INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 10, 227–246 (1990)

SELF-ADAPTIVE HIERARCHIC FINITE ELEMENT SOLUTION OF THE ONE-DIMENSIONAL UNSATURATED FLOW EQUATION

LINDA M. ABRIOLA AND JOHN R. LANG

Department of Civil Engineering, The University of Michigan, Ann Arbor, MI, U.S.A.

SUMMARY

The advantages associated with the use of self-adaptive methods for the solution of problems which require the prediction of a frontal position in time are well known. In this paper a self-adaptive finite element solution for the non-linear unsaturated flow equation is developed using hierarchic *p*-version enrichment of the interpolating space. Additional computational advantages are demonstrated for an iteration scheme in which iterations after enrichment are performed only over a subdomain. Numerical solutions are presented for a one-dimensional infiltration scenario.

KEY WORDS Adaptive Finite element method *p*-version Unsaturated flow Hierarchic Non-linear

INTRODUCTION AND BACKGROUND

Many problems of environmental significance require the prediction of the displacement of a front with time. When solving such problems numerically, methods which provide greater resolution at the front where the state variable is changing most rapidly are computationally desirable. These models are commonly described as 'self-adaptive' in the literature. By enriching the approximation, based on feedback from previous solutions, self-adaptive methods can produce improved solutions, both in terms of efficiency and reliability.[1]

In this paper a self-adaptive finite element method is presented for the numerical solution of the non-linear Fokker–Planck equation for the transient isothermal flow of water into a non-swelling unsaturated soil.[2] The solution is obtained in one-dimension, with initial and boundary conditions and functional forms for the non-linear terms after Haverkamp *et al.*[3]

In the presented application the saturation front is moving vertically with time, making a self-adaptive method advantageous computationally by allowing greater resolution at the front for a given number of degrees of freedom. Many numerical models have been developed for the solution of problems in unsaturated flow. A number are reviewed by van Genuchten.[4] These models include solutions based on a dynamic simulation language,[5] finite differences[3,6–9] (including a two-phase numerical model which allows the air pressure to vary[10]), finite element approximations[11–13] and a multigrid finite difference solution[14] for the steady state problem. None of the numerical models mentioned above is self-adaptive. An adaptive model developed for unsaturated flow using a Eulerian–Langrangian finite element approach is discussed by Sorek and Braester,[15] although no numerical results are presented.

0271–2091/90/030227–20$10.00

Received 16 November 1988
Revised 25 April 1989

The self-adaptive algorithm described herein is developed using hierarchic basis functions. In this algorithm the local accuracy of the solution is improved by increasing the order of the basis functions at the front. Such techniques are commonly known as *p*-version finite element methods. The model presented in this paper is an extension and modification of preliminary work presented by the first author.[16] Self-adaptive hierarchic *p*-version finite element methods have been shown to possess several advantages over other self-adaptive finite element methods of the so-called '*h*' or '*r*' variety.[17–19] These advantages are discussed in greater detail later in this paper. The discussion includes the results of numerical simulations which support the selection of hierarchic *p*-version enrichment for this application. *h*-Version methods refine the solution by introducing additional nodes to an interpolation space of fixed order, while *r*-version methods refine the solution by redistributing a fixed number of nodes in space. Both methods have been widely used.[20–25] Hierarchic basis functions have the property that the introduction of additional degrees of freedom simply adds additional terms to the approximating functions while retaining the terms computed previously. This results in computational efficiency owing to the retention of previously computed terms and to the suitability of the final matrix structure for the use of various efficient solution schemes.[26–28] The hierarchic *p*-version finite element method has been used extensively to solve linear fracture mechanics and elasticity problems.[29–31] Applications to linear elliptic and hyperbolic problems,[32] the generalized linear eigenvalue problem,[33] heat conduction[20] and flow problems[34] have also been reported. While the effectiveness of the hierarchic finite element approach has been extensively documented for linear problems, little work has been done to examine its utility for non-linear problems. Babuska and Rheinbolt investigated some non-linear structural problems[35] and found that, for a given number of degrees of freedom, higher-order elements performed better than lower-order elements. This paper represents a relatively unique application of the self-adaptive *p*-version finite element method to a non-linear transient problem of environmental significance.

MATHEMATICAL MODELLING

Governing equation

The governing equation for one-dimensional flow of water in an unsaturated vertical column is given by

$$C\frac{\partial h}{\partial t} = \frac{\partial}{\partial z}\left[K k_{\mathrm{rw}}\left(\frac{\partial h}{\partial z} - 1\right)\right], \tag{1}$$

where $C = n\,\mathrm{d}s/\mathrm{d}h$ is the soil moisture capacity, n is the matrix porosity, h is the suction head, s is the water saturation, z is the vertical co-ordinate (directed downwards), t is the time co-ordinate, K is the saturated hydraulic conductivity and k_{rw} is the relative permeability of the water phase. This equation neglects the effects of matrix and fluid compressibility and assumes that the air phase is static. Since the soil moisture capacity C and the relative permeability k_{rw} are functions of h, equation (1) is a non-linear partial differential equation in the suction head h.

Application of the Galerkin method

In order to solve equation (1) numerically, Galerkin's method is applied to the governing equation. First, the continuous function h is replaced by the trial function $\hat{h}$:

$$h(z, t) \approx \hat{h}(z, t) = \sum_{j=1}^{N} w_j(z) h_j(t), \tag{2}$$

where w_j are members of a family of hierarchic basis functions which will be discussed in greater detail in a following section, $h_j(t)$ are undetermined nodal coefficients and N is the number of nodes. Since k_{rw} and ds/dh are functions of h, they can also be expanded in terms of the basis functions $w_k(z)$ as

$$k_{rw}(z, t) \approx \hat{k}_{rw}(z, t) = \sum_{k=1}^{N} w_k(z) k_{rwk}(t), \tag{3}$$

$$\frac{ds}{dh}(z, t) \approx \frac{d\hat{s}}{dh}(x, t) = \sum_{k=1}^{N} w_k(z) \frac{ds}{dh_k}(t). \tag{4}$$

The two parameters K and n are assumed piecewise constant over each element. After substituting equations (2), (3) and (4) into the governing equation (1), Galerkin's method of weighted residuals is employed. Application of Green's theorem to the gradient term yields a matrix equation of the form

$$[\mathbf{A}]\{\mathbf{h}\} + [\mathbf{B}]\{d\mathbf{h}/dt\} = \{\mathbf{f}\}, \tag{5}$$

where the representative matrix and vector elements are given by

$$A_{ij} = \sum_{e=i}^{i+1} \sum_{k=e}^{e+1} \int_{\Omega^e} K^e k_{rwk} w_k \frac{dw_j}{dz} \frac{dw_i}{dz} d\Omega, \tag{6a}$$

$$B_{ij} = \sum_{e=i}^{i+1} \sum_{k=e}^{e+1} \int_{\Omega^e} n^e \frac{ds}{dh_k} w_k w_j w_i d\Omega, \tag{6b}$$

$$f_i = \sum_{e=i}^{i+1} \left(\sum_{k=e}^{e+1} \int_{\Omega^e} K^e k_{rwk} w_k \frac{dw_i}{dz} d\Omega - \int_{\Gamma^e} w_i q d\Gamma \right), \tag{6c}$$

where $i, j, k \in [1, N]$; $e \in [1, N-1]$. In equations (6a–c) the global domain has been subdivided into elements Ω^e. Γ^e is the element boundary. Now a backward difference formula is used to approximate the time derivative in equation (5), resulting in the following matrix equation:

$$[\mathbf{A}]^{t+\Delta t}\{\mathbf{h}\}^{t+\Delta t} + [\mathbf{B}]^{t+\Delta t}(\{\mathbf{h}\}^{t+\Delta t} - \{\mathbf{h}\}^{t})/\Delta t = \{\mathbf{f}\}^{t+\Delta t}. \tag{7}$$

The superscripts in equation (7) refer to the time level of evaluation of each term. Equation (7) is fully implicit and can be rearranged to yield

$$[\mathbf{M}]^{t+\Delta t}\{\mathbf{h}\}^{t+\Delta t} = \{\mathbf{r}\}^{t+\Delta t}, \tag{8}$$

where

$$[\mathbf{M}]^{t+\Delta t} = [\mathbf{A}]^{t+\Delta t} + [\mathbf{B}]^{t+\Delta t}/\Delta t, \tag{9a}$$

$$[\mathbf{r}]^{t+\Delta t} = [\mathbf{f}]^{t+\Delta t} + [\mathbf{B}]^{t+\Delta t}\{\mathbf{h}\}^{t}/\Delta t. \tag{9b}$$

Selection of basis functions

The basis functions were selected to be members of a family of C^0-continuity hierarchic Lagrangian basis functions.[36] The work presented herein employs only the linear and quadratic members of the hierarchic basis function set ($p \leqslant 2$). The linear functions are the familiar chapeau functions which can be written in local co-ordinates $-1 \leqslant \xi \leqslant 1$ as

$$\omega^{-1} = 0{\cdot}5(1 - \xi) \tag{10a}$$

$$\omega^{1} = 0{\cdot}5(1 + \xi). \tag{10b}$$

The quadratic basis function is parabolic with a zero value at the element nodes and a second derivative of one at the element midpoint:

$$\omega^0 = 0{\cdot}5(\xi^2 - 1). \tag{10c}$$

By using the three basis functions, the trial function $\hat{h}$ can be written for a given element as

$$\hat{h}(z, t) = h_{-1}(t)\omega^{-1} + h_1(t)\omega^1 + h_0''(t)\omega^0. \tag{11}$$

In equation (11) h_0'' is the second derivative of h with respect to the local spatial co-ordinate evaluated at the element midpoint. Thus the three degrees of freedom for a given quadratic element using these basis functions are the two nodal variables h_{-1} and h_1 and the so-called 'nodeless' variable h_0''.

This particular choice of basis functions possesses an important property. The quadratic trial function is formed by adding the product of h_0'' and the quadratic basis function to the linear trial function. If the global finite element matrix is formed using linear basis functions, selective enrichment of the solution is achieved by simply adding to the global matrix appropriate rows and columns containing the new terms resulting from the addition of quadratic terms to the trial function. Therefore previously formed matrices using lower-order basis functions are imbedded in matrices formed with higher-order basis functions. Thus the computational effort spent in forming the original global matrix with linear basis functions is retained. In addition, the initial solution for h, obtained from iterations over the matrix equation using entirely linear basis functions, can serve directly as a first estimate for iterations over the enriched mesh without interpolation.

Matrix structure

If the nodeless variables are ordered so that they follow all the nodal values of h, the matrix equation (8) can be rewritten after enrichment with quadratic basis functions as

$$\left[\begin{array}{c|c} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \hline \mathbf{M}_{21} & \mathbf{M}_{22} \end{array}\right] \left\{\begin{array}{c} \mathbf{h} \\ \hline \mathbf{h}'' \end{array}\right\} = \left\{\begin{array}{c} \mathbf{r}_1 \\ \hline \mathbf{r}_2 \end{array}\right\}. \tag{12}$$

In equation (12) the matrix $\mathbf{M}_{11}$ is a tridiagonal matrix composed of the same terms as the equivalent Galerkin formulation using only linear basis functions. To formulate $\mathbf{M}_{12}$ and $\mathbf{M}_{21}$, the corresponding matrices **A** and **B** need to be examined. In the following development the non-linear coefficients ds/dh and k_{rw} are expanded across an element only in terms of ω^{-1} and ω^1. This was done to avoid reformulation of $\mathbf{M}_{11}$ after enrichment and to maintain the linearity of equation (16a). Since ω^0 is zero at the element boundaries, there is no connectivity between elements in $\mathbf{M}_{12}$ and $\mathbf{M}_{21}$. Therefore $\mathbf{M}_{12}$ will have only diagonal and subdiagonal terms and $\mathbf{M}_{21}$ will have only diagonal and superdiagonal terms. Typical forms for the diagonal and off-diagonal terms of **A** and **B** in $\mathbf{M}_{12}$ are given by

$$A_{i,i} = \sum_{k=\pm 1} \int_{\Omega^e} K^e k_{rwk} \omega^k \frac{d\omega^0}{dz}\frac{d\omega^{-1}}{dz} d\Omega, \tag{13a}$$

$$A_{i,i-1} = \sum_{k=\pm 1} \int_{\Omega^e} K^e k_{rwk} \omega^k \frac{d\omega^0}{dz}\frac{d\omega^{1}}{dz} d\Omega, \tag{13b}$$

$$B_{i,i} = \sum_{k=\pm 1} \int_{\Omega^e} n^e \frac{ds}{dh_k} \omega^k \omega^0 \omega^{-1} d\Omega, \tag{13c}$$

$$B_{i,i-1} = \sum_{k=\pm 1} \int_{\Omega^e} n^e \frac{ds}{dh_k} \omega^k \omega^0 \omega^{1} d\Omega. \tag{13d}$$

In equations (13b, d) integration is over element $i-1$, while in equations (13a, c) integration is over element i. For $\mathbf{M}_{21}$,

$$A_{i,i} = \sum_{k=\pm 1} \int_{\Omega^e} K^e k_{\mathrm{rw}k} \omega^k \frac{\mathrm{d}\omega^{-1}}{\mathrm{d}z} \frac{\mathrm{d}\omega^0}{\mathrm{d}z} \mathrm{d}\Omega, \tag{13e}$$

$$A_{i,i+1} = \sum_{k=\pm 1} \int_{\Omega^e} K^e k_{\mathrm{rw}k} \omega^k \frac{\mathrm{d}\omega^1}{\mathrm{d}z} \frac{\mathrm{d}\omega^0}{\mathrm{d}z} \mathrm{d}\Omega, \tag{13f}$$

$$B_{i,i} = \sum_{k=\pm 1} \int_{\Omega^e} n^e \frac{\mathrm{d}s}{\mathrm{d}h_k} \omega^k \omega^{-1} \omega^0 \mathrm{d}\Omega, \tag{13g}$$

$$B_{i,i+1} = \sum_{k=\pm 1} \int_{\Omega^e} n^e \frac{\mathrm{d}s}{\mathrm{d}h_k} \omega^k \omega^1 \omega^0 \mathrm{d}\Omega. \tag{13h}$$

In equations (13e–h) integration is over element i. $\mathbf{M}_{12}$ and $\mathbf{M}_{21}$ can be formed from equations (13a–h) using (9a). $\mathbf{M}_{22}$ can likewise be formed using equation (9a) and the following expressions for $\mathbf{A}$ and $\mathbf{B}$:

$$A_{i,i} = \sum_{k=\pm 1} \int_{\Omega^e} K^e k_{\mathrm{rw}k} \omega^k \frac{\mathrm{d}\omega^0}{\mathrm{d}z} \frac{\mathrm{d}\omega^0}{\mathrm{d}z} \mathrm{d}\Omega, \tag{13i}$$

$$B_{i,i} = \sum_{k=\pm 1} \int_{\Omega^e} n^e \frac{\mathrm{d}s}{\mathrm{d}h_k} \omega^k \omega^0 \omega^0 \mathrm{d}\Omega. \tag{13j}$$

In equations (13i, j) integration is over element i. In a similar fashion $\mathbf{r}_1$ and $\mathbf{r}_2$ are formed:

$$\{\mathbf{r}_1\}^{t+\Delta t} = \{\mathbf{f}\}^{t+\Delta t} + ([\mathbf{B}]^{t+\Delta t}\{\mathbf{h}\}^t + [\mathbf{B}_1]^{t+\Delta t}\{\mathbf{h}''\}^t)/\Delta t, \tag{14a}$$

$$\{\mathbf{r}_2\}^{t+\Delta t} = \{\mathbf{f}\}^{t+\Delta t} + ([\mathbf{B}_2]^{t+\Delta t}\{\mathbf{h}\}^t + [\mathbf{B}_3]^{t+\Delta t}\{\mathbf{h}''\}^t)/\Delta t. \tag{14b}$$

Here $\mathbf{B}$ refers to equation (6b), $\mathbf{B}_1$ to equations (13c, d), $\mathbf{B}_2$ to equations (13g, h) and $\mathbf{B}_3$ to equation (13j).

In this work equations (12) are solved sequentially at each time step using the Thomas algorithm and Newton–Raphson iterations in a scheme patterned after the block Gauss–Seidel iterations used by Peano *et al.*[26] The actual solution is in incremental form to minimize truncation error. First the subproblem

$$[\mathbf{M}_{11}]\{\mathbf{h}\} = \{\mathbf{r}\} \tag{15}$$

is solved using the Thomas algorithm and a modified Newton–Raphson method. Hereafter iterations using equation (15) will be referred to as type A iterations. In the solution procedure the Jacobian is updated after each type A iteration. The converged solution is then used as the first estimate for the following two-step procedure:

$$[\mathbf{M}_{22}]\{\mathbf{h}''\}^{\eta+1} = \{\mathbf{r}_2\} - [\mathbf{M}_{21}]\{\mathbf{h}\}^{\eta}, \tag{16a}$$

$$[\mathbf{M}_{11}]\{\mathbf{h}\}^{\eta+1} = \{\mathbf{r}_1\} - [\mathbf{M}_{12}]\{\mathbf{h}''\}^{\eta+1}. \tag{16b}$$

Here the superscripts η and $\eta+1$ indicate the iteration level. Since $\mathbf{M}_{22}$ is diagonal, equation (16a) can be solved directly for h''. Equation (16b) is solved in the same fashion as was equation (15). These iterations are hereafter termed type B iterations. The Jacobian for type B iterations is updated only on the first iteration. After each type B iteration the matrices are updated (except for the Jacobian) and steps (16a) and (16b) are repeated until a convergence criterion is satisfied. It should be noted here that other solution schemes are possible and may be more effective, especially when solutions are sought for problems with two or three dimensions. In their

examination of the hierarchic solution of linear convection-dominated flow, Wiberg and Möller[32] used several conjugate gradient methods. Other authors have noted that the use of hierarchic basis functions in conjunction with the multigrid algorithm may also be very efficient.[27] This last approach will be discussed in a later section of this paper.

COMPUTATIONAL CONCEPTS

Three features relating to the performance of the computational scheme need to be presented. These are: (1) the procedure used to increase the time step in an attempt to optimize the balance between solution accuracy and computational effort; (2) the convergence criteria used to determine when the error in the solution of the non-linear problem at a given time step has been sufficiently reduced by the Newton–Raphson iterations; and (3) the procedure used to select elements where quadratic terms are to be added self-adaptively.

In this work the time step size is controlled within the code by the number of type A iterations required to achieve convergence for the subproblem (15). If the solution requires less than four type A iterations at the previous time level, the time step is multiplied by 1·5 to obtain the new time step size. The number of type B iterations does not influence the time step size. A small time step was used at the start of most simulations to account for the discontinuous initial conditions. For all the simulations contained herein, the initial time step was $5{\cdot}0 \times 10^{-5}$ h. It should be noted that no significant difference in solution quality was observed between a solution computed using the 'natural' time step and a solution computed using a time step constrained to a reasonably smaller value, as long as the grid spacing was not extremely coarse. This was true for both the linear and self-adaptive elements. Solutions computed using coarser discretizations generally used larger time steps.

The convergence criterion used for all simulations presented in this paper is

$$\| \mathbf{h}^{\eta+1} - \mathbf{h}^{\eta} \|_{\infty} / \| \mathbf{h}^{\eta+1} \|_{\infty} \leqslant e_{\mathrm{r}}, \tag{17}$$

where the superscript on $\mathbf{h}$ indicates the iteration level during a given time step. Since the differences are computed by the program directly, this approach is relatively efficient. For all the simulations in this paper a value of 10^{-4} was used for e_{r}. Solution quality deteriorated when larger values of e_{r} were used, and computational effort increased unnecessarily when smaller values were employed. This criterion was used to determine convergence of both type A and B iterations. The code that generated the simulations presented in this paper has an option to use residuals as an error criterion, but this option was not employed for any of the simulations presented in this paper. Convergence criteria based on some estimator of local error[17,18,21,35,37] are also possible, but these are generally more expensive to compute and have not been used in this work.

Quadratics are introduced within an element if the change in the saturation per unit length in the ith element exceeds the criterion e_{q}:

$$|s_{i+1} - s_i| / \Delta z \leqslant e_{\mathrm{q}}. \tag{18}$$

Here the subscript on s indicates the node number. This criterion is determined using the saturations from the type A iterations. Note that the saturation can be viewed as a non-linear normalization of the suction head to the range 0·0–1·0. Therefore e_{q} is a criterion based upon the maximum permissible gradient of a non-linearly normalized variable. Other criteria are presently being explored in conjunction with the extension of this method to two dimensions. The selection of the value of e_{q} will be discussed later since it is dependent upon the given discretization and type of adaptive method used for the solution of a particular problem.

EXAMPLE NUMERICAL SIMULATION

The finite element model developed above was used to simulate water infiltration into a column of homogeneous sand. The following expressions were employed for the saturation and the relative permeability:

$$k_{rw} = A/(A + |h|^B), \tag{19}$$

$$s = a(s_s - s_r)/(a + |h|^b) + s_r. \tag{20}$$

Here A, B, a and b are empirical constants and s_s and s_r are the saturated and residual water levels in the medium. Table I contains the values of the various constants and numerical criteria as well as the boundary and initial conditions used for the numerical simulations presented in this paper.

The initial simulations modelled a 50 cm sand column and saturation profiles were generated 0·1 h after infiltration had begun (near-field problem). This allowed comparison with Philip's quasi-analytical solution computed by Haverkamp *et al.*[3] Figure 1 shows the improvement in the resolution of the front resulting from the use of the self-adaptive hierarchic elements. Both the self-adaptive hierarchic solution, hereafter known as the self-adaptive solution, and the strictly linear element solution, hereafter known as the linear solution, were computed using 15 elements ($\Delta z = 4{\cdot}0$ cm). The self-adaptive solution was quadratic in elements 3, 4 and 5 on the front. There is a considerable improvement in the solution at the front after the quadratic elements are introduced. This improvement required only three additional degrees of freedom. In order to validate the code, a numerical solution was computed using 100 linear elements ($\Delta z = 0{\cdot}5$ cm). For this mesh, both the self-adaptive and linear solutions closely matched the quasi-analytical solution.

In order to determine the potential savings in computational effort resulting from the use of the self-adaptive scheme, a somewhat different problem was considered. A column 500 cm long was modelled with the final saturation profiles generated after an elapsed simulation time of 3·0 h (far-field problem). For most simulations the time step size was not changing after 1·0 simulated hour. The use of the far-field problem reduced the relative importance of the initial start-up period of the computation (application of discontinuous initial conditions) upon the total computational

Table I. Parameters for example simulations

n	0·30
K	34·0 cm h^{-1}
s_s	0·95
s_r	0·250
A	1·175 × 10^6
B	4·74
a	1·611 × 10^6
b	3·96

Boundary and initial conditions

$s_w = 0{\cdot}333$ at $t = 0$	$0 < z < 50$ cm	(1)
	$0 < z < 500$ cm	(2)
$s_w = 0{\cdot}890$ at $z = 0$	$t \geqslant 0$	(1, 2)
$s_w = 0{\cdot}333$ at $z = 50$ cm	$t \geqslant 0$	(1)
$z = 500$ cm	$t \geqslant 0$	(2)

(1) Near-field problem.
(2) Far-field problem.

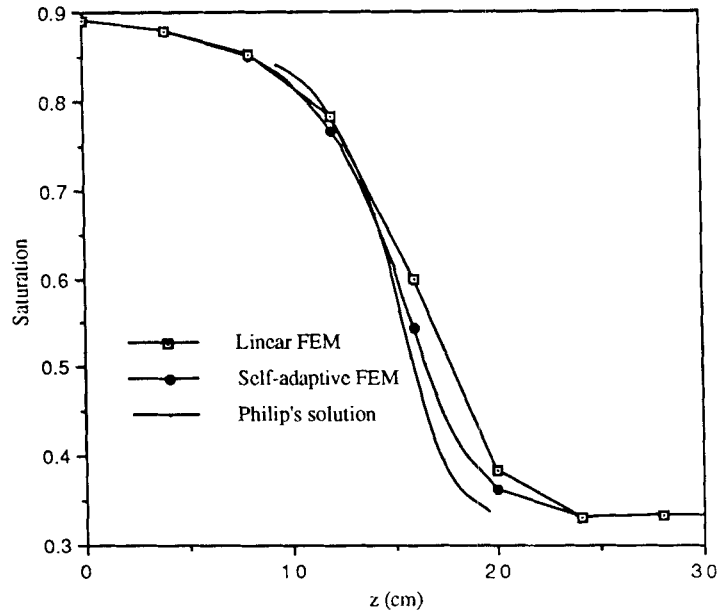

Figure 1. Near-field comparison of self-adaptive and linear FEM; $\Delta z = 4{\cdot}0$ cm for both FEM solutions

effort. This permitted a comparison of computational efficiency between different finite element schemes at the maximum time step allowed by the arbitrary technique used to control the time step size.

As a measure of the relative accuracies of different finite element solutions, two approximate error norms were computed. The L_∞ and the discrete L_2 norms were determined using a fine mesh numerical approximation of the true solution (1000 linear elements, $\Delta z = 0{\cdot}5$ cm). A solution computed using this mesh size compared very well with Philip's quasi-analytical solution for the near-field problem. Since the magnitude of the discrete L_2 norm depends on the number of nodes as well as the quality of the solution, it serves best as a measure of the relative qualities of solutions computed at a given spatial discretization. The L_∞ norm is not as sensitive to the nodal number, although it is somewhat sensitive to the location of the nodes relative to the front. It can be used as a measure of relative solution quality at different spatial discretizations. It should be noted that for both error norms excessively coarse grids can lead to misleading numbers since the front may be spanned by only one or two elements. The use of continuous norms would avoid these problems but would require a much greater computational effort.

Figure 2 shows a comparison of several finite element solutions of the far-field problem. The plain line indicates a finite element solution computed using 1000 linear finite elements. The 100-element self-adaptive solution had 104 degrees of freedom or three added quadratic elements. The coarse linear solution was computed with 250 elements (251 degrees of freedom). Table II lists both the L_2 and L_∞ error norms associated with each solution as well as the computational effort each solution required. A 500-element linear solution (501 degrees of freedom) and a 250-element self-adaptive solution (259 degrees of freedom or eight quadratic elements) are also included in Table II to facilitate computational efficiency comparisons. These additional solutions were very similar to the 1000-element linear solution as indicated by the small error norms and were not plotted since the resolution of the plot was not adequate to show the differences between the two

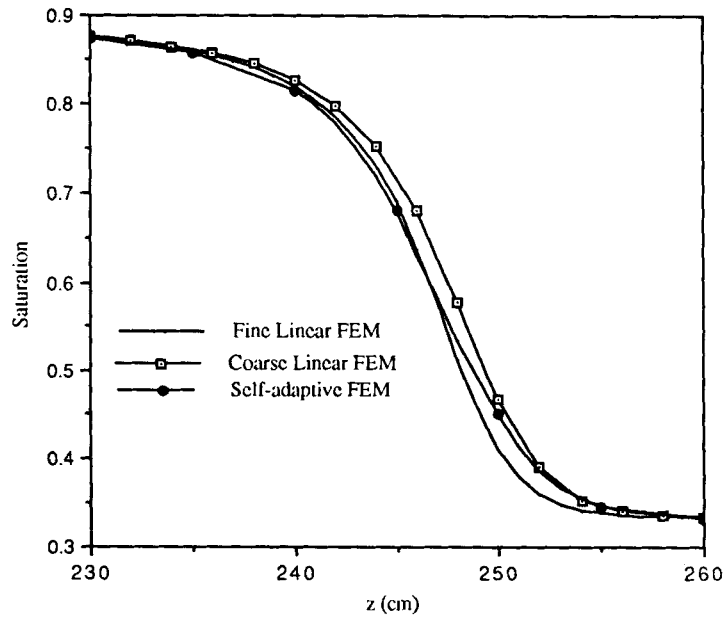

Figure 2. Far-field comparison of self-adaptive and linear FEM; $\Delta z = 2{\cdot}0$ cm for the coarse linear FEM; $\Delta z = 5{\cdot}0$ cm for the self-adaptive FEM

Table II. Far-field comparison

Number (type) of elements	Error norms		CPU time (s)
	L_2	L_∞	
1000 (linear)	—	—	2184
500 (linear)	2·50	1·08	1044
333 (linear)	4·94	2·57	716
250 (linear)	8·42	5·10	464
250 (self-adaptive)	1·58	1·02	947
100 (self-adaptive)	4·06	3·64	316
50 (self-adaptive)	5·12	3·71	77

solutions. Plots examined at a higher resolution indicated that the two additional solutions behaved in the same fashion as did the solutions presented in Figure 2.

Computational effort is reported in Table II as the number of CPU seconds required to execute the program. A Sun 3/160 workstation was used for all simulations. The self-adaptive finite element scheme using 100 elements required approximately 30% less CPU time than the 250-element linear solution. On the basis of the L_∞ norms the self-adaptive solution was also about 26% superior to the linear solution even though the grid spacing of the self-adaptive solution was much coarser and the execution time was less. An examination of Figure 2 indicates that the self-adaptive solution also located the front more accurately than did the linear solution. Each solution used the same maximum time step size of 0·0146 s so that the time step size did not affect the comparison between the two solutions. These same qualitative observations also apply to a comparison of the 500-element linear solution and the 250-element self-adaptive solution. The

250-element self-adaptive solution generated a solution of accuracy comparable to the 500-element linear solution at a CPU time saving of approximately 10%.

Both the 1000-element linear and the two self-adaptive solutions predict the first half of the front identically. The error associated with the self-adaptive solutions occurs primarily at the tail of the front. This observed behaviour leads to the conclusion that the self-adaptive solutions are better predictors for the front's location than are the linear solutions, although both solutions may have similar maximum error norms. As the grid coarsens the linear solutions have a tendency to move to the right, overestimating the extent of the front's movement (Figure 3). Alternatively, the self-adaptive solutions exhibit increasing amounts of numerical dispersion but continue to predict the same location for the front (Figure 4) regardless of grid density.

DISCUSSION

There are two major issues to be examined in the application of a particular numerical scheme: accuracy and efficiency. In the previous section is was demonstrated that the self-adaptive approach used in this work could produce solutions which closely approximated Philip's quasi-analytical solution. In this section some comparisons are made with other self-adaptive schemes, and the effects of time step size, enrichment criteria and convergence criteria on solution quality and efficiency are discussed. Alternative methods to speed convergence are also explored.

In order to make a complete assessment of computational efficiency and accuracy, other self-adaptive schemes were considered. The self-adaptive finite element method presented in this paper is currently being extended to two-dimensional saturated–unsaturated flow. Eventually, the two-dimensional simulator of saturated–unsaturated flow will be coupled to a multi-component transport simulator. Since it is not realistic to assume that the zones requiring

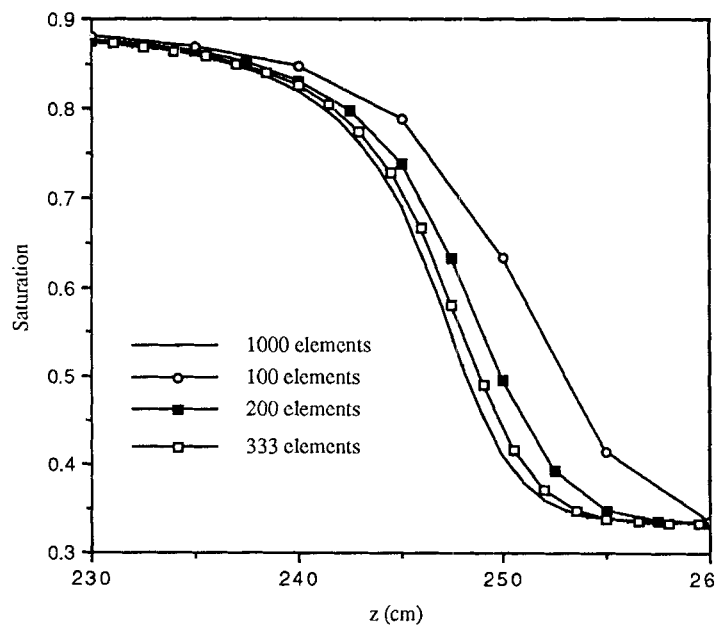

Figure 3. Far-field solution with linear elements; $\Delta z = 5{\cdot}0$ cm with 100 elements; $\Delta z = 2{\cdot}5$ cm with 200 elements; $\Delta z = 1{\cdot}5$ cm with 333 elements

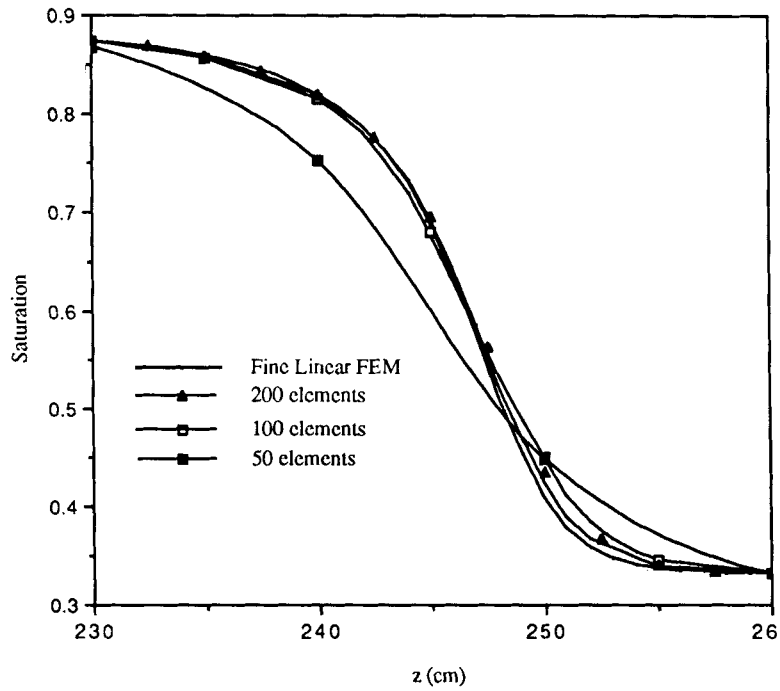

Figure 4. Far-field solution with self-adaptive elements; $\Delta z = 10{\cdot}0$ cm with 50 elements; $\Delta z = 5{\cdot}0$ cm with 100 elements; $\Delta z = 2{\cdot}5$ cm with 200 elements

refinement will coincide for the coupled flow and transport equations, the use of *r*-version enrichment would likely result in very different discretizations for each of the coupled problems. This would require the use of complicated interpolation operators to pass required nodal values between the various coupled simulators. Additionally, the implementation of a truly two-dimensional *r*-version finite element simulator is not straightforward and would likely be far more complicated than the use of *p*-version enrichment. Therefore the use of *r*-version adaptation was not considered to be appropriate for this application.

The use of *h*-version adaptation was also considered. It has been noted by other authors that, for problems in linear fracture mechanics and elasticity, *p*-version enrichment required fewer added degrees of freedom than *h*-version refinement to achieve comparable accuracy.[17,30] In two dimensions they observed that the *p*-version required one-fifth to one-tenth the number of degrees of freedom required by the *h*-version. This behaviour is also observed in the results presented in Table II and discussed in the previous section. For solutions of comparable accuracy, the self-adaptive elements were less than one-half the size of the non-self-adaptive elements. The results summarized in Table II demonstrate that degrees of freedom added hierarchically via a higher-order basis function are more effective at reducing error than the same number of degrees of freedom added by reducing the element dimensions.

In order to compare the relative computational efficiencies of the *h*- and *p*-versions, self-adaptive numerical simulations were performed for the far-field problem using both methods. A self-adaptive *h*-version simulation with $e_q = 10^{-3}$ and an initial grid of 50 elements ($\Delta z = 10$ cm) required approximately the same CPU time as a *p*-version simulation with the same parameters. The accuracy of the *p*-version was markedly superior. Discrete error norms from the *h*-version simulation were over 50% larger than the norms from the *p*-version simulation. The self-adaptive

h-version method also overpredicted the front's position as was noted for the non-self-adaptive simulations discussed earlier. One important advantage of self-adaptive methods is to permit the use of relatively coarse grids while generating solutions of acceptable accuracy. The numerical comparisons presented above suggest that for relatively coarse discretizations the self-adaptive p-version method is superior to the self-adaptive h-version method.

Another self-adaptive scheme could be developed based upon the combined use of standard linear and quadratic (three-node) elements. The performance of the standard quadratic elements would most likely be similar to that of the hierarchic quadratic elements. An application of this sort would require the complete reformulation of the coefficient matrices for any enriched elements in addition to an increase in the dimensions of the stored matrix system. On the basis of the above considerations there is no reason to select such a scheme from the standpoint of computational efficiency or accuracy. The remainder of this discussion will be devoted to the p-version scheme previously described.

The effect of the number of quadratic elements used at the front upon the solution quality and efficiency was examined. Quadratic terms were added to a particular element when the magnitude of $\Delta s/\Delta z$ for the given element exceeded an arbitrary criterion. For most of the numerical experiments presented herein, 10^{-2} was used. Figure 5 shows the results of decreasing this criterion for the far-field problem using a discretization of 50 elements. A value of the criterion less than 10^{-4} did not improve the solution and required additional computational effort. Using the optimized criterion of 10^{-4} yielded an improved solution but did not affect the number of type B iterations required for a convergent solution. This criterion is somewhat dependent upon the discretization of the domain since it has been observed that for finer discretizations a slightly larger criterion gave optimum results. Note that the use of this criterion does not result in the addition of quadratic elements ahead of the front (i.e. in the direction of front movement). This is a consequence of the uniform initial conditions imposed upon the domain. The slight numerical dispersion observed at the front's tail (see Figure 5) may be a result of the limited number of quadratic elements in that zone of the front. Another mesh enrichment criterion may be needed in order to obtain optimal results. An enrichment criterion based on the curvature of an element may be appropriate. Other potential criteria may be developed from various local error estimators.[17,18,21,35,37]

From Table II it is apparent that the time step size was not as significant a factor in determining the total computational effort as was the number of degrees of freedom, N. The reported computational times for the 250- and 100-element solutions of a given type are roughly proportional to the corresponding ratios of N for each solution. This occurred despite the large difference in time step size for the two different discretizations. For simulations over longer time periods the maximum time step size will assume greater importance when considering computational efficiency.

Table III presents an operation count for each approach based on N. Note that the self-adaptive approach requires more operations per degree of freedom. This is due to the type B iterations required after introduction of the nodeless variables h'' and to the additional operations required to compute the nodeless variables. In general the number of quadratic elements, N_Q, is much less than N (N_Q is also the number of nodeless variables). Thus the additional computation required by the self-adaptive approach should primarily be a function of the number of iterations (η_2) required to compute the new solution for h after the nodeless variables are introduced. This was shown to be true by computing solutions with different numbers of quadratic elements at a given discretization. In practice η_2 was contained in the range $\eta_1 < \eta_2 < 2\eta_1$. Thus, in order to achieve a savings in computational effort, the number of degrees of freedom (approximately equal to the number of nodes) used by the self-adaptive solution must be less than one-half to one-third

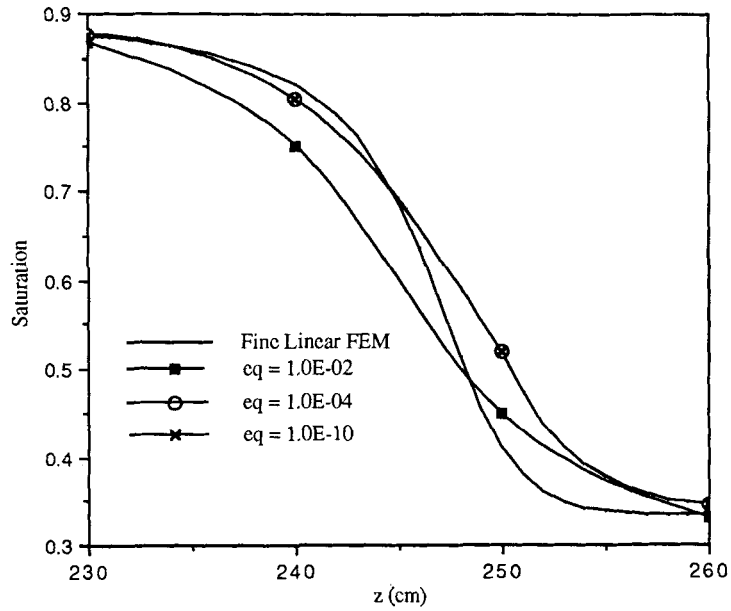

Figure 5. Far-field comparison of the effect of reducing the enrichment criterion; $e_q = 10^{-2}$, 3 quadratic elements; $e_q = 10^{-4}$, 12 quadratic elements; $e_q = 10^{-10}$, 46 quadratic elements

Table III. Operations count per time step

Linear FEM:
$\eta_1 \times N \times 180$ operations (solve for h)
Hierarchic FEM:
linear domain
$\eta_1 \times N \times 180$ operations (solve for h)
enriched domain
$+(\eta_2 - 1) \times N \times 100 + N \times 180$ operations (solve for h) $+ N \times 2$ operations (set N_Q) $+ \eta_2 \times N_Q \times 80$ operations (solve for h'')

the number of degrees of freedom required by the linear solution. This condition was generally satisfied as reported in Table II, where solutions of comparable or superior accuracy were generated by the self-adaptive method using elements between two and three times larger than corresponding solutions computed using only type A iterations. It is interesting to note that the self-adaptive method seems to have greater advantages at coarser discretizations. The above analysis suggests that additional savings in computational effort using the self-adaptive scheme developed in the previous section can be sought in two areas: (1) a reduction in type A iterations and (2) a reduction in type B iterations.

First, a reduction in type A iterations was sought that would not result in a degradation of the final solution quality. From Figure 2 there appears to be substantial differences in the shape and

location of the fronts predicted by each type of solution. This suggests that type A iterations converge more slowly when equation (15) is formed using a solution from the previous time step which was computed by type B iterations. Since the final solution is a result of type B iterations, it may not be necessary to perform type A iterations to convergence. This required a modification of the time step adjustment procedure and made direct comparisons between the two different solution techniques difficult. Numerical trials indicated that more than one type A iteration was necessary to achieve convergence. Significant savings would not be anticipated if η_1 was limited to 2 or 3, since the manner in which the time step was set kept η_1 to 4 or less and since η_2 was generally much larger than η_1. Therefore efforts to improve the computational efficiency by reducing the number of type A iterations were not pursued further.

Secondly, several approaches were investigated to reduce the number of type B iterations. The required type B iterations were typically twice the number of type A iterations per time step. Neither the solution quality nor the computational efficiency improved when the Jacobian matrix was evaluated more frequently during type B iterations as was done during the type A iterations. This behaviour was unexpected and most likely problem-specific.

The use of relaxation to reduce type B iterations was also investigated. The variable h was relaxed during both type A and B iterations and the nodeless variable h'' was relaxed during type B iterations. Relaxation was accomplished by weighting the Δh or $\Delta h''$ computed during each iteration as shown below:

$$h^{\eta+1} = h^{\eta} + \phi \Delta h^{\eta+1}, \qquad 0 < \phi < 2, \tag{21a}$$

$$h''^{\eta+1} = h''^{\eta} + \phi \Delta h''^{\eta+1}, \quad 0 < \phi < 2. \tag{21b}$$

In each equation the superscript indicates the iteration level during a time step. When equation (21a) was used only on the first type B iteration, values of ϕ greater than 1·8 reduced the number of type B iterations by approximately 17%. Otherwise, for this problem, relaxation as described above appeared to have a negligible effect upon both the solution quality and the computational efficiency.

In an effort to speed convergence, different schemes to estimate h'' for the first type B iteration in a time step were examined. These are summarized in Table IV. This series of numerical experiments used smooth initial conditions which allowed a uniform time step size. The most effective estimate resulted from the use of equation (22):

$$h_i''^{t+\Delta t} = (1-\lambda) h_i''^{t} + \lambda h_{i-1}''^{t}, \quad 0 \leqslant \lambda \leqslant 1. \tag{22}$$

Here λ is a measure of the front's spatial movement over a single time step. Equation (22) is only valid for movement from node $i-1$ to node i. For these numerical experiments the weighting factor was calculated *a priori*. Solution quality was not affected by these modifications. The optimum value of λ for this problem was found to be 0·13, which is slightly less than the value expected from a calculation based solely upon the movement of the front, $\lambda = 0{\cdot}15$, and resulted in a reduction in η_2 of approximately 17%. This approach is clearly restricted to single-front

Table IV. Estimation of h'' in method I hierarchic iterations

Estimation equation (on first type B iteration)	Type A iterations	Type B iterations
$h_i''^{t+\Delta t} = 0$	37	76
$h_i''^{t+\Delta t} = h_i''^{t}$	40	77
$h_i''^{t+\Delta t} = (1-\lambda) h_i''^{t} + \lambda h_{i-1}''^{t}$	40	64

problems. Note from Table IV that zero is a better estimator of h'' than the value from the previous time step.

In summary, numerical experiments revealed that improvements in solution quality could be obtained by adjusting the number of quadratic elements introduced at the front. This improvement in accuracy did not increase computational cost. Modest success was also achieved in efforts to reduce computational cost. This was accomplished by overrelaxing h as in equation (21a) during the first type B iteration and by providing a better estimate of h'' with equation (22). A final area of investigation attempted to improve the computational cost and efficiency of the self-adaptive approach by reducing the number of degrees of freedom over which the type B iterations were performed. This work will be discussed in detail in the next section.

SUBDOMAIN DISCUSSION

In this section a modified iteration scheme is presented to account for the non-linearity of the governing equation. This approach can most appropriately be called a self-adaptive subdomain method, hereafter cited as method II, and is loosely based on the self-adaptive multigrid algorithm proposed by Brandt.[38] The subdomain iteration scheme differs from the self-adaptive hierarchic finite element scheme discussed in the preceding sections (method I) in that iterations after enrichment of the interpolating space are performed only over the subdomain of enriched elements. This contrasts with the conventional iteration scheme where iterations after enrichment are performed over a mixed interpolation space covering the global domain. This approach is similar in some respects to the use of telescopic mesh refinement by Ward *et al.*[39] or to the adaptive local grid refinement of Schmidt and Jacobs[40] and Bramble *et al.*[41] In these works, however, h-version mesh refinement was employed. The use of hierarchic basis functions and p-version enrichment offers substantial advantages over h-version refinement. Zhu and Craig[27] discuss some of these advantages, which include easy programming and reduced computational time for a solution of a given quality.

An important benefit of the self-adaptive method I is the identification of regions where the solution is changing rapidly. This identification is used to select elements which will be enriched with the quadratic terms and can serve as an *a priori* error indicator. This identification can be exploited in order to improve the computational efficiency of the self-adaptive finite element method presented in this paper. The elements where quadratics are to be added can be viewed as regions of high-frequency error and the elements which are to remain linear can be viewed as regions of low-frequency error. Linear elements (i.e. low-order accuracy with reduced computational expense) are used to correct for the low-frequency part of the error and quadratic elements (i.e. high-order accuracy with additional computational expense) are used to correct for the high-frequency part of the error. This analysis is similar to the reasoning behind the two-grid arrangement of the self-adaptive multigrid method.[38] Most multigrid work to date uses sequences of increasingly fine meshes to account for the higher-frequency error components and therefore presents some of the same computational difficulties inherent in h-version self-adaptive methods. The use of hierarchic basis functions has been examined in this context and appears to have similar advantages to those discussed earlier in this paper for self-adaptive p-version finite element analysis.[27] In the work presented in this section the low-frequency error is corrected for by the type A iterations over the global domain. Selected elements are then enriched and type B iterations are performed only over the enriched elements, correcting for the high-frequency error.

Although an efficient exact solver, the Thomas algorithm, is available for the one-dimensional unsaturated flow problem, the self-adaptive multigrid concept can be applied to the iteration scheme used to account for the non-linearity of this differential equation. If the

behaviour of the solution at a given time step is examined, it becomes evident that the low-frequency error is controlled by the type A iterations at the beginning of the time step. The subsequent type B iterations, performed after adding quadratic terms to the appropriate elements, function mainly to reduce the high-frequency error at or near the front. The solution elsewhere in the domain is not changed during type B iterations. By performing type B iterations over a reduced domain, a substantial saving in computational effort is possible. For most problems the reduction in the number of degrees of freedom over which the type B iterations will be computed will be large ($N - N_Q - 1$, where $N \gg N_Q$).

In the self-adaptive subdomain method the same criterion used for enrichment of the interpolating space in the preceding simulations ($\Delta s/\Delta z$) is used to define the subdomain where the high-frequency error is potentially significant. Numerical experiments revealed that this criterion had to be less than or equal to 10^{-4} when using 100 elements for the far-field problem in order to introduce enough quadratic elements to avoid oscillations near the subdomain boundaries. Another potential criterion would be the magnitude of the change in s during the last type A iteration. This criterion must be normalized with the time step in order to avoid oscillations in the location of the subdomain boundaries whenever the time step size increased. In the presented simulations the criterion based upon the change in s over an element, equation (18), is used to determine the subdomain.

A subdomain consists entirely of quadratic elements and is decoupled from the global domain by the insertion of first type boundary conditions at each end of the subdomain. The solution obtained using type A iterations provides the values used as boundary conditions. Type B iterations are then performed over the smaller quadratic subdomain until the convergence criterion is satisfied. The solution obtained from type B iterations then replaces the previous solution obtained using type A iterations at the common nodes.

A significant improvement in computational efficiency results from the use of method II. The far-field problem was solved using a discretization of 100 elements. Figure 6 compares solutions computed using 1000 and 100 linear elements with solutions computed using 100 self-adaptive elements using methods I and II. Table V summarizes the relative computational efforts required by each scheme. Comparing the method I solution computed using a criterion on $\Delta s/\Delta z$ of 10^{-2} and the method II solution computed using a criterion on $\Delta s/\Delta z$ of 10^{-4}, method II required one more type B iteration per time step than did method I. Method II would not converge when using a value of $e_q > 10^{-4}$. A method I solution computed using the same criterion of 10^{-4} on $\Delta s/\Delta z$ required the same number of type B iterations as did the method II solution. However, since the type B iterations performed by method II were over a substantially fewer number of degrees of freedom than the type B iterations performed by method I, a significant saving in computational effort resulted from the use of method II as compared to either method I solution. Method II produced a solution of superior quality to that produced by method I (approximately 40% measured by each norm) in 50·3% less CPU time. A method II solution with 100 elements is slightly superior in the L_∞ norm (approximately 15%) than the non-self-adaptive solution with 333 elements ($\Delta z = 1{\cdot}5$ cm), with over a fourfold reduction in required CPU time. These results are also summarized in Table V. Note in Figure 6 that the dispersion observed previously at the tail of the front (Figure 4) is reduced by the use of method II.

The number of type B iterations required by the method II solution increases with the size of the enriched subdomain (Table V). This is expected since the larger subdomains incorporate areas where the high-frequency error is increasingly less significant. The computation time is relatively insensitive to small changes in the size of the subdomains since the contribution of the type B iterations to the total computation time is small. The same is not true for the error norms, which are much more sensitive to the size of the subdomain. This may be a result of greater efficiency of

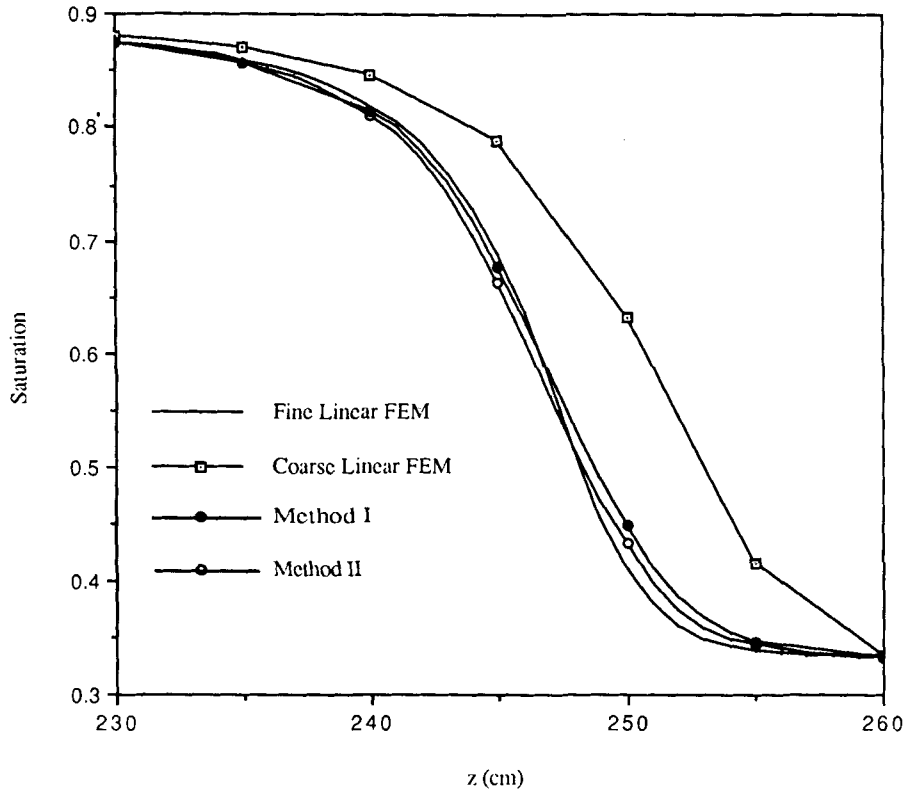

Figure 6. Far-field comparison of method I, $e_q = 10^{-2}$, and method II, $e_q = 10^{-4}$

Table V. Comparison of method I and method II self-adaptive hierarchic schemes for the far-field problem

Element type	Errors norms L_2	Errors norms L_∞	Iterations Type A	Iterations Type B	CPU time (s)	Number of degrees of freedom in enriched domain
($\Delta z = 5{\cdot}0$ cm)						
Linear	19·09	14·90	887	—	133	—
Method I	4·06	3·64	885	1580	313	104 ($e_q = 10^{-2}$)
	5·83	5·18	885	1729	333	113 ($e_q = 10^{-4}$)
Method II	2·80	2·18	893	1732	155	25 ($e_q = 10^{-4}$)
	5·83	5·18	893	1884	171	43 ($e_q = 10^{-6}$)
($\Delta z = 1{\cdot}5$ cm)						
Linear	4·94	2·57	1378	—	686	—

the type B iterations when the high-frequency error is more evenly distributed over the subdomain. In general the use of method II produced greatly superior results at a given discretization, both in terms of solution accuracy and in required computation time, in comparison with method I or the non-self-adaptive finite element approach.

CONCLUSIONS

The principal intent of this research was to investigate the use and refinement of the self-adaptive hierarchic finite element method for the solution of the non-linear unsaturated flow equation. The selection of this particular self-adaptive approach was based upon considerations relating to the intended application of the developed method and results of numerical simulations which indicated that hierarchic enrichment of the interpolation space yielded advantages in efficiency and accuracy over other self-adaptive approaches. The self-adaptive hierarchic method used was shown to be more accurate at a given discretization than the non-self-adaptive linear finite element method used as a basis for comparison. In addition, the original self-adaptive finite element method investigated was shown to be more efficient than the non-self-adaptive finite element method in computing solutions of comparable accuracy. Not only did the self-adaptive method require less CPU time, but substantially fewer degrees of freedom were required, reducing demands upon computer memory. The saving appeared to become more significant as the grid spacing coarsened.

A modification of the original self-adaptive approach was then investigated. The self-adaptive subdomain method can be described as an application of a self-adaptive hierarchic two-grid iteration scheme to account for the non-linearity of the governing equation. This method yielded significant reductions in the computational time required to generate a solution of a given accuracy, nearly 50% when compared to the original self-adaptive method. When compared to a non-self-adaptive finite element solution of comparable accuracy, the use of method II resulted in over a fourfold reduction in required CPU time. In addition, improvements in the solution accuracy were observed at a given discretization when compared to the original self-adaptive approach.

This work has demonstrated the utility of the self-adaptive hierarchic finite element method for the solution of the non-linear unsaturated flow equation. In particular, the application of the self-adaptive subdomain method shows great promise. Several issues are currently being investigated: (1) the use of higher-order elements ($p > 2$); (2) application to problems of higher dimension; and (3) the development and use of local error estimators, both as convergence and enrichment criteria. The extension of the self-adaptive hierarchic finite element method to problems of higher dimension shows particular promise. While the matrix structure is not as elegant in higher dimensions ($\mathbf{M}_{22}$ may not be diagonal, depending upon the choice of basis functions), the ease of transferring nodal variables and related coefficients still leads to potential computational advantages. The nodal connectivity does not change when the interpolation space is enriched hierarchically. The application of slice-successive-overrelaxation methods to extend a 2D code to 3D would be straightforward since each slice could be adapted independently of the other slices without losing correspondence at the nodes. The ability of the presented method to generate acceptable solutions on coarse grids suggests that the use of higher-order basis functions may be well suited to the solution of two- and three-dimensional problems on the coarse meshes required to simulate large-field-scale problems.

REFERENCES

1. I. Babuska and W. Gui, 'Basic principles of feedback and adaptive approaches in the finite element method', *Comput. Methods Appl. Mech. Eng.*, **55**, 27–42 (1986).
2. P. S. Huyakorn and G. F. Pinder, *Computational Methods in Subsurface Flow*, Academic Press, New York, 1983, pp. 146–150.
3. R. Haverkamp, M. Vauclin, J. Touma, P. J. Wierenga and G. Vachaud, 'A comparison of numerical simulation models for one-dimensional infiltration', *Soil Sci. Soc. Am. J.* **41**, 285–294 (1977).
4. M. Th. van Genuchten, 'Progress in unsaturated flow and transport modeling', *Rev. Geophys.* **25**, 135–140 (1987).

5. S. I. Bhuiyan, E. A. Hiler, C. H. M. Van Bavel and A. R. Aston, 'Dynamic simulation of vertical infiltration into unsaturated soils', *Water Resources Res.* **7,** 1597–1606 (1971).
6. A. E. Riesenauer, 'Methods for solving problems of multidimensional partially saturated steady flow in soils', *J. Geophys. Res.*, **68,** 5725–5733 (1963).
7. R. A. Freeze, 'Three-dimensional, transient, saturated-unsaturated flow in a groundwater basin', *Water Resources Res.*, **7,** 347–366 (1971).
8. W. F. Brutsaert, 'A functional iteration technique for solving the Richards equation applied to two-dimensional infiltration problems', *Water Resources Res.*, **7,** 1583–1596 (1971).
9. R. L. Cooley, 'A finite difference method for unsteady flow in variably saturated porous media: application to a single pumping well', *Water Resources Res.*, **7,** 1607–1625 (1971).
10. H. J. Morel-Seytoux and J. A. Billica, 'A two-phase numerical model for prediction of infiltration: applications to a semi-infinite soil column', *Water Resources Res.* **21,** 607–615 (1985).
11. J. J. Rulon, R. Rodway and R. A. Freeze, 'The development of multiple seepage faces on layered slopes', *Water Resources Res.*, **21,** 1625–1636 (1985).
12. S. P. Neuman, 'Saturated–unsaturated seepage by finite elements', *J. Hydraul. Div. (ASCE)*, **99,** 2233–2250 (1973).
13. M. Th. van Genuchten, 'A comparison of numerical solutions of the one-dimensional unsaturated–saturated flow and mass transport equations', *Adv. Water Resources*, **2,** 47–55 (1982).
14. T. J. McKeon and W.-S. Chu, 'A multigrid model for steady flow in partially saturated porous media', *Water Resources Res.*, **23,** 542–550 (1987).
15. S. Sorek and C. Braester, 'An adaptive Eulerian–Lagrangian approach for the numerical simulation of unsaturated flow', *Proc. Sixth Int. Conf. on Finite Elements in Water Resources*, Lisbon, 1986, pp. 87–100.
16. L. M. Abriola, 'Finite element solution of the unsaturated flow equation using hierarchic basis functions', *Proc. Sixth Int. Conf. on Finite Elements in Water Resources*, Lisbon, 1986, pp. 125–133.
17. P. K. Basu and A. G. Peano, 'Adaptivity in *p*-version finite element analysis', *J. Struct. Eng.*, **109,** 2310–2324 (1983).
18. O. C. Zienkiewicz, J. P. De S. R. Gago and D. W. Kelly, 'The hierarchical concept in finite element analysis', *Comput. Struct.*, **16,** 53–65 (1983).
19. I. Babuska and B. Szabo, 'On the rates of convergence of the finite element method', *Int. j. numer. methods eng.*, **18,** 323–341 (1982).
20. N. Kikuchi, 'Adaptive grid-design methods for finite element analysis', *Comput. Methods Appl. Mech. Eng.*, **55,** 129–160 (1986).
21. L. Demkowicz, Ph. Devloo and J. T. Oden, 'On an *h*-type mesh refinement strategy based on minimization of interpolation errors', *Comput. Methods Appl. Mech. Eng.*, **53,** 67–89 (1985).
22. R. E. Benner, Jr., H. T. Davis and L. E. Scriven, 'An adaptive finite element method for steady and transient problems', *SIAM J. Sci. Statist. Comput.* **8,** 529–549 (1987).
23. J. Z. Zhu and O. C. Zienkiewicz, 'Adaptive techniques in the finite element method', *Commun. Appl. Numer. Methods*, **4,** 197–204 (1988).
24. M. Bieterman and I. Babuska, The finite element method for parabolic equations. Part I: *A posteriori* error estimation, *Numer. Math.*, **40,** 339–371 (1982).
25. M. Bieterman and I. Babuska, 'The finite element method for parabolic equations. Part II: *A posteriori* error estimation and adaptive approach', *Numer. Math.*, **40,** 373–406 (1982).
26. A Peano, A. Pasini, R. Riccioni and L. Sardella, 'Adaptive approximations in finite element structural analysis', *Comput. Struct.*, **10,** 333–342 (1979).
27. J. Z. Zhu and A. W. Craig, 'Finite element multigrid algorithms and their application to engineering problems', in J. Middleton and G. N. Pande (eds), *NUMETA 85 Numerical Methods in Engineering: Theory and Applications*, Vol. 2, A. A. Balkema, Rotterdam and Boston, 1985, pp. 927–932.
28. M. A. Crisfield, 'Some recent research on numerical techniques for structural analysis', in J. Middleton and G. N. Pande (eds), *NUMETA 85 Numerical Methods in Engineering: Thoery and Applications*, Vol. 2, A. A. Balkema, Rotterdam and Boston, 565–575 (1985).
29. A. G. Peano, B. A. Szabo and A. K. Mehta, 'Self-adaptive finite elements in fracture mechanics', *Comput. Methods Appl. Mech. Eng.*, **16,** 69–80 (1978).
30. I. Babuska, B. A. Szabo and I. N. Katz, 'The *p*-version of the finite element method', *SIAM J. Numer. Anal.*, **18,** 515–545 (1981).
31. M. R. Dorr, 'The approximation of solutions of elliptic boundary-value problems via the *p*-version of the finite element method', *SIAM J. Numer. Anal.*, **23,** 58–77 (1986).
32. N.-E. Wiberg and P. Möller, 'Formulation and solution of hierarchical finite element equations', *Int. j. numer. methods eng.*, **26,** 2131–1233 (1988).
33. O. Friberg, P. Möller, D. Makovicka and N.-E. Wiberg, 'An adaptive procedure for eigenvalue problems using the hierarchical finite element method', *Int. j. numer. methods eng.*, **24,** 319–335 (1987).
34. L. Demkowicz, J. T. Oden and T. Strouboulis, 'Adaptive finite elements for flow problems with moving boundaries. Part I: Variational principles and *a posteriori* estimates', *Comput. Methods Appl. Mech. Eng.*, **46,** 217–251 (1984).
35. I. Babuska and W. C. Rheinbolt, 'Computational error estimates and adaptive processes for some nonlinear structural problems', *Comput. Methods Appl. Mech. Eng.*, **34,** 895–937 (1982).
36. A. Peano, 'Hierarchies of conforming finite elements for plane elasticity and plate bending', *Comput. Math. Appl.*, **2,** 211–224 (1976).

37. O. C. Zienkiewicz and J. Z. Zhu, 'A simple error estimator and adaptive procedure for practical engineering analysis', *Int. j. numer. methods eng.*, **24,** 337–357 (1987).
38. A. Brandt, 'Multi-level adaptive solutions to boundary-value problems' *Math. Comput.*, **31,** 333–390 (1977).
39. D. S. Ward, D. R. Buss, J. W. Mercer and S. S. Hughes, 'Evaluation of a groundwater corrective action at the Chem-Dyne hazardous waste site using a telescopic mesh refinement modeling approach', *Water Resources Res.*, **23,** 603–617 (1987).
40. G. H. Schmidt and F. J. Jacobs, 'Adaptive local grid refinement and multi-grid in numerical reservoir simulation', *J. Comput. Phys.*, **77,** 140–165 (1988).
41. J. H. Bramble, R. E. Ewing and J. E. Pasciak, 'A preconditioning technique for the efficient solution of problems with local grid refinement', *Comput. Methods Appl. Mech. Eng.*, **67,** 149–159 (1988).