to the problem so the significance of temporal adaptivity is likely to increase even further. In the present model the adaptivity in time is still extremely important however, as illustrated when the vent is switched off for a period of time for example. In this situation the flow becomes diffusion and gravity driven and the size of the time steps can increase by many orders of magnitude.

The final generalization of this work that we mention here is the implementation of a general (i.e. non-axisymmetric) model in three space dimensions. This will allow multiple vents to be considered as well as simple attempts to incorporate physical features of the vadose zone. In three dimensions it is anticipated that mesh adaptivity, based upon the refinement and derefinement of tetrahedra (as in [26, 27] for example), will be of even more benefit than for the cylindrically symmetric problems considered in here.

## References

- [1] Babuska, I., Zienkiewicz, O.C., Gago, J. and Oliviera, E.R. de A. (eds.), *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, Wiley, New York (1986).

- [2] Baehr, A.L. and Corapcioglu, M.Y., A compositional multiphase model for groundwater contamination by petroleum products, 2. Numerical solution, *Water Resources Research*, 23, pp. 201–213 (1987).
- [3] Bedient, P.B., Hanadi, S.R., Newell, C.J., *Ground Water Contamination – Transport and Remediation*, PTR Prentice Hall, New Jersey (1994).
- [4] Berzins, M., Capon, P.J. and Jimack, P.K., On spatial adaptivity and interpolation when using the method of lines, *Appl. Numer. Math.*, 26, pp. 117–133 (1998).
- [5] Berzins, M. and Furzeland, R.M., An adaptive theta method for the solution of stiff and nonstiff differential equations, *Appl. Numer. Math.*, 9, pp. 1–19 (1992).
- [6] Berzins, M., Pennington, S.V., Pratt, P.R. and Ware, J.M., SPRINT2D software for convection dominated PDEs, In *Modern Software Tools in Scientific Computation*, eds. E. Arge, A.M. Bruaset and H.P. Langtangen, Birkhauser, pp. 63–80 (1997).
- [7] Berzins, M., Fairlie, R., Pennington, S.V., Ware, J.M. and Scales, L.E., SPRINT2D: Adaptive software for PDEs, To appear in *ACM Trans. Math. Software* (1998).
- [8] Brusseau, M.L., Transport of organic chemicals by gas advection in structured or heterogeneous porous media: Development of a model and application to column experiments, *Water Resources Research*, 27, pp. 3189–3199 (1991).
- [9] Corapcioglu, M.Y. and Panday, S., Compositional multiphase flow models, in *Advances in Porous Media, Volume 1*, ed. M.Y. Corapcioglu, Elsevier, pp. 1–59 (1991).
- [10] Corapcioglu, M.Y. and Baehr, A.L., A compositional multiphase model for groundwater contamination by petroleum products, 1. Theoretical considerations, *Water Resources Research*, 23, pp. 191–200 (1987).
- [11] Corapcioglu, M.Y. and Hossain, M.A., Theoretical modeling of biodegradation and biotransformation of hydrocarbons in subsurface environments, *J. Theor. Biol.*, 142, pp. 503–516 (1990).
- [12] Cowper, G.R., Gaussian quadrature formulas for triangles, *International Journal of Numerical Methods in Engineering*, 7, pp. 405–410 (1973).
- [13] Forsyth, P.A., Simulation of nonaqueous phase groundwater contamination, *Adv. Water Resources*, 11, pp. 74–83 (1988).
- [14] Forsyth, P.A. and Shao, B.Y., Numerical simulation of gas venting for NAPL site remediation, *Adv. Water Resources*, 14, pp. 354–367 (1991).
- [15] Frind, E.O., Simulation of long-term transient density-dependent transport in groundwater, *Adv. Water Resources*, 5, pp. 73–88 (1982).
- [16] Gamliel, A. and Abdul, A., Numerical investigations of optimal well spacing and the effect of screen length and surface sealing on gas flow toward an extraction well, *J. Contaminant Hydrology*, 12, pp. 171–191 (1993).
- [17] Hansbo, P. and Johnson, C., Adaptive streamline diffusion methods for compressible flow using conservation variables, *Computer Methods in Applied Mechanics and Engineering*, 87, pp. 267–280 (1991).
- [18] Hornung, R.D. and Trangenstein, J.A., Adaptive mesh refinement and multilevel iteration for flow in porous media, *J. Comp. Phys.*, 136, pp. 522–545 (1997).

- [19] Huyakorn, P.S. and Nilkuha, K., Solution of transient transport equation using an upstream finite element scheme, *Appl. Math. Modelling*, 3, pp. 7–17 (1979).
- [20] Lingineni, S. and Dhir, V.K., Modelling of soil venting processes to remediate unsaturated soils, *J. Envir. Engng.*, 118, pp. 135–152 (1992).
- [21] Lingineni, S. and Dhir, V.K., Controlling transport processes during NAPL removal by soil venting, *Adv. Water Resources*, 20, pp. 157–169 (1997).
- [22] Löhner, R., An adaptive finite element scheme for transient problems in CFD, *Comput. Methods Appl. Mech. Engrg.*, 50, pp. 323–338 (1987).
- [23] Numerical Algorithms Group Limited, *The NAG Fortran Library Manual – Mark 18*, NAG, Oxford (1998).
- [24] Shakib, F., Hughes, T.J.R. and Johan, Z., A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations, *Computer Methods in Applied Mechanics and Engineering*, 89, pp. 141–219 (1991).
- [25] Sleep, B.E. and Sykes, J.F., Modeling the transport of volatile organics in variably saturated media, *Water Resources Research*, 25, pp. 81–92 (1989).
- [26] Speares, W. and Berzins, M., A 3-d unstructured mesh adaptation algorithm for time-dependent shock dominated problems, *Int. J. Num. Meth. in Fluids*, 25, 81–104, 1997.
- [27] Touheed, N., Selwood P., Jimack, P.K. and Berzins, M., Parallel dynamic load-balancing for the solution of transient CFD problems using adaptive tetrahedral meshes, in *Parallel Computational Fluid Dynamics – Recent Developments and Advances Using Parallel Computers*, ed. D.R. Emerson *et al.*, Elsevier, pp.81–88 (1998).
- [28] Unger, A.J.A., Sudicky, E.A. and Forsyth, P.A., Mechanisms controlling vacuum extraction coupled with air sparging for remediation of heterogeneous formations contaminated by dense nonaqueous phase liquids, *Water Resources Research*, 31, pp. 1913–1925, 1995.
- [29] Unger, A.J.A., Forsyth, P.A. and Sudicky, E.A., Variable spatial and temporal weighting schemes for use in multi-phase compositional problems, *Adv. Water Resources*, 19, pp. 1–27 (1996).
- [30] Van Eyk, J., *Petroleum Bioventing – An Introduction*, A.A.Balkema, Rotterdam (1997).

## A Summary of notation

$\alpha_g$	Pore-space volume fraction of gas phase
$\alpha_w$	Pore-space volume fraction of water phase
$\alpha_o$	Pore-space volume fraction of oil phase
$\rho_g$	Density of the gas phase
$\rho_w$	Density of the water phase
$\rho_o$	Density of the oil phase
$c_s$	Total mass per unit volume sorbed to soil
$X_g^l$ ( $C_g^l$ )	Mass fraction (mass concentration) of light oil in gas phase
$X_o^l$ ( $C_o^l$ )	Mass fraction (mass concentration) of light oil in oleic phase
$X_w^l$ ( $C_w^l$ )	Mass fraction (mass concentration) of light oil in aqueous phase
$X_s^l$ ( $C_s^l$ )	Mass fraction (mass concentration) of light oil in sorbed phase
$X_g^a$ ( $C_g^a$ )	Mass fraction (mass concentration) of air in gas phase
$X_o^h$ ( $C_o^h$ )	Mass fraction (mass concentration) of heavy oil in oleic phase
$X_w^w$ ( $C_w^w$ )	Mass fraction (mass concentration) of water in aqueous phase
$X_s^h$ ( $C_s^h$ )	Mass fraction (mass concentration) of heavy oil in sorbed phase
$\underline{q}$	Darcy velocity (gas phase)
$P$	Gas phase pressure
$\varepsilon$	Soil porosity
$R$	Universal gas constant
$T$	Temperature
$\lambda^a$	Molecular weight of air
$\lambda^l$	Molecular weight of light oil
$\lambda^h$	Molecular weight of heavy oil
$k$	Absolute permeability
$K_g$	Relative permeability
$\mu$	Dynamic viscosity of gas phase
$P_{sat}^l$	Saturated vapour pressure of light oil
$P_\infty$	Atmospheric pressure
$S_w^h$	Solubility of heavy oil in water
$H_w^l$	Henry's Law constant for light oil in water
$K_D^l$	Soil sorption coeff. for light oil
$K_D^h$	Soil sorption coeff. for heavy oil
$\mathbf{D}_g^l$	Dispersivity tensor for light oil in gas phase
$\mathbf{D}_w^l$	Dispersivity tensor for light oil in oil phase
$\mathbf{D}_o^l$	Dispersivity tensor for light oil in water phase
$\mathbf{D}_g^a$	Dispersivity tensor for air in gas phase
$\mathbf{D}_w^w$	Dispersivity tensor for water in water phase
$\mathbf{D}_o^h$	Dispersivity tensor for heavy oil in oil phase
$T_g$	Tortuosity of gas phase
$T_w$	Tortuosity of water phase
$T_o$	Tortuosity of oil phase
$a_L$	Longitudinal dispersion length
$a_T$	Transverse dispersion length
$e$	Pressure at vent as a proportion of atmospheric pressure
$\rho_b$	Soil bulk density

Table 2: Brief explanation of notation used.

## B Example parameters

The oleic components used in the example results are Toluene for the light oil and n-Dodecane for the heavy oil. The parameter values (in SI units) are:

$$\varepsilon = 0.2$$

$$T = 293.0 \text{ K}$$

$$\lambda^a = 28.9 \times 10^{-3} \text{ kg/mol}$$

$$\lambda^l = 92.2 \times 10^{-3} \text{ kg/mol}$$

$$\lambda^h = 170.34 \times 10^{-3} \text{ kg/mol}$$

$$k = 3.0 \times 10^{-10} \text{ m}^2$$

$$\mu = 18.1 \times 10^{-6} \text{ Pa s}$$

$$P_{sat}^l = 2920.0 \text{ Pa}$$

$$P_\infty = 101.0 \times 10^3 \text{ Pa}$$

$$D_g^l = 8.7 \times 10^{-6} \text{ m}^2/\text{s}$$

$$D_w^l = 8.0 \times 10^{-10} \text{ m}^2/\text{s}$$

$$D_o^l = 8.0 \times 10^{-10} \text{ m}^2/\text{s}$$

$$S_w^h = 3.7 \times 10^{-6} \text{ kg/m}^3$$

$$H_w^l = 522.76 \text{ m}^3 \text{ Pa/mol}$$

$$K_D^l = 0.234 \text{ m}^3/\text{kg}$$

$$K_D^h = 1.26 \times 10^3 \text{ m}^3/\text{kg}$$

$$\rho_o = 749.0 \text{ kg/m}^3$$

$$\rho_w = 998.0 \text{ kg/m}^3$$

$$\rho_b = 2.08 \text{ kg/m}^3$$

$$R = 8.32 \text{ J/mol/C}$$

$$\alpha_L = 0.5 \times 10^{-3} \text{ m}$$

$$\alpha_T = 0.25 \times 10^{-4} \text{ m}$$

$$n = 8.5$$

$$X_h = 0.9$$

$$P_v = 99.0 \times 10^3 \text{ Pa}$$

The parameters for the spatial domain are:

$$Z_L = 0.75 \text{ m}$$

$$Z_U = 1.5 \text{ m}$$

$$H = 3 \text{ m}$$

$$R_V = 0.3 \text{ m}$$

$$R_C = 4.0 \text{ m}$$

$$R_D = 15.0 \text{ m}$$