INEXACT NEWTON METHODS AND THE METHOD OF LINES FOR SOLVING RICHARDS' EQUATION IN TWO SPACE DIMENSIONS*

MICHAEL D. TOCCI†, C. T. KELLEY†, CASS T. MILLER‡, AND CHRISTOPHER E. KEES‡

Abstract. Richards' equation (RE) is often used to model flow in unsaturated porous media. This model captures physical effects, such as sharp fronts in fluid pressures and saturations, which are present in more complex models of multiphase flow. The numerical solution of RE is difficult not only because of these physical effects but also because of the mathematical problems that arise in dealing with the nonlinearities. The method of lines has been shown to be very effective for solving RE in one space dimension. When solving RE in two space dimensions, the nonlinear equations that must be solved in an implicit time-stepping approach usually become impractical to solve using a direct solver (e.g., LU decomposition). In this work, we show how Newton-iterative methods can be applied in this context; these methods use iterative linear solvers, such as the Krylov methods GMRES and Bi-CGSTAB. We present a theorem on the convergence of inexact Newton methods, and based on this theorem, an adaptive method that selects an appropriate linear tolerance. Numerical results show the method to be effective compared with an existing approach.

Key words. Richards' equation, method of lines, inexact Newton methods

AMS subject classifications. 65F10, 65H10, 65M06, 65M20, 76S05

1. Introduction.

This paper extends the work presented in [14] and [7], where the method of lines (MOL) was shown to be an effective approach to solve Richards' equation (RE) [11] numerically in one space dimension. Here we extend that work to two space dimensions.

RE models variably-saturated water flow in a rigid porous media. RE results from a mass conservation law for a two-fluid system (water and gas) in which the assumption of constant gas-phase pressure has been applied. This assumption is justified for many systems because a very small gas-phase pressure gradient is needed to support the flow of a gas phase compared to the pressure gradient needed to support an equal volumetric flow of an aqueous phase. Constitutive relations are required to close the equation; we detail this formulation below.

The pressure head form of RE in two space dimensions (x, z) is

$$[c(\psi) + S_s S_a(\psi)] \frac{\partial \psi}{\partial t} = \frac{\partial}{\partial z}\left[K(\psi)\left(\frac{\partial \psi}{\partial z} + 1\right)\right] + \frac{\partial}{\partial x}\left[K(\psi)\frac{\partial \psi}{\partial x}\right] \tag{1.1}$$

where ψ is pressure head; $c(\psi) = \partial\theta/\partial\psi$ is the specific moisture capacity; $\theta(\psi)$ is the volumetric fraction of the water phase; S_s is the specific storage, which accounts for the slight compressibility of water; $S_a(\psi) = \theta(\psi)/n$ is the aqueous-phase saturation; n is the porosity of the porous media; and $K(\psi)$ is the variably-saturated hydraulic conductivity. It is common to let $A(\psi) = c(\psi) + S_s S_a(\psi)$, where $A(\psi)$ is called the accumulation term. In this formulation, the z axis is the vertical direction oriented positively upward. In order to close (1.1), relations among ψ, $\theta(\psi)$, and $K(\psi)$

*Version of November 20, 1997. This research was supported by Army Research Office grant #DAALA03-92-G-0111, a Cray Research Corporation Fellowship, National Science Foundation grant #DMS-9700569, US Army contract #DACA39-95-K-0098, and a U. S. Department of Education GAANN fellowship. Computing activity was partially supported by an allocation from the North Carolina Supercomputing Center.

†Center for Research in Scientific Computation, Department of Mathematics, Box 8205, North Carolina State University, Raleigh, NC 27695-8205, USA mdtocci@unity.ncsu.edu, Tim_Kelley@ncsu.edu

‡Center for Multiphase Research, Department of Environmental Sciences and Engineering, CB 7400, University of North Carolina, Chapel Hill, NC 27599-7400, USA casey_miller@unc.edu, chris_kees@unc.edu

must be specified. We use the van Genuchten [17] and Mualem [10] relationships in this work. First, we define the effective saturation, S_e, using the van Genuchten relation:

$$S_e(\psi) = \frac{\theta - \theta_r}{\theta_s - \theta_r} = (1 + |\alpha_\nu \psi|^{n_\nu})^{-m_\nu} \tag{1.2}$$

where θ_r is the residual volumetric water content, θ_s is the saturated volumetric water content, α_ν is an experimentally-determined coefficient that is related to the mean pore size, n_ν is an experimentally determined coefficient related to the variation in pore sizes, and $m_\nu = 1 - 1/n_\nu$.

The variably saturated hydraulic conductivity is defined using Mualem's model:

$$K(\psi) = K_s S_e^{1/2} \left[1 - \left(1 - S_e^{1/m_\nu}\right)^{m_\nu}\right]^2 \tag{1.3}$$

where K_s is the water-saturated hydraulic conductivity. Parameters in the saturation and conductivity relations influence the speed and slope of moving fronts in the solution, thereby affecting the difficulty of the problem.

We use the method of lines (MOL) to solve RE numerically in this work. The MOL is a common approach [1] used to solve time-dependent partial differential equations (PDE's). In this approach, the PDE is discretized in space using finite differences or finite elements, and then the resulting system of ordinary differential equations (ODE's) is solved using an ODE or differential algebraic equation (DAE) solver. In [7], [9], and [14] we showed how the spatial discretization of RE was most effectively solved as a DAE [*i.e.* not dividing by $c(\psi) + S_s S_a(\psi)$ in (1.1)]. In our work so far, we have used the DAE solver DASPK [4] to solve the resulting ODE's. This code uses backward differentiation formulae (BDF) with a predictor-corrector approach to solve systems of DAE's. The work presented here is not specific to this code, and could apply in general to any BDF approach.

When solving RE in one space dimension, the size of the spatial mesh, which is the size of the system of DAE's, is relatively small. Hence we can use a direct linear solver (LU decomposition) to solve the linear systems that arise in the time integration. But in higher dimensions, the problems become prohibitively large for a direct linear solver, so an iterative method must be used. We consider the Krylov space methods GMRES [12] and Bi-CGSTAB [16]. GMRES has better convergence properties but requires more memory. These memory requirements make Bi-CGSTAB appealing for very large problems.

Using the MOL approach to solve (1.1) requires the solution of a nonlinear system at every time step. The standard approach for most ODE/DAE codes is to use Newton's method or a variant such as the chord method. In this paper we consider another variant called an inexact Newton method [6], [5]. In an inexact Newton method, the linear system for the Newton step is solved using an iterative method, such as those described above. Previous work [2], [3], [4] has shown how to implement this method into an ODE/DAE solver, but some questions remain on how to terminate the linear iteration effectively. If the termination criteria is too tight, then the method may be inefficient, and if the termination criteria is too loose, the method may no longer be accurate. To select an appropriate termination criterion, we present a theorem in §2 and, in §3, an adaptive algorithm and some numerical results.

2. Inexact Newton Methods. In this section, we present an extension of the convergence results in [2] and [3] on the convergence of Newton's method when an iterative method is used to solve the linear system for the step. Our convergence result shows the relation between the effect of the linear solver's convergence criterion and the reduction in the nonlinear error in a way that can be used to design a strategy for managing the linear solver as the temporal integration progresses.

Before applying our analysis to RE, we will present some results for a general nonlinear system, which is written as

$$F(x) = 0 \tag{2.1}$$

The difference between a standard implementation of Newton's method and an inexact Newton method lies in how the step is calculated. For standard Newton's method, one iterate can be written as

$$\begin{aligned} s &= -F'(x_c)^{-1}F(x_c) \\ x_+ &= x_c + s \end{aligned}$$

The notation used here is defined as follows. The current iterate is x_c, and the iterate we are solving for is x_+. The Newton step to go from x_c to x_+ is defined as s. Similarly, the current error is defined as $e_c = x_c - x^*$ and the error at the next iterate is $e_+ = x_+ - x^*$, where x^* is the exact solution to (2.1). An LU decomposition is typically used to calculate s.

For an inexact Newton method, s is calculated so that

$$\|F'(x_c)s + F(x_c)\| \leq \eta \|F(x_c)\|$$

This approach means that we apply an iterative method to the linear system

$$F'(x_c)s = -F(x_c) \tag{2.2}$$

until the relative residual is reduced by a factor of η, where η is called the linear tolerance, or forcing term. It is assumed here that the initial iterate for the linear solver is $s = 0$, as is true for most ODE/DAE solvers. It can be shown that if η is small enough, then such a scheme converges linearly and the error satisfies [6]

$$\|e_+\| \leq K(\|e_c\| + \eta)\|e_c\|$$

Since the initial residual for ODE and DAE solvers is often quite small, we would rather solve (2.2) until

$$\|F'(x_c)s + F(x_c)\| \leq \eta \tag{2.3}$$

This decision, explained in more detail in [2] and [4], is based on the desire for efficiency. It is well known that Newton's method is not guaranteed to converge when using (2.3) as a termination criteria for the linear method, although it is known that the method will not diverge [2]. With a few further assumptions, this next theorem will show the local convergence rate of Newton's method is linear.

THEOREM 2.1. *Let the standard assumptions [6] (Lipschitz continuity and nonsingularity of F' near x^*, a root of F) for local quadratic convergence of Newton's method hold. Assume that $x_+ = x_c + s$ where the step s satisfies*

$$\|F'(x_c)s + F(x_c)\| \leq \eta_c \tag{2.4}$$

Let $0 < \epsilon < 1$ be the desired nonlinear tolerance, and assume that $||e_c|| \geq \epsilon$. Let $0 < \delta < 1$, then if

$$\eta_c \leq \frac{\epsilon\delta}{||F'(x_c)^{-1}||} \tag{2.5}$$

then the error satisfies

$$||e_+|| \leq K||e_c||^2 + \delta||e_c|| \tag{2.6}$$

Proof: Let the linear residual be,

$$r = -F'(x_c)s - F(x_c)$$

Then,

$$s + F'(x_c)^{-1}F(x_c) = -F'(x_c)^{-1}r$$

and

$$e_+ = e_c + s = e_c - (F'(x_c)^{-1}F(x_c) + F'(x_c)^{-1}r)$$

Using the assumption on the linear residual in (2.4) we have,

$$\begin{aligned} ||e_+|| &= ||e_c + s|| \\ &\leq ||e_c - F'(x_c)^{-1}F(x_c)|| + ||F'(x_c)^{-1}r|| \\ &\leq ||e_c - F'(x_c)^{-1}F(x_c)|| + ||F'(x_c)^{-1}||\,||r|| \\ &\leq K||e_c||^2 + ||F'(x_c)^{-1}||\eta_c \end{aligned}$$

This last step is true because standard Newton's method converges quadratically [6]. Using (2.5) and the fact that $||e_c|| \geq \epsilon$ we have,

$$||e_+|| \leq K||e_c||^2 + \delta||e_c||$$

This completes the proof. □

This theorem states the requirements necessary for local linear convergence of the nonlinear iteration. In order to have linear convergence of the complete nonlinear iteration, the initial iterate must be close to the exact solution. The next theorem states this result precisely.

THEOREM 2.2. *Let the assumptions in Theorem 2.1 hold. If there is a γ such that $\gamma \geq \epsilon$ and $K\gamma + \delta < 1$ where K is from (2.6), and $||x_0 - x^*|| \leq \gamma$ then the inexact Newton iteration*

$$x_{n+1} = x_n + s_n$$

where

$$||F'(x_n)s_n + F(x_n)|| \leq \eta_n$$

will satisfy the following relation for all $n \geq 0$ such that $||e_n|| \geq \epsilon$

$$||e_{n+1}|| \leq (K\gamma + \delta)||e_n|| < ||e_n|| \tag{2.7}$$

Proof: Since we are making the same assumptions as in Theorem 2.1, (2.6) still holds. Reduce γ and/or δ until

$$K\gamma + \delta < 1$$

If $\gamma < \epsilon$ the theorem does not hold. If $\gamma \geq \epsilon$ then for all $n \geq 0$ such that $||e_n|| \geq \epsilon$ we have

$$||e_{n+1}|| \leq (K||e_n|| + \delta)||e_n|| \leq (K\gamma + \delta)||e_n|| < ||e_n||$$

This proves (2.7) and the theorem. $\Box$

These two theorems show that under certain conditions, the inexact Newton method used in DASPK will converge linearly until the nonlinear tolerance is satisfied. A heuristic argument can be made that either these conditions are satisfied, or the code can detect when the conditions are not true. One assumption that must be made is that the code can detect when the error has dropped below the tolerance and terminate the iteration. The validity of this assumption is further discussed in [7]. In Theorem 2.2, we had to assume that $\gamma \geq \epsilon$ to get linear convergence. This means that the initial error can not be required to be below the nonlinear tolerance. It can be assumed that the code can detect when this happens because it makes an estimate of the convergence rate during the iteration. If this estimate is too large, then the iteration is terminated and there is a stepsize and/or an order reduction. So it can be argued that the conditions of these theorems are not satisfied when the stepsize and/or order is too large, and that the code can detect when this occurs and remedy the situation.

The next step in developing the adaptive scheme is to apply Theorem 2.1 to the nonlinear system that must be solved in a DAE code. Assume the DAE we are solving can be written as $F(t, y, y') = 0$. Using the notation from DASPK [4], the nonlinear system to be solved is

$$F(t, y, \alpha y + \beta) = 0 \tag{2.8}$$

where α depends on the order of the BDF being used, and β depends on previous solution values. For simplicity, write (2.8) as $F(y) = 0$. Also, since we are using a linear iterative method, there must be some kind of reasonable preconditioner P so that $P \approx F'(y)$. So the nonlinear equation that must be solved can be written as

$$G(y) = P^{-1}F(y) = 0 \tag{2.9}$$

The final step is to decide what norm to use to terminate the iteration. In DASPK, the nonlinear iteration is terminated when

$$||e_+||_{WRMS} = ||D^{-1}e_+||_2 \leq \epsilon \tag{2.10}$$

where D^{-1} is a scaling matrix with the tolerances built in, and ϵ is the nonlinear tolerance, which is set to 0.33 in DASPK. The norm $||\cdot||_{WRMS}$ is called the weighted root mean square norm and is defined by

$$||x||_{WRMS} = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_{(i)}}{rtol|y_{(i)}| + atol}\right)^2\right]^{1/2} \tag{2.11}$$

where $rtol$ and $atol$ are relative and absolute tolerances provided by the code user. The way this norm is defined implies that if $||x||_{WRMS} < 1$, then the vector x meets the error requirements. Therefore, setting $\epsilon = 0.33$ in (2.10) implies that DASPK is requiring the nonlinear system be solved to a lower tolerance than the user is requesting. For the rest of this section, it is assumed that $||\cdot|| = ||\cdot||_{WRMS}$ and $rtol = atol = tol$ unless otherwise noted. Of course an estimate for e_+ must be used in (2.10), since it is unknown. This topic is further discussed in [1], [7], and [14].

3. Adaptive Method and Numerical Results. In this section we present a method to select an appropriate linear tolerance based on the analysis in §2. A two dimensional (2-D) test problem is also presented, along with numerical results comparing the adaptive method to the default method in DASPK.

3.1. Adaptive Scheme. Using the definition of the error norm in (2.11) and the nonlinear function given in (2.9), along with Theorem 2.1, we can form an algorithm that will automatically choose an appropriate value of η. Note that we are applying the theorem to the function G, which is the preconditioned function. The theorem states that if we force η to satisfy (2.5), then Newton's method will converge linearly with a convergence rate of δ. The problem now becomes how do we estimate $||G'(y_c)^{-1}||$ efficiently and accurately. Using the statistical method presented in [8], it is possible to get an estimate for $||G'(y_c)^{-1}||$ by solving a linear system.

The algorithm to set η is as follows. The value of the nonlinear tolerance ϵ is set to 0.33 in DASPK. Several different values of δ will be tested in the next section. The final step in the algorithm is to estimate $||G'(y_c)^{-1}||$, which is done as follows. First, choose a random vector z so that $||z|| = 1$. Then the estimate is

$$||G'(y_c)^{-1}|| \approx \frac{||G'(y_c)^{-1}z||}{E_n}$$

where n is the size of the matrix and E_n is defined as

$$E_n = \begin{cases} \frac{1\cdot 3\cdots(n-2)}{2\cdot 4\cdots(n-1)} & \text{for } n \text{ odd} \\ \\ \frac{2}{\pi}\frac{2\cdot 4\cdots(n-2)}{1\cdot 3\cdots(n-1)} & \text{for } n \text{ even} \end{cases}$$

This approximation involves solving a linear system using the same iterative method as in the inexact Newton iteration. Some extra error will show up in our approximation, but since we only need a gross approximation of the norm, that error should not greatly affect our results. The linear tolerance for this solutin is set to 0.05 at present. This value is the result of experimentation, and further research may be needed for this issue.

Once an estimate for this norm is made, the value of η can be set using the algorithm

$$\eta = \frac{\epsilon\delta}{\nu} \tag{3.1}$$

where

$$\nu = \begin{cases} 1, & ||G'(y_c)^{-1}|| \leq 100 \\ \\ \frac{||G'(y_c)^{-1}||}{100}, & ||G'(y_c)^{-1}|| > 100 \end{cases} \tag{3.2}$$

This is an empirical formula that only affects the linear tolerance when the preconditioner is not optimal. In our case, we are using Jacobi preconditioning, which is easy to implement but is not as effective as other methods such as domain decomposition [13]. The strategy used in DASPK currently is to set $\nu = 1$ in (3.1), which agrees with our algorithm only when the preconditioner is very effective.

Calculate ν at every Newton iterate or even at every time step is inefficient, so a decision must be made on when to make this calculation. The estimate is made on the first time step and then 50 time steps later. From then on, if the estimate has changed by a factor of more than 100, indicating the norm is rapidly changing, the number of time steps between updating the estimate is reduced by 10 to a minimum of 10, otherwise it is increased by 10 to a maximum of 100.

3.2. Heterogeneous Test Problems. In order to test the performance of the adaptive scheme, we considered two test problems, which have identical spatial domains but slightly different boundary conditions and different temporal domains. The spatial domain for the test problem is a 2-D rectangle with dimensions $[0, 5\text{ m}] \times [0, 10\text{ m}]$ discretized so $\Delta x = 0.05$ m and $\Delta z = 0.1$ m. The spatial domain is occupied by two different types of porous media, the location of which is shown in Figure 3.1. Table 3.1 shows the parameter values corresponding to these materials.

The initial and boundary conditions are as follows,

$$\frac{\partial \psi}{\partial x}(x = 0, z, t) = 0$$

$$\frac{\partial \psi}{\partial x}(x = 5, z, t) = 0$$

$$\frac{\partial \psi}{\partial z}(x, z = 0, t) = -1$$

$$\psi(1.65 < x < 3.35, z = 10, t) = \psi_0$$

$$\frac{\partial \psi}{\partial z}(0 < x < 1.65, z = 10, t) = -1$$

$$\frac{\partial \psi}{\partial z}(3.35 < x < 5, z = 10, t) = -1$$

$$\psi(x, z, t = 0) = -z$$

The value of ψ_0 is set to 0.1 m for Problem 1, and to -0.5 m for Problem 2. Problem 1 is more difficult for the DAE solver than Problem 2, as a result of the development of a saturated region in the former case.

TABLE 3.1
Model parameter values for Material A and Material B.

Variable	Material A	Material B
θ_r	1.02×10^{-1}	9.30×10^{-2}
θ_s	3.68×10^{-1}	3.01×10^{-1}
α_ν (m^{-1})	3.35×10^{0}	5.47×10^{0}
n_ν	2.00×10^{0}	4.26×10^{0}
K_s (m/day)	7.97×10^{0}	5.04×10^{0}
S_s (m^{-1})	0.00×10^{0}	1.00×10^{-6}

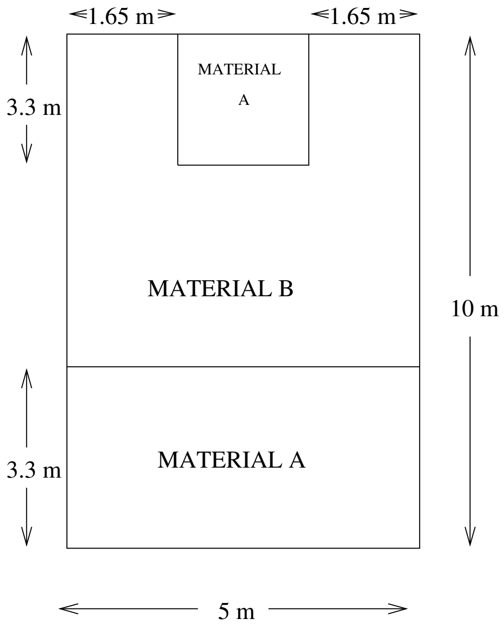

FIG. 3.1. *The 2-D domain used for all computations in this chapter.*

The temporal domain for Problem 1 is $t \in [0, 0.35]$. The temporal domain for Problem 2 is $t \in [0, 9.0]$. These domains differ substantially because of the desire to develop a moisture infiltration profile that moved substantially into the domain for both cases and the slightly different boundary conditions.

We use a spatial discretization such that material properties change along boundaries between nodes, such that parameter $p_{i,j}$ corresponds to a material property located in the positive x and z direction from a node positioned at (x_i,z_j). Following this convention, we compute $A(\psi_{i,j})$ as

$$A(\psi_{i,j}) = A_{i-1,j-1}(\psi_{i,j}) + A_{i,j-1}(\psi_{i,j}) + A_{i-1,j}(\psi_{i,j}) + A_{i,j}(\psi_{i,j})$$

Conductivities are computed using an arithmetic mean approximation, which for the constant spatial discretization approach used here gives

$$\begin{aligned}
K_{i-1/2,j} &= [K_{i-1,j-1}(\psi_{i-1,j}) + K_{i-1,j}(\psi_{i-1,j}) + K_{i-1,j-1}(\psi_{i,j}) + K_{i-1,j}(\psi_{i,j})]/4 \\
K_{i+1/2,j} &= [K_{i,j-1}(\psi_{i,j}) + K_{i,j}(\psi_{i,j}) + K_{i,j-1}(\psi_{i+1,j}) + K_{i,j}(\psi_{i+1,j})]/4 \\
K_{i,j-1/2} &= [K_{i-1,j-1}(\psi_{i,j-1}) + K_{i,j-1}(\psi_{i,j-1}) + K_{i-1,j-1}(\psi_{i,j}) + K_{i,j-1}(\psi_{i,j})]/4 \\
K_{i,j+1/2} &= [K_{i-1,j}(\psi_{i,j}) + K_{i,j}(\psi_{i,j}) + K_{i-1,j}(\psi_{i,j+1}) + K_{i,j}(\psi_{i,j+1})]/4
\end{aligned}$$

3.3. Numerical Results. In this section, we discuss some numerical results for the problems described above. The plots in Figures 3.2 and 3.3 show the solutions to these problems run with *tol* = 1.0e-8 and $\delta = 1.0e-4$, which means that $\eta = 3.3e-5$. We will use these as our exact solutions to compare to other numerical solutions. The mesh used has $\Delta x = 0.05$ m and $\Delta z = 0.1$ m which means there are 99×99 ODE's to be solved. The way we will compare solutions is to take either a scaled l_1 or l_2 norm of the difference between the exact solution and the computed solution. If we let ψ_N be the numerical solution and ψ_{exa} be the exact solution as defined above then the two error norms are defined as,

$$l_p \text{ error } = (\Delta x \Delta z)^{1/p} \|\psi_N - \psi_{exa}\|_p,\ p = 1, 2 \tag{3.3}$$

This is similar to the coarse error defined in [7]. Also, there are some implementation details that must be given. The preconditioner used in all of these experiments is Jacobi (or diagonal) preconditioning and the preconditioner is calculated at the beginning of every time step. The action of a Jacobian on a vector v is calculated by a difference quotient using,

$$G'(y)v \approx \frac{G(y + hv) - G(y)}{h}$$

In DASPK, the difference step h is taken to be one because v is assumed to be small in the weighted root mean square norm. This works well for GMRES, but did not work for Bi-CGSTAB. Therefore we set $h = 1.0e - 5$ for all of the experiments.

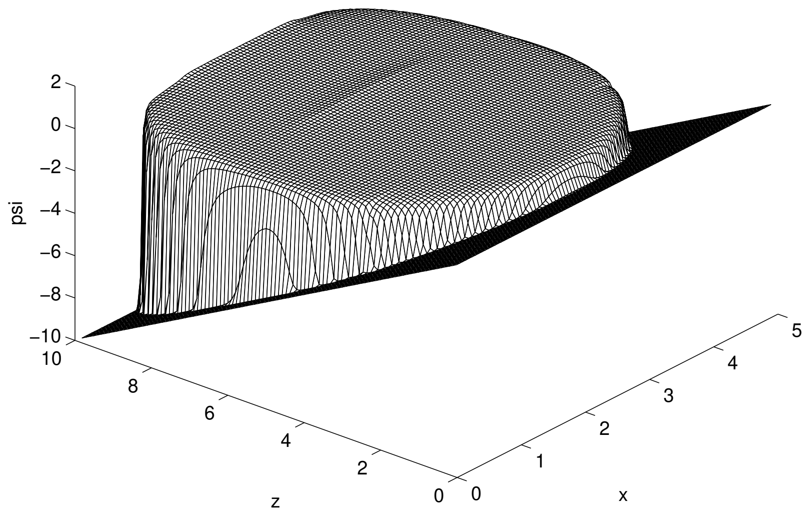

FIG. 3.2. *A plot of the 'exact' solution to Problem 1.*

Results shown in [15] indicate that using a linear tolerance that is too large can result in incorrect solutions or convergence problems, and using a linear tolerance that is too small can result in an inefficient method. The goal of the adaptive scheme outlined in §3.1 is to automatically select an η restrictive enough to give reasonable results without becoming inefficient. To test this, we ran DASPK on the two problems with two different tolerances, and for each of the tolerances various values of δ were chosen. This was tested against our adaptive scheme with $\delta = 0.05, 0.1, 0.2$, by comparing the scaled l_2 errors against the number of function calls. In this case, a function call is counted whenever the nonlinear function G in (2.9) is evaluated. This set of experiments was performed using both GMRES and Bi-CGSTAB as the linear solver.

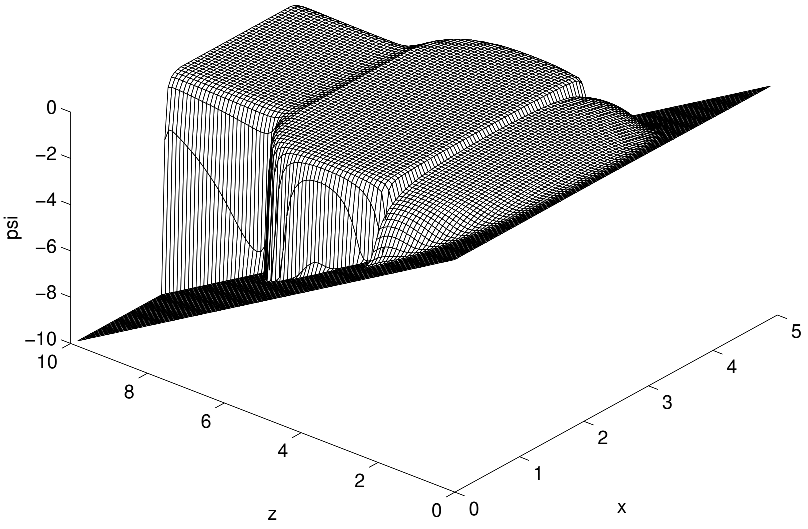

FIG. 3.3. *A plot of the 'exact' solution to Problem 2.*

The numerical results for the default scheme and the adaptive scheme with various δ's are shown in Tables 3.2 – 3.5. For each of the problems and linear methods, two different tolerances to DASPK are used. The error values are scaled l_2 differences as defined in (3.3). We will first look at the results when using GMRES as the linear solver. For each of the two tolerances used, as we reduce δ we can only expect the error to be reduced to a certain level. Any further reduction of δ will only result in a less efficient method. A good example of this behavior can be seen in Table 3.3. For the default method with *tol* = 1e-3, reducing δ below 2.5e-2 does not result in much improvement in the error, but increases the work by about 25 % at the smallest reported δ. Similarly for *tol* = 5e-5, using a δ below 6.25e-3 does not result in much improvement in the error. The results for Problem 1 shown in Table 3.2 are not as dramatic, but a different δ for each of the tolerances produces the best results; $\delta = 3.125e-3$ for a tolerance of 1e-3 and $\delta = 1.563e-3$ for a tolerance of 5e-5. These optimal δ's are also different than the ones for Problem 2. If we use the adaptive scheme with $\delta = 0.1$ for both of these problems, we can get results that are slightly better than the best default when *tol* = 5e-5, and within about 10-15 % for *tol* = 1e-3. The major advantage of the adaptive method is that we were able to get these results without trying many different δ's. It also should be noted that the default δ used in DASPK is 0.05 which did not give satisfactory results for any of the runs.

The numerical results for Bi-CGSTAB are shown in Tables 3.4 and 3.5 and they are slightly different than for GMRES. The first major difference is that for *tol* = 1e-3, the results were not clear which could be a result of Bi-CGSTAB terminating with a slightly less accurate result than GMRES. Therefore, we used *tol* = 5e-4 for our larger tolerance. Another difference is that for Problem 1, the method failed for the larger tolerance with the two largest δ's. Also for this problem, the error is continuing to improve even when $\delta = 1.563e-3$ for the larger tolerance, but for the other tolerance the error is close to optimal when using $\delta = 6.25e-3$. For the adaptive scheme on this problem with $\delta = 0.1$, the results are close to the optimal for the larger tolerance and within

TABLE 3.2
Error and number of function calls for GMRES on Problem 1.

		tol = 1e-3		*tol* = 5e-5	
	δ	F-CALLS	l_2 error	F-CALLS	l_2 error
default	5.000e-02	17160	9.09e-01	54367	3.44e-02
	2.500e-02	21113	7.30e-01	59916	1.36e-02
	1.250e-02	21281	3.32e-01	59100	5.99e-03
	6.250e-03	24655	8.61e-02	53488	4.14e-03
	3.125e-03	23845	4.45e-02	58783	8.49e-04
	1.563e-03	26650	5.14e-02	61509	5.96e-04
adaptive	2.000e-01	25378	4.35e-02	55578	1.36e-03
	1.000e-01	25734	5.07e-02	59885	4.64e-04
	5.000e-02	27066	2.98e-02	64512	4.04e-04

TABLE 3.3
Error and number of function calls for GMRES on Problem 2.

		tol = 1e-3		*tol* = 5e-5	
	δ	F-CALLS	l_2 error	F-CALLS	l_2 error
default	5.000e-02	3378	1.46e-01	7161	2.26e-03
	2.500e-02	3627	4.44e-02	9589	8.14e-04
	1.250e-02	3863	4.14e-02	7857	9.09e-04
	6.250e-03	3998	4.04e-02	7893	3.69e-04
	3.125e-03	4318	3.16e-02	8290	3.51e-04
	1.563e-03	4517	4.40e-02	8625	2.97e-04
adaptive	2.000e-01	4162	4.46e-02	7516	1.42e-03
	1.000e-01	4217	4.23e-02	7752	4.72e-04
	5.000e-02	4891	5.99e-02	8035	8.52e-04

about 10 % for the smaller tolerance. Again, an advantage of the adaptive scheme is that we did not have to try many different δ's to get a near optimal result. The results are very similar for Problem 2 for both the default scheme and the adaptive scheme.

The main conclusions that should be drawn from these results is that the optimal δ for the default scheme varies with tolerance, linear method and problem. But the adaptive scheme is able to either get close or beat the optimal result for all of the problems. It is also true that for the problems shown here, we could choose the smallest δ and get results close to the adaptive scheme. While this may be true for the problems shown here, it can be assumed with some confidence that this will not be true for other problems, or even if a better preconditioner is used for these problems. But the adaptive scheme presented here has a theory behind it which supports the notion that the scheme will be able to respond to these different situations and return a numerical result which is close to the optimal.

4. Conclusions. In this paper we have extended our previous work on numerical methods for solving Richards' equation to two space dimensions. This resulted in the additional complexity of using an iterative linear solver, as opposed to a direct linear solver, to solve the linear systems

TABLE 3.4
Error and number of function calls for Bi-CGSTAB on Problem 1.

	δ	*tol* = 5e-4 F-CALLS	*tol* = 5e-4 l_2 error	*tol* = 5e-5 F-CALLS	*tol* = 5e-5 l_2 error
default	5.000e-02	*fail*	*fail*	67260	1.66e-02
	2.500e-02	*fail*	*fail*	73545	5.84e-03
	1.250e-02	24033	4.87e-01	65247	5.83e-03
	6.250e-03	26299	2.88e-01	72452	4.24e-04
	3.125e-03	30159	1.15e-01	79513	8.11e-04
	1.563e-03	35711	2.26e-02	84012	3.89e-04
adaptive	2.000e-01	29573	2.25e-01	78572	4.56e-04
	1.000e-01	34069	3.13e-02	82726	4.58e-04
	5.000e-02	36042	1.37e-02	88826	3.74e-04

TABLE 3.5
Error and number of function calls for Bi-CGSTAB on Problem 2.

	δ	*tol* = 5e-4 F-CALLS	*tol* = 5e-4 l_2 error	*tol* = 5e-5 F-CALLS	*tol* = 5e-5 l_2 error
default	5.000e-02	3943	8.40e-01	8600	1.81e-03
	2.500e-02	4262	1.82e-01	9039	7.00e-04
	1.250e-02	4476	1.35e-01	9032	3.26e-04
	6.250e-03	4999	6.10e-02	9762	3.61e-04
	3.125e-03	4824	2.77e-02	9836	4.53e-04
	1.563e-03	5337	1.36e-02	10386	8.12e-05
adaptive	2.000e-01	4591	2.54e-01	9523	3.74e-04
	1.000e-01	4779	7.89e-02	9846	7.72e-04
	5.000e-02	5297	3.56e-02	10083	3.52e-04

associated with Newton's method. In this approach, called an inexact Newton method, the selection of the linear tolerance is particularly important. We have presented a theorem that states the conditions on the linear tolerance which guarantee local convergence of the nonlinear iteration. This theorem is slightly different from standard results, because the linear iteration is terminated on absolute residuals rather than relative residuals. The evaluation of the appropriate linear tolerance requires finding the norm of the inverse Jacobian, which is accomplished at minimal cost using a statistical method. Numerical results were shown using both GMRES and Bi-CGSTAB as the linear solver, and these results indicate that our method selects an appropriate linear tolerance. Further research needs to be performed to determine if this method can be applied to more general ODE's and DAE's.

REFERENCES

[1] Kathyrn E. Brenan, Stephen L. Campbell, and Linda R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. Number 14 in Classics in Applied Mathematics. SIAM, Philadelphia, 1996.

[2] Peter N. Brown and Alan C. Hindmarsh. Matrix-free methods for stiff systems of ODE's. *SIAM Journal of Numerical Analysis*, 23(3):610 – 638, 1986.

[3] Peter N. Brown and Alan C. Hindmarsh. Reduced storage matrix methods in stiff ODE systems. *Applied Mathematics and Computation*, 31:40 – 91, 1989.

[4] Peter N. Brown, Alan C. Hindmarsh, and Linda R. Petzold. Using Krylov methods in the solution of large-scale differential-algebraic systems. *SIAM Journal on Scientific Computing*, 15(6):1467 – 1488, 1994.

[5] R. Dembo, S. Eisenstat, and T. Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400 – 408, 1982.

[6] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Number 16 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1995.

[7] C. T. Kelley, C. T. Miller, and M. D. Tocci. Termination of Newton/chord iterations and the method of lines. Technical Report CRSC-TR96-19, North Carolina State University, May 1996. SIAM Journal on Scientific Computing, to appear.

[8] C. S. Kenney and A. J. Laub. Small-sample statistical condition estimates for general matrix functions. *SIAM Journal on Scientific Computing*, 15(1):36 – 61, 1994.

[9] C. T. Miller and C. T. Kelley. A comparison of strongly convergent solution schemes for sharp front infiltration problems. In A. Peters, G. Wittum, B. Herrling, U. Meissner, C.A. Brebbia, W.G. Gray, and G.F. Pinder, editors, *Computational Methods in Water Resources X, Vol. 1*, pages 325–332. Kluwer Academic Publishers, 1994.

[10] Y. Mualem. A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resources Research*, 12(3):513 – 522, 1976.

[11] L. A. Richards. Capillary conduction of liquids through porous media. *Physics*, 1:318 – 333, 1931.

[12] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856 – 869, 1986.

[13] Barry Smith, Petter Bjørstad, and William Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, Cambridge, 1996.

[14] M. D. Tocci, C. T. Kelley, and C. T. Miller. Accurate and economical solution of the pressure-head form of Richards' equation by the method of lines. *Advances in Water Resources*, 20(1):1 – 14, 1997.

[15] Michael D. Tocci. *Numerical Methods for Variably Saturated Flow and Transport Models*. PhD thesis, North Carolina State University, 1997.

[16] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631 – 644, 1992.

[17] M. T. van Genuchten. Predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, 44(5):892 – 898, 1980.